Felipe Araújo de Lima
João Vitor Almeida de Lima

GEM - Sistema de Gerenciamento de Programas de Monitoria

Recife

2019

Felipe Araújo de Lima
João Vitor Almeida de Lima

GEM - Sistema de Gerenciamento de Programas de Monitoria

Trabalho de conclusão de curso apresentado ao Programa de TADS do DASE do Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco como requisito parcial para conclusão do curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Instituto Federal de Educação Ciência e Tecnologia de Pernambuco - IFPE - Recife

Curso Tecnologia em Análise e Desenvolvimento de Sistemas

Orientador: Me. Marcos André da Silva Costa

Recife

2019

Ficha elaborada pela bibliotecária Ana Lia Evangelista CRB4/974

L732g
2019 Lima, Felipe Araújo de.
GEM – Sistema de gerenciamento de programas de monitoria / Felipe Araújo de Lima, João Vitor Almeida de Lima --- Recife : O autor, 2019.
55f. il. Color.

TCC (Tecnólogo em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Pernambuco, DASE 2019.

Inclui Referências.

Orientador: Profº Me. Marcos André da Silva Costa.

1. Monitoria. 2. Sistema WEB. 3. Gerenciamento. 4. Desenvolvimento. 5. Scrum. 6. Java. I. Título.

CDD 005.7406 (21ed.)

Felipe Araújo de Lima
João Vitor Almeida de Lima

GEM - Sistema de Gerenciamento de Programas de Monitoria

Trabalho de conclusão de curso apresentado ao Programa de TADS do DASE do Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco como requisito parcial para conclusão do curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Trabalho aprovado. Recife, 06 de Junho de 2019:

Me. Marcos André da Silva Costa
Marcos André da Silva Costa

Dr. Paulo Mauricio Goncalves Junior
Convidado 2

Esp. Esneilton Oliveira do Nascimento
Convidado 3

Recife
2019

Agradecimentos

Agradecemos a Deus por nos dar saúde, força e sabedoria para superar as dificuldades de cada dia. A nossos familiares e amigos, em especial Rapha Ursão, Danilo Dino e Iury Fafá. A todos os professores que se dedicaram, em especial ao nosso professor orientador Marcos André pela paciência e compromisso conosco.

"Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar:

1. Indivíduos e interação entre eles mais que processos e ferramentas

2. Software em funcionamento mais que documentação abrangente

3. Colaboração com o cliente mais que negociação de contratos

4. Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda." (Manifesto para o desenvolvimento ágil de software, 2001, Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern,Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas)

Resumo

Monitoria é um programa muito importante para o desenvolvimento e formação de estudantes de cursos técnicos e de graduação. Nesta obra descrevemos o desenvolvimento de um sistema para gerenciar programas de monitoria, em especial em Institutos Federais de Educação. O GEM é um sistema que possui maior compatibilidade com navegadores atuais e que permite a gerência por parte do aluno e professores de todo o processo de monitoria de forma facilitada e reduzindo a utilização de papel durante todo o processo. Utilizamos como tecnologias para o desenvolvimento do GEM: Java Web, Javascript, PostgreSQL, entre outras; Como metodologia de desenvolvimento, escolhemos o Scrum de forma adaptada para a nossa realidade; Testes automatizados também foram empregados na verificação e validação do projeto.

Palavras-chave: Monitoria. Sistema Web. Gerenciamento. Desenvolvimento. Scrum. Java.

Abstract

Teacher assistance programs are of great importance on the academic development of undergraduate students. In this text, we describe the development of a system to manage teacher assistance programs, especially for Federal Education Institutes. The GEM is a system that possesses a higher compatibility with current browsers and that allows the management of the whole teaching assistance processes, made by students and teachers, facilitating the interaction between all the stakeholders and reducing the consumption of paper. We utilized Java EE, Javascript and PostgreSQL as technologies to the development of the web system. As development methodology, we have chosen Scrum in an adapted way to reflect our reality. Automatized tests were also utilized in the validation and verification of the project.

Keywords: Teacher Assistance. Web System. Management. Development. Scrum. Java.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
TADS	Tecnologia em Análise e Desenvolvimento de Sistemas
TI	Tecnologia da Informação
IFPE	Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBDOR	Sistema de Gerenciamento de Banco de Dados Objeto Relacional
JPA	Java Persistence API
EJB	Enterprise Java Beans
API	Application Programming Interface (Interface de Programação de Aplicativos)

Sumário

1 Introdução

Desde os primórdios da civilização compreende-se que o ensino não é tarefa única e exclusiva do professor, já na Universidade Medieval observava-se a presença de monitores, denominados 'repetidores' que reproduziam a matéria desenvolvida por seus mestres (Ullman e Bohnen, 1994, p. 43). Hoje, programas de monitoria estão presentes em disciplinas de cursos superiores, visando incentivar o aprofundamento de discentes em uma área; Introduzir o contato acadêmico/profissional, além de proporcionar assistência adicional para atuais alunos da disciplina.

A atual lei de Diretrizes e Bases da Educação Nacional (Lei nº 9.394, de 20 de dezembro de 1996), se refere à existência do monitor nos seguintes termos: "os discentes da educação superior poderão ser aproveitados em tarefas de ensino e pesquisa pelas respectivas instituições, exercendo funções de monitoria, de acordo com seu rendimento e seu plano de estudos"(Art84). Dada a importância do programa de monitoria e aos últimos avanços da tecnologia, é comum encontrar sistemas que auxiliam este processo.

Este trabalho visa introduzir um novo sistema de gerenciamento de monitoria, tendo como base o atual sistema em atividade no Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (Campus Recife) e o Edital de Programa Institucional de Monitoria da mesma instituição. Este novo sistema, buscará atender à partes do processo de monitoria que ainda são feitas completamente sem o auxílio da tecnologia, como por exemplo a administração de relatórios de atividades e resultado de inscrições.

1.1 Motivação

O Edital de Programa Institucional de Monitoria do Instituto Federal de Pernambuco - Campus Recife (IFPE, 2017) define o programa como um incentivo à formação acadêmica que visa à ampliação dos espaços de aprendizagem, à melhoria da qualidade de Ensino e ao desenvolvimento da autonomia e formação integral dos estudantes.

Ainda segundo o edital, este programa tem por objetivo: "Promover o desenvolvimento de aptidões para a docência; Intensificar e assegurar a complementação da formação acadêmica do estudante monitor; Possibilitar o compartilhamento de conhecimentos através da interação entre estudantes e professores nas atividades acadêmicas relativas as atividades do ensino; Proporcionar o aprofundamento dos conhecimentos teóricos e metodológicos que aliados à práxis pedagógica, forneçam subsídios para futura inserção no mundo do trabalho; Contribuir para a melhoria da qualidade de ensino favorecendo a redução dos problemas de repetência e evasão dos estudantes".

Dados os possíveis benefícios que serão alcançados e a importância de um programa como esse no processo de aprendizagem, faz-se necessária a gerência de forma eficiente desse programa, que pode ser alcançado com inclusão de um sistema para gerencia-lo.

1.2 Justificativa

Várias etapas são realizadas no Programa de Monitoria, desde a inscrição dos componentes curriculares, inscrições dos alunos até a entrega do relatório final e a emissão da Declaração de Monitoria. Atualmente o sistema de monitoria do IFPE, apenas acompanha e presta suporte as etapas de inscrição, não possuindo sequer a divulgação do resultado de seleção dos monitores. Com a democratização e a facilitação do acesso à internet, o acesso às aplicações web tornou-se simplificado, com isso, é de suma importância que o sistema de monitoria acompanhe e auxilie tanto o aluno quanto o professor durante todas as etapas que constituem este processo.

Atualmente, o portal de monitoria do IFPE não oferece nenhum recurso de responsividade para diferentes dispositivos, além de não possuir suporte a diferentes browsers, limitando-se apenas ao Mozilla Firefox.

Após a avaliação da atual solução e da necessidade levantada pela comisssão de monitoria, concluímos que faz-se necessária a atualização e reformulação completa do sistema de monitoria atual para melhor prestar suporte tanto à docentes quanto à discentes, acompanhando-os durante as várias etapas do processo de monitoria.

2 Objetivos

2.1 Objetivo Geral

O desenvolvimento de uma ferramenta que venha suprir as necessidades acadêmicas e administrativas geradas pelo Programa de Monitoria, auxiliando de forma objetiva todas as etapas do processo, desde a inscrição até o término do Programa.

2.2 Objetivo específico

- Entendimento do regulamento de monitoria;
- Escolha de ferramentas que facilitem e tornem possíveis implementar as funcionalidades do sistema;
- Planejamento das iterações
- Utilização de metodos ágeis no desenvolvimento da solução;
- Testes automatizados para verificação do sistema;
- Desenvolvimento das funcionalidades do sistema.

3 Fundamentação Teórica

Neste capítulo inicialmente serão abordados os fundamentos teóricos que embasaram o desenvolvimento deste trabalho. Inicialmente serão aboradados princípios básicos do desenvolvimento ágil de software. Em seguida, serão abordadas metodologias de testes que foram aplicadas ao longo do desenvolvimento do sistema. Na sequência, serão vistos conceitos mais específicos das tecnologias utilizadas no trabalho.

3.1 Desenvolvimento Ágil

Desenvolvimento Ágil de software define várias metodologias que diferem bastante do modelo de desenvolvimento em cascata, muito utilizado no passado. Segundo o manifesto ágil, quatro valores são adotados:

1. Indivíduos e interação entre eles mais que processos e ferramentas;
2. Software em funcionamento mais que documentação abrangente;
3. Colaboração com o cliente mais que negociação de contratos;
4. Responder a mudanças mais que seguir um plano.

3.1.1 Equipe e Processo de Desenvolvimento

Com base nestes valores e foco em entregar software funcional ao final de cada semana, membros de times ágeis possuem uma organização e definição de resposabilidades diferentes. Nestes tipos de projetos e times, responsabilidades de cada indivíduo se estendem a círculos de conhecimento que podem alcançar áreas fora de sua especialidade.

Co-localização, capacidade de auto-organização e uma forte participação do cliente são chaves para uma boa organização e fluidez de um time ágil ao longo do processo de desenvolvimento de um projeto. Ao contrário de metodologias de desenvolvimento tradicionais, durante um processo de desenvolvimento ágil são executadas atividades de forma contínua, integrando Análise, Teste, Design (Projeto) e Codificação.

3.1.2 Scrum

O Scrum é definido como "Um framework dentro do qual pessoas podem tratar de resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível"(SCHWABER; SUTHERLAND, 2016). Foi criado

por Ken Schwaber e Jeff Sutherland, dois participantes do Manifesto Ágil. Fundamentado no empirismo para tomada de decisões, ou seja, nas experiências.

No Scrum, a equipe é composta por pessoas que adotam os seguintes papéis:

1. Scrum master: É o integrante responsável pelo apoio da equipe durante as sprints, agindo como facilitador no daily Scrum e removendo obstáculos levantados pela equipe durante estas reuniões, além de ajudar a equipe a adotar as práticas e entender as regras do Scrum.

2. Product Owner: Representa o cliente na equipe, este integrante é quem define quais são as prioridades do que deve ser feito e, no caso de dúvidas sobre a execução de algo ou o resultado final de alguma história, é o Product Owner que é responsável por responder quaisquer questionamentos.

3. Time de desenvolvimento: Com número recomendado entre três e nove integrantes, são responsáveis por implementar as histórias que compõem o backlog do produto. Ainda sobre o tamanho do time de desenvolvimento, um número menor que três participantes pode gerar dificuldade ao se tentar gerar um incremento potencialmente liberável.

3.1.2.1 Backlog do Produto

O backlog é um artefato que define todas as necessidades de implementações e alterações futuras no produto ao longo da vida do projeto. É mantido pelo Product Owner, sendo alterado constantemente visando sempre se adequar às necessidades do cliente. Segundo definição de Schwaber: "O backlog do produto lista todas as características, funções, requisitos, melhorias e correções que formam as mudanças que devem ser feitas no produto nas futuras versões"(SCHWABER; SUTHERLAND, 2016).

3.1.2.2 Sprints

Sprints são as iterações no Scrum, geralmente tem duração de duas semanas, mas podem possuir duração de até um mês. Com o início da Sprint sendo marcado pela definição de um backlog de funcionalidades à serem implementadas, e o término com a implementação concretizada destas atividades. Uma Sprint possui algumas etapas definidas para ajudar na organização do planejamento e revisão de resultados, são estas por ordem de acontecimento:

1. Reunião de planejamento da Sprint: Composta por todos os membros do time Scrum, o Product Owner deve informar quais atividades do backlog são as prioridades e esclarecer quaisquer detalhes sobre o resultado final esperado pelo cliente.

Com as prioridades detalhadas em mãos, o time de desenvolvimento se reúne para compor o backlog da sprint, colocando o que é possível ser entregue durante a duração adotada.

2. Reunião diária: Tem como maior objetivo transparecer os maiores impedimentos enfrentados por cada membro do time de desenvolvimento. Também visa deixar claro para todos os membros quais atividades foram feitas por cada membro, resultando na consciência do time inteiro sobre o atual estado da Sprint e dos objetivos mais críticos do dia. Geralmente é feita em pé para acelerar mais e possui duração máxima de quinze minutos.

3. Reunião de revisão: Com duração de até 4 horas, o time Scrum se reúne com os stakeholders e o Product Owner para apresentar o que está pronto. Com base no estado atual do backlog, são decididas as próximas coisas à se fazer podendo haver mudanças no backlog do produto.

4. Reunião de retrospectiva: Nesta última etapa, uma auto avaliação é feita pelo time Scrum e com base nela é criado um plano de melhorias para a próxima sprint. Nesta avaliação são levantados pontos em relação às pessoas e suas relações, ferramentas e processos utilizados ao longo da sprint, com foco em potenciais melhorias.

3.2 Kanban

Kanban é uma palavra de origem japonesa que significa "Cartões sinalizadores", estes cartões são utilizados no sistema de mesmo nome chamado Kanban para representar tarefas a serem feitas (ANDERSON, 2011).

A Toyota foi a criadora desse sistema utilizado para sinalizar passos em seus processos de fabricação. Como a característica mais forte desse sistema é a visualização, as equipes começaram a se comunicar com mais facilidade, organizando e melhorando o que e quando deveria ser feito. Relacionado com o conceito de sistemas de produção puxados onde cada operação do processo é alimentada pela demanda da etapa anterior.

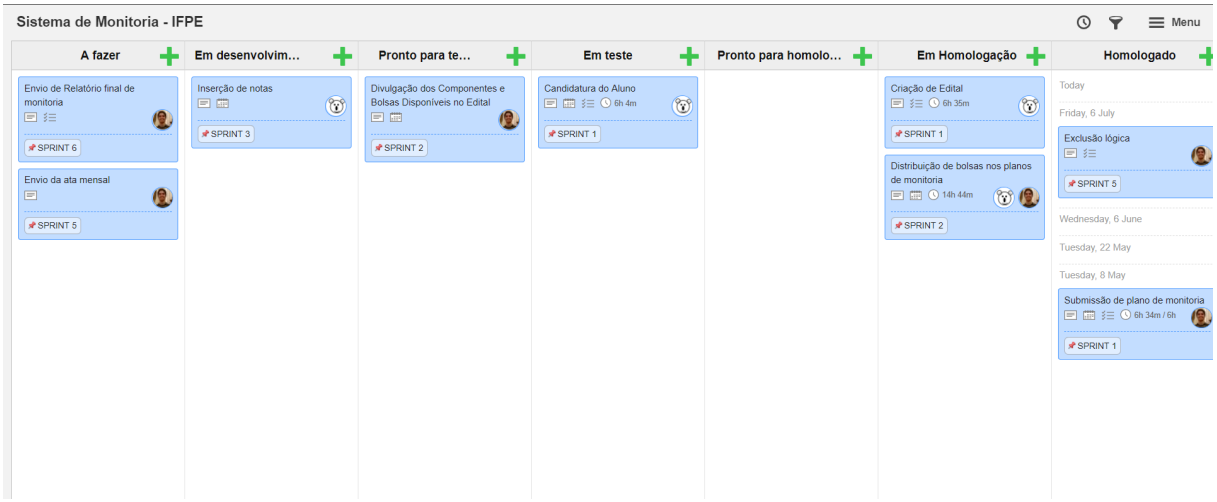
Foi desenvolvido então, o método Kanban, que é aplicado no auxílio de equipes ágeis no desenvolvimento de software. Esse método é definido por um conjunto de princípios e práticas chaves.

A primeira delas é representação das tarefas e do fluxo do trabalho, seja em um painel na parede ou em um painel eletrônico. É comum escrever nos cartões desse painel as tarefas a serem desempenhadas no trabalho. As etapas necessárias a cumprir essas tarefas ficam em colunas do painel, na ordem em que elas forem executadas, contendo a fila de entrada, etapas do processo, e fila de tarefas concluídas. É importante criar um

painel que reflita o processo de desenvolvimento atual, com a intenção de melhorá-lo ao longo do tempo.

Na Figura 1 está o quadro kanban utilizado no desenvolvimento do GEM. O quadro foi dividido em sete etapas. Histórias selecionadas no backlog para a próxima entrega são inseridas na coluna "A fazer". Qualquer um dos desenvolvedores pode selecionar uma tarefa a movendo para a seção "Em desenvolvimento". Estando pronta para teste, um outro membro da equipe irá testar o que foi desenvolvido, transferindo a tarefa para a seção "Em teste". Nas fases "Em desenvolvimento"e "Em teste", existe a limitação da quantidade de tarefas por vez para evitar gargalos no desenvolvimento. Após testada, se aprovada, a tarefa estará "Pronta para homologação", caso contrário voltará para "Em desenvolvimento"para correções e ajustes. Na fase "Em homologação"a história é validada pelos stakeholders[1](no caso, o orientador), caso a tarefa seja "Homologada", é dada como finalizada, caso contrário voltará para fase "Em desenvolvimento".

Figura 1 – Quadro Kanban durante o desenvolvimento do GEM



Fonte: Próprios autores

Outra prática chave do método kanban é a limitação dos trabalhos em progresso, garantir que seja definido um limite do número de tarefas em cada etapa do fluxo de trabalho de cada vez. Quando uma tarefa avança, abre espaço para uma nova tarefa chegar e entrar na etapa de desenvolvimento. Os limites de trabalho em processo para as etapas do fluxo de trabalho devem ser definidos com um número médio de itens por pessoa, par de desenvolvedores ou equipe pequena e colaborativa. Não deve ser desperdiçado um tempo excessivo tentando determinar o limite perfeito, basta escolher um número que está perto o suficiente, e fazer ajustes ao decorrer do processo se necessário. Nem todos os passos devem ter um limite.

[1] Pessoas e grupos mais importantes para um planejamento estratégico do desenvolvimento do software, ou seja, as partes interessadas.

Segundo David Anderson os beneficios alcançados com a utilização do método Kanban são a otimização dos processos existentes, a entrega com qualidade superior, a exatidão da previsibilidade do tempo de entrega, a satisfação dos funcionários, proporciona folga para permitir melhorias, simplifica a priorização, fornece transparência no projeto e operação do sistema e ainda possibilita o surgimento de uma organização de "alta maturidade"(ANDERSON, 2011).

3.3 Desenvolvimento Orientado por Comportamento

O Desenvolvimento Orientado por comportamento (BDD - Behavior-driven development) baseia-se no desenvolvimento orientado a testes (TDD - test-driven development) (ROSE; WYNNE; HELLESØY, 2015), trabalhando de fora para dentro, começando com um teste de aceitação que descreve o comportamento do sistema do ponto de vista do cliente. Os testes de aceitação devem ser escritos como exemplos que qualquer pessoa da equipe pode compreender. O processo de escrita desses exemplos é utilizado para se obter dos stakeholders um retorno e descobrir se a construção e o planejamento estão sendo feitos da maneira certa antes de começar de fato a programação. Ao se fazer isso, é feito um esforço deliberado para desenvolver uma linguagem compartilhada e onipresente para falar sobre o sistema.

Em sua essência, o BDD é simplesmente a idéia de que o desenvolvimento de software deve ser conduzido tanto por proficiência técnica quanto por interesses do cliente. Na prática, o BDD faz uso de softwares especializados para atingir os objetivos desejados. A principal ferramenta do método é a linguagem específica de domínio simples (também conhecida como DSL - domain-specific language). Em vez de linhas complexas de código, essa linguagem usa palavras normais e construções lógicas para expressar como o software deve se comportar.(BEHAVIOUR-DRIVEN.ORG, 2016)

O BDD segue o princípio básico de que cada unidade de software deve ser testada individualmente. O processo geralmente é assim:

1. Um teste é projetado para a unidade de software específica
2. O teste é feito para falhar
3. A unidade é então implementada no teste
4. O teste é feito novamente, verificando se a implementação da unidade faz com que seja bem-sucedida

Este esboço básico permite o teste de software de alto e baixo nível, bem como qualquer outro teste entre eles. Ao usar a metodologia BDD, os testes devem ser especificados em termos do comportamento desejado da unidade em questão. Esse comportamento

se resume nos requisitos definidos pelo cliente que encomendou a criação do software. (BEHAVIOUR-DRIVEN.ORG, 2016)

3.3.1 Cucumber

O Cucumber é uma ferramenta de código aberto criada com o objetivo de dar suporte ao BDD. Ele lê as especificações executáveis escritas em texto simples e valida se o software faz o que essas especificações dizem. As especificações consistem em vários exemplos ou cenários (CUCUMBER.IO, 2018). Por exemplo:

Figura 2 – Exemplo de cenário Cucumber

```
# language: pt
Funcionalidade: Inserção de Notas
  Esse cenário descreve a inserção das notas da seleção para a monitoria
  O professor acessa a listagem dos alunos inscritos em um dos seus componentes
curriculares

  Contexto:
      Dado que o usuário está logado com perfil de professor
      E esteja na página de gerencia dos planos de monitoria
      E seleciona a opção de inserir notas

  Cenario: notas diferentes classificação automática
      Quando o professor inserir as notas de seleção e as medias e salvar
      Então deve mostrar a classificação atualizada de cada aluno

  Cenario: aluno possui reprovação
      Quando o professor indicar que aluno possui reprovação e salvar
      Então deve desclassificar o aluno
```

Fonte: Próprios autores

Cada cenário é uma lista de etapas para o Cucumber trabalhar. Um conjunto de definições de etapas é mapeado a partir de linguagem legível para código que executam qualquer ação que esteja sendo descrita. Em um conjunto de testes maduro, a execução da etapa em si, provavelmente será apenas uma ou duas linhas de código, que utilizam uma biblioteca de suporte para executar tarefas comuns no sistema. Às vezes, isso pode envolver o uso de uma biblioteca de automação, como a biblioteca de automação do navegador Selenium, para interagir com o próprio sistema.

O Cucumber verifica se o software está em conformidade com a especificação e gera um relatório indicando sucesso ou falha para cada cenário. Para que o Cucumber entenda os cenários, eles devem seguir algumas regras básicas de sintaxe, chamadas Gherkin.

Gherkin é um conjunto de regras gramaticais que torna o texto estruturado o suficiente para o Cucumber entender, porém, simples o suficiente para que qualquer pessoa

que participe do processo de desenvolvimento, incluindo o cliente, consiga ler e entender. Esse texto também serve como documentação para o software que está sendo desenvolvido. O cenário acima (Ver figura Figura 2.) está escrito em Gherkin.

O Gherkin serve vários propósitos:

1. Especificação executável não ambígua
2. Teste automatizado usando Cucumber
3. Documenta como o sistema realmente se comporta

A gramática do Cucumber existe em diferentes línguas para muitos idiomas falados, para que seja possível utilizar as palavras-chave em seu próprio idioma, inclusive português. Os documentos da Gherkin são armazenados em arquivos de texto ***.feature*** e são tipicamente versionados no controle de código-fonte ao lado do software (CUCUMBER.IO, 2018).

3.3.2 Selenium

O Selenium é um conjunto de ferramentas para automatizar navegadores da web em várias plataformas. (SELENIUMHQ.ORG, 2018) Podendo ser utilizado para automatizar aplicativos da web para fins de teste.

O Cucumber não é muito mais do que uma ferramenta que pode analisar arquivos de recursos do Gherkin e executar definições de etapas. Não sabe como falar com bancos de dados, aplicativos da web ou qualquer sistema externo. Para isso é necessário a utilização de bibliotecas e ferramentas, usando as definições de etapa e códigos de suporte para se conectarem a sistemas externos.

A biblioteca Java mais popular para interagir programaticamente com um aplicativo da Web é o Selenium WebDriver. Ele fornece uma API para acessar páginas da Web e interagir com elas de uma maneira semelhante à de um usuário real - inserindo texto em campos e áreas de texto, marcando caixas de seleção, clicando em links e botões e assim por diante. O Selenium WebDriver permite que você conecte drivers diferentes que executam essas interações em vários navegadores, como Firefox, Internet Explorer e Chrome. Com isso, é possível realizar a automatização dos testes escritos em Gherkin(CUCUMBER.IO, 2018).

4 Desenvolvimento

O processo de desenvolvimento do GEM foi iniciado durante a disciplina de Desenvolvimento de Software Corporativo já com o intuito de evoluir para um projeto de conclusão de curso. O professor e orientador Marcos André teve papel de dono do produto, enquanto os autores faziam o papel do time de desenvolvimento.

Ao longo da disciplina de Projeto de Desenvolvimento de Software Corporativo foi definido um roadmap[1] inicial do produto, assim como a duração de duas semanas por sprint. Estas sprints eram iniciadas por uma reunião onde era realizada uma revisão do que havia sido alcançado anteriormente, sendo mostrados os testes Cucumber em execução, adequando o roadmap do produto e as próximas features à serem desenvolvidas com base no feedback do dono do produto. Ao longo da execução da sprint, semanalmente eram feitas reuniões para discussão de dúvidas técnicas e de negócio com o professor e orientador Marcos André, e a redistribuição de atividades para que o objetivo da sprint fosse alcançado.

4.1 Papéis

A aplicação possui três tipos de perfil para atender as especificações da comissão responsável pelo processo do programa de monitoria. A definição de perfil é realizada automaticamente durante o cadastro no sistema através do domínio do e-mail. Os domínios associados a cada perfil são parametrizados nos arquivos de configuração da aplicação.

4.1.1 Aluno

Com o perfil de Aluno é possível se candidatar a uma vaga do programa de monitoria, enviar atas mensais, construir e submeter o relatório final.

4.1.2 Professor

O usuário Professor é capaz de cadastrar novos componentes curriculares, submeter planos de monitoria para componentes em que for responsável, inserir as notas de seleção dos alunos para monitoria, receber as atas mensais de frequência dos monitores e aprovar relatórios finais de seus componentes. Além das funcionalidades disponíveis para um usuário com perfil de Professor, caso este seja classificado como coordenador, ele tam-

[1] Ferramenta visual e descritiva que aponta como será o produto ou projeto a cada período de sua evolução

bém é responsável por aprovar o plano de monitoria indicando a quantidade de vagas para voluntários e bolsistas.

4.1.3 Comissão

A comissão além ter as mesmas permissões que o perfil de professor, pode também adicionar cursos, escolhendo o seu coordenador; Criar e editar editais; Indicar a quantidade de vagas e bolsas por curso no edital; Acessar relatórios e gerenciar classificados. Os dominíos de e-mail para comissão e professor são compartilhados, por isso é necessário indicar quais são os professores que possuem perfil de comissão. A comissão tem o poder de promover um professor à um membro da comissão.

4.1.4 Público

Na página inicial do GEM é possivel acessar os resultados sem a necessidade de ter perfil no sistema, dando mais publicidade aos resultados e informações divulgadas, assim, oferencendo mais transparência ao processo. A única exigência de segurança para acessar dados públicos é uma validação por CAPTCHA[1].

[1] Utilizado como uma ferramenta anti-spam é o Acrônimo da expressão "Completely Automated Public Turing test to tell Computers and Humans Apart"(Teste de Turing público completamente automatizado para diferenciação entre computadores e humanos)

4.2 Permissões

A tabela 1 detalha as permissões de cada perfil no GEM:

Tabela 1 – Permissões do GEM

Funcionalidade	**Aluno**	**Professor**	**Comissão**
Gerência de cursos	Não	Não	Sim
Gerência de componentes curriculares	Não	Sim	Sim
Gerência de editais	Não	Não	Sim
Distribuir Bolsas por Curso	Não	Não	Sim
Submeter Plano de Monitoria	Não	Sim	Sim
Distribuir Bolsas por Plano de Monitoria	Não	Sim*	Sim
Homologar Plano de Monitoria	Não	Sim*	Sim
Candidatura a monitoria	Sim	Não	Não
Homologar Candidatos	Não	Não	Sim
Inserir Notas de Avaliação	Não	Sim	Sim
Submeter ata mensal	Sim**	Não	Não
Aprovar frequência de ata mensal	Não	Sim	Sim
Receber frequência	Não	Não	Sim
Visualizar frequências recebidas	Não	Sim	Sim
Enviar relatório final	Sim**	Não	Não
Homologar relatório final	Não	Sim	Sim
Visualizar relatórios finais recebidos	Não	Sim	Sim

*Apenas se o Professor for coordenador de curso.
**Apenas se o aluno for monitor.

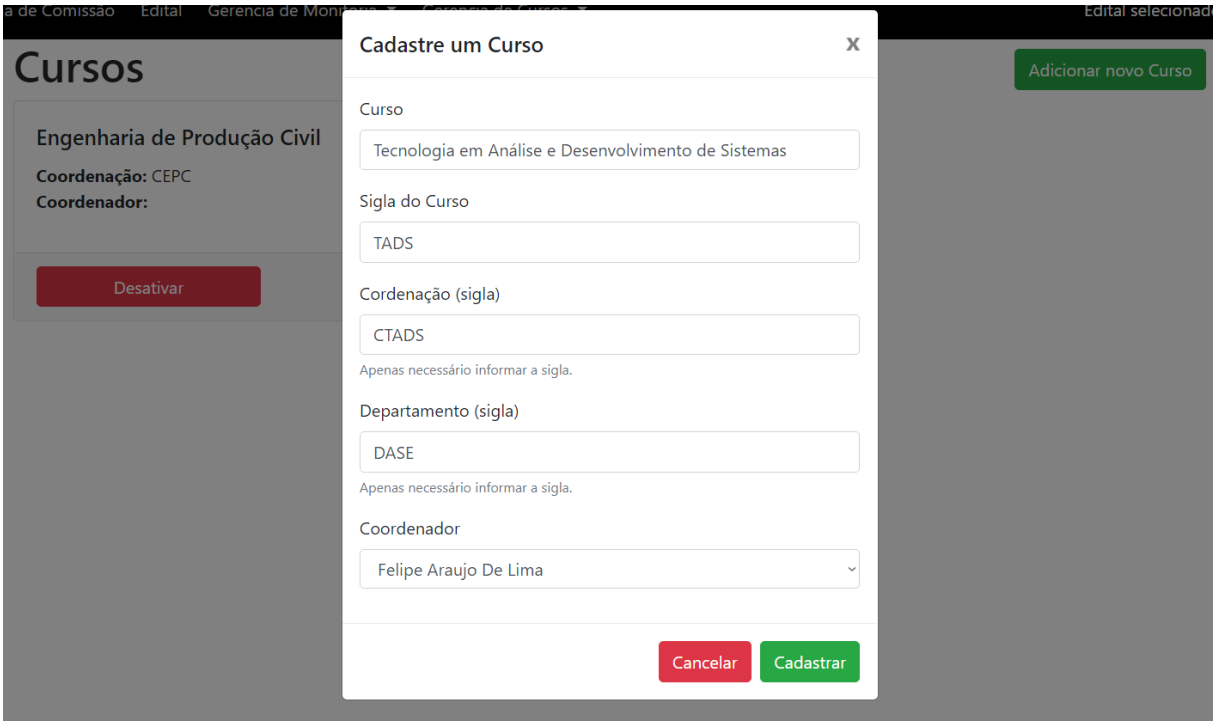
Fonte: Tabela elaborada pelos autores.

4.3 Funcionalidades

4.3.1 Gerenciar cursos

Na Gerência de Cursos é possivel realizar o cadastro, edição e desativação de cursos no sistema por parte da comissão. Nos cadastros também é informado o coordenador de cada curso, vinculando algumas permissões específicas para o usuário coordenador.

Figura 3 – Cadastro de curso

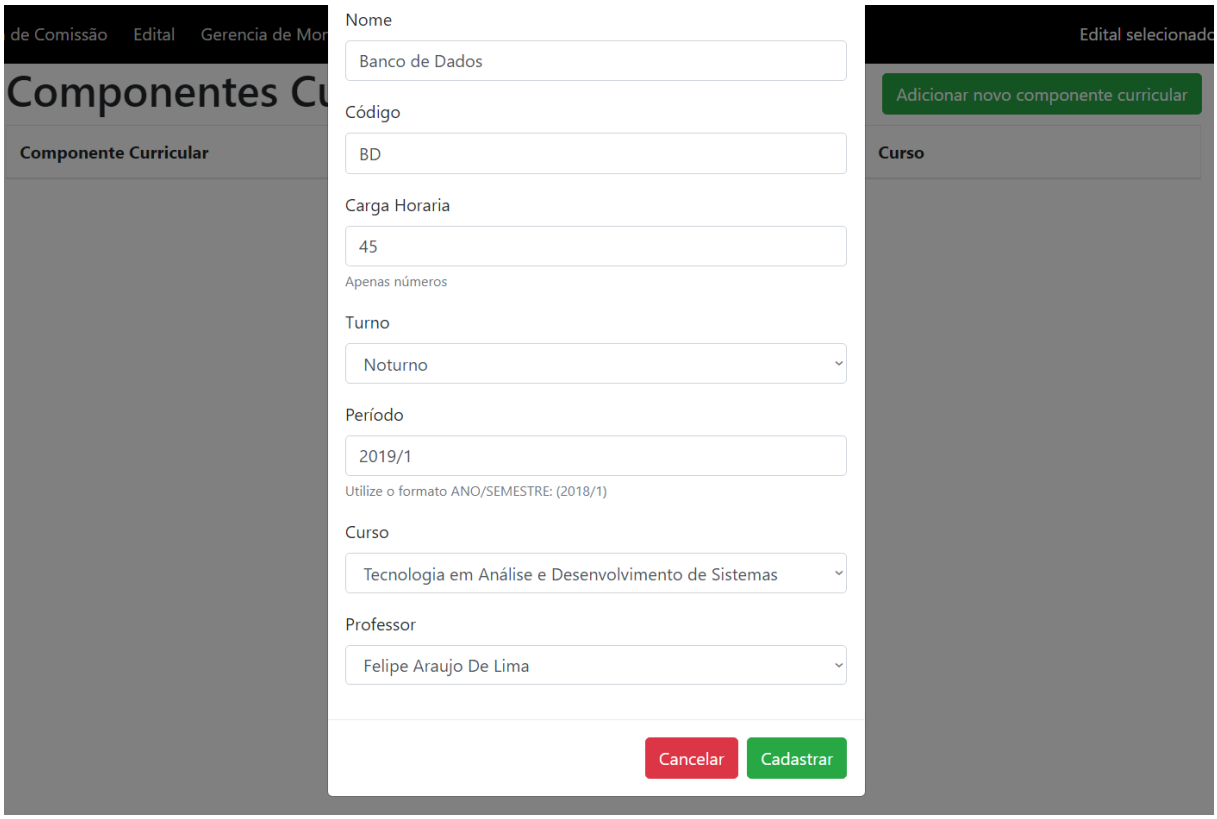


Fonte: Imagem elaborada pelos autores

4.3.2 Gerenciar componentes curriculares

Na tela de Gerência de Componentes Curriculares a comissão, os coordenadores e professores podem adicionar componentes curriculares, indicando qual o curso e o professor responsável. Também é possivel registrar a carga horária e o turno que servirá para consulta do candidato à monitoria.

Figura 4 – Cadastro de componente curricular

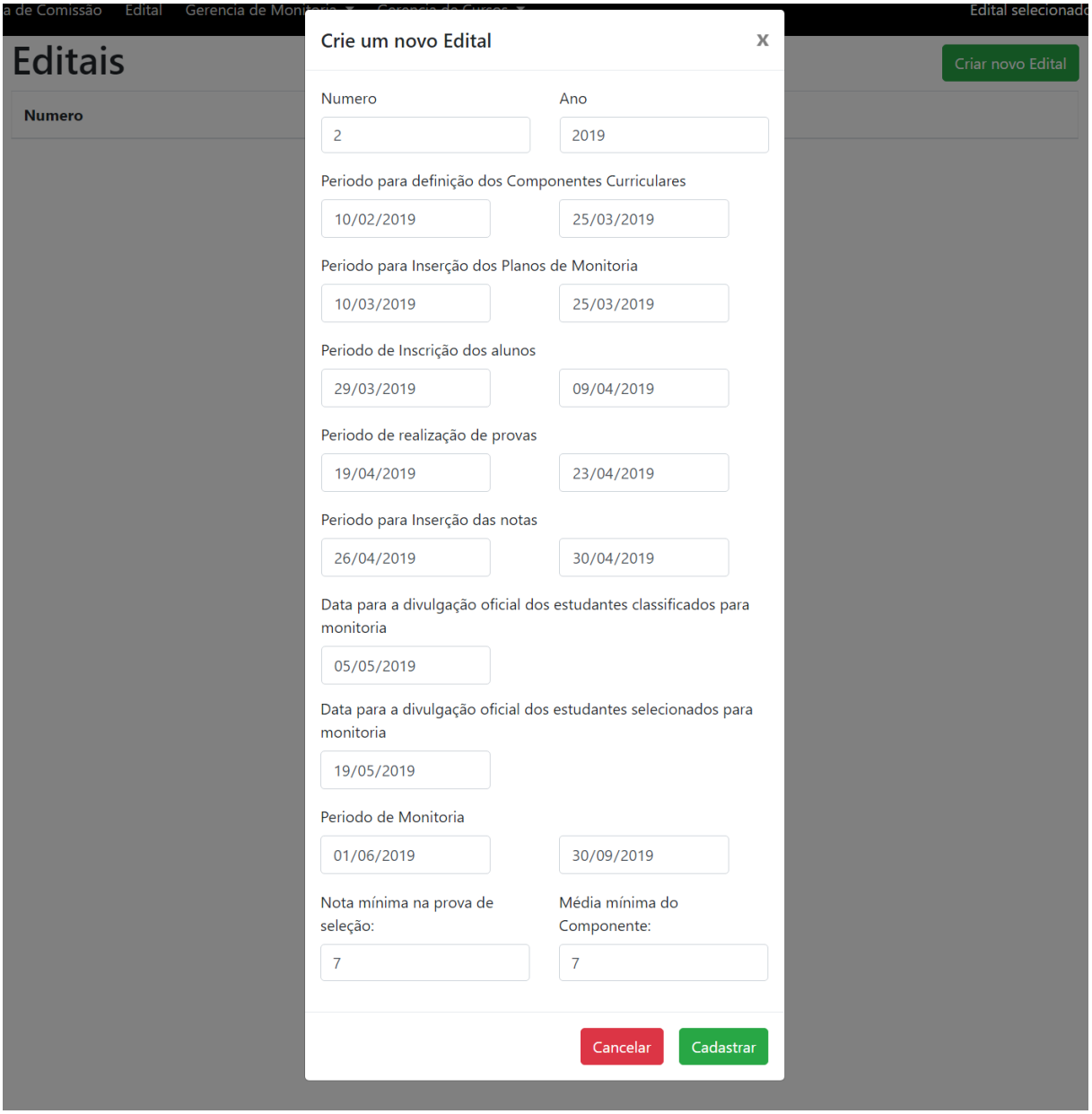


Fonte: Imagem elaborada pelos autores

4.3.3 Gerenciar editais

Todo o controle temporal do programa de monitoria é realizado através da criação e alteração do Edital. Ao criar ou editar um edital é possivel indicar as datas iniciais e finais de cada etapa do processo, conforme imagem 5. Além disso, deve ser inserido as notas minimas para a prova de seleção e a média minima do componente.

Figura 5 – Criação de edital



Fonte: Imagem elaborada pelos autores

4.3.4 Distribuição de bolsas por curso

Após a criação do edital é possivel registrar a quantidade de bolsas por curso. Por exemplo, na imagem 6 foram disponibilizadas 3 bolsas para os cursos. Abaixo o teste de comportamento escrito para essa funcionalidade:

```
Funcionalidade: Distribuir bolsas para cursos

Contexto:
    Dado que o usuario esta logado como comissao
    E esteja na pagina de edital
    E crie um edital valido

Cenario: Definir quantidade de bolsas para um curso
    Quando gerenciar um edital especifico
    E definir quantidade de bolsas para um curso
    Entao o sistema deve especificar aquela quantidade para o esquema

Cenario: Criar um lancamento quando ja houver outro criado
    Quando gerenciar um edital especifico
    E ja existir um lancamento criado para determinado curso
    E escolher lancar bolsas para aquele curso
    Entao o sistema informa que ja existe um esquema criado
```

Figura 6 – Distribuição de bolsas

Fonte: Imagem elaborada pelos autores

Só é possivel deixar um edital vigente após a comissão indicar a quantidade de bolsas por curso, mesmo que a quantidade disponibilizada seja zero. A opção para alterar a vigência é acessada na função para alterar o edital.

```
Funcionalidade: Distribuir bolsas para cursos

Contexto:
    Dado que o usuario esta logado como comissao
    E esteja na pagina de edital
    E crie um edital valido

Cenario: Definir um edital como vigente sem distribuir bolsas
    Quando existirem cursos sem bolsas explicitamente lancadas
    E definir um edital como vigente
    Entao o sistema deve informar que se deve definir
    uma quantidade de bolsas
```

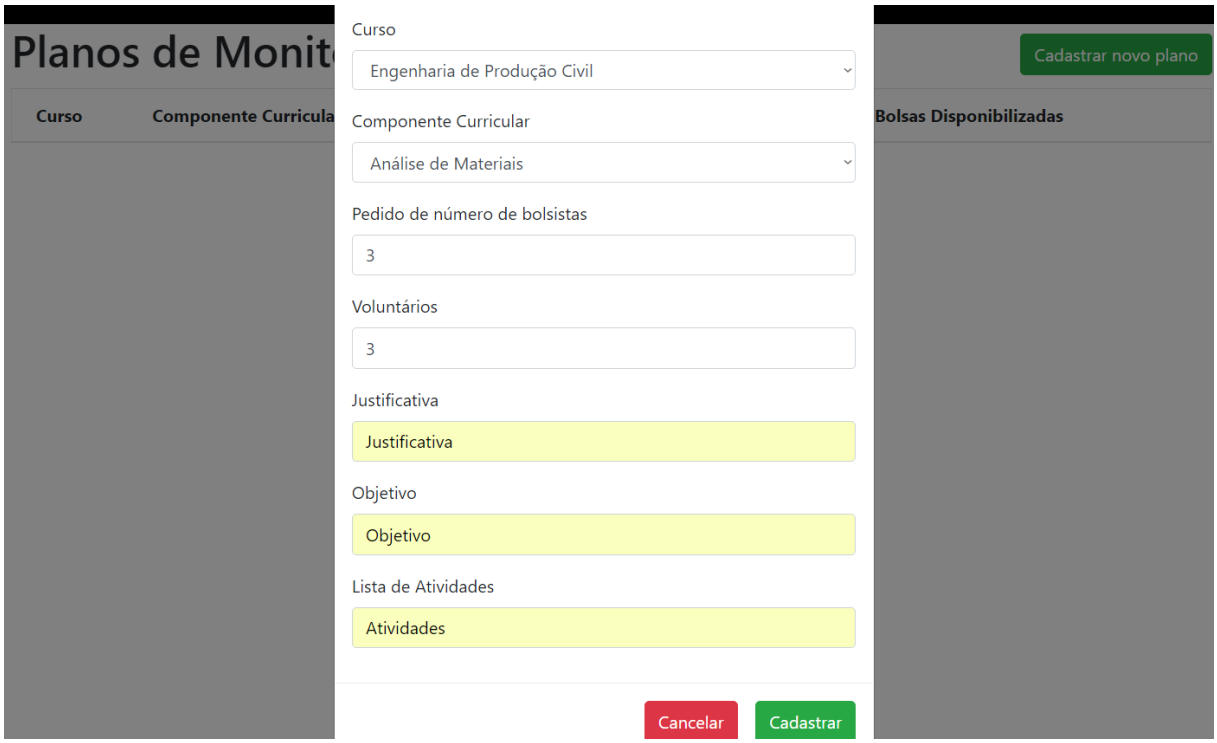
Figura 7 – Alterar vigência

Fonte: Imagem elaborada pelos autores

4.3.5 Gerenciar planos de monitoria

Em gerência de planos de monitoria, os professores podem realizar no periodo estabelecido no edital, o pedido de monitores para o(s) componente(s) em que são responsável atraves da submissão de um plano de monitoria. Definindo a quantidade de voluntários e bolsistas conforme demonstrado na imagem 8. A comissão e os coordenadores também pode incluir planos, porém é uma atividade atribuída ao professor, pois ele possui informações que devem ser preenchidas como justificativa, objetivo e lista de atividades.

Figura 8 – Criação de plano de monitoria

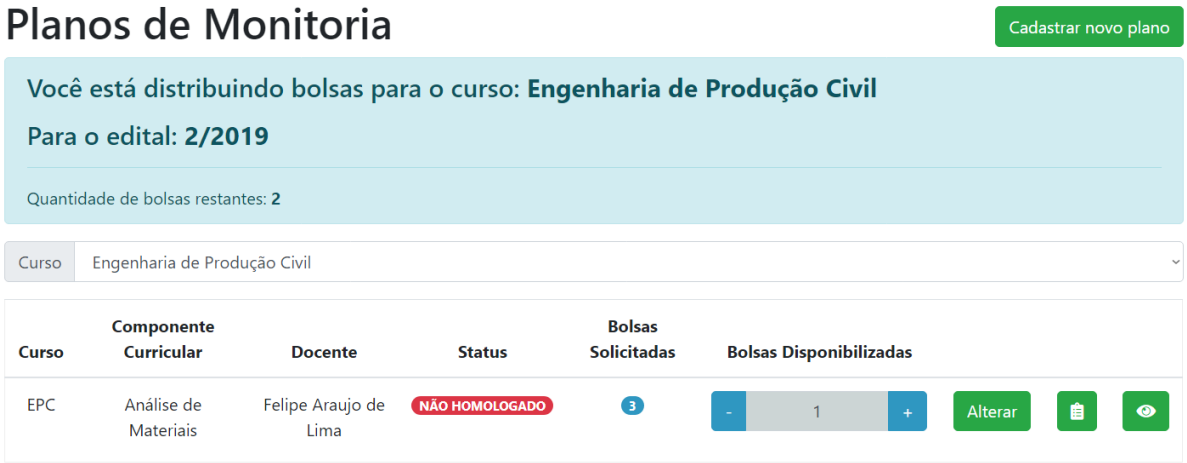


Fonte: Imagem elaborada pelos autores

Para o plano ficar disponível para candidatura dos estudantes é necessário que ele seja homologado pelo coordenador do curso ou comissão. O coordenador também pode editar o plano e pode alterar a quantidade de vagas de voluntários e bolsas disponibilizadas, por exemplo na imagem 9 foram solicitadas 3 bolsas, porém só foi disponibilizada 1 vaga para bolsista. Após homologar o plano, é possível indeferir para poder editar. A opção para homologar ou indeferir está disponível ao acessar a alteração do plano. Abaixo o teste de comportamento desta funcionalidade:

```
Funcionalidade:
    Distribuir bolsas para planos de monitoria e homologar os planos

Contexto:
    Dado que o usuario esta logado como coordenador ou comissao
    E esteja na pagina de gerencia de planos

Cenario: Distribuir um numero invalido de bolsas para um plano
    Quando tentar diminuir o numero de bolsas de um plano
    para um numero invalido
    Entao o sistema nao deve permitir a alteracao do plano

Cenario: Distribuir um numero valido de bolsas para um plano
    Quando existirem bolsas disponiveis o suficiente
    E tentar adicionar um numero valido de bolsas para um plano
    Entao o sistema deve realizar a distribuicao

Cenario: Homologar um plano
    Quando finalizar a distribuicao de bolsas
    E homologar o plano de monitoria
    Entao o sistema deve atualizar o status do plano para homologado
```

Figura 9 – Gerência de plano de monitoria

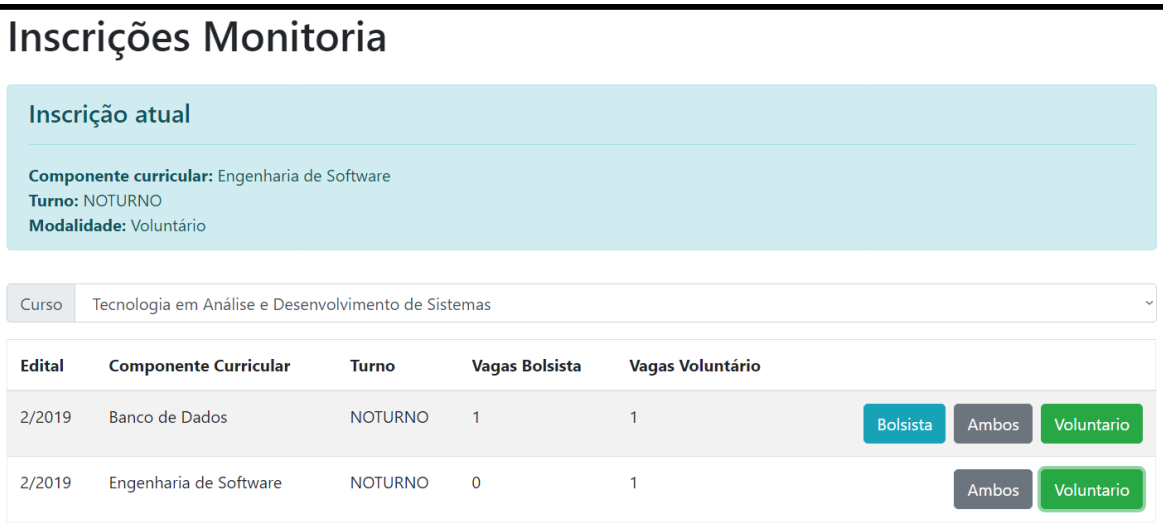


Fonte: Imagem elaborada pelos autores

4.3.6 Candidatura à Monitoria

No periodo determinado, os estudantes podem acessar a area de "Inscrição de Candidatos", selecionar um curso e escolher dentre uma lista de componentes, a modalidade da vaga desejada. É possivel escolher entre bolsista e voluntário. Ao escolher a modalidade como "Ambos", a preferência de classificação de acordo com as notas é para as vagas com bolsas. Caso não exista vaga para alguma modalidade em um Componente Curricular, não será exibido a opção para se candidatar, conforme imagem 12.

Figura 10 – Candidatura à Monitoria



Fonte: Imagem elaborada pelos autores

4.3.7 Inserção de Notas dos Candidatos

Durante o periodo destinado para a inserção das notas o professor deverá lançar as notas para que o sistema possa classificar os candidatos à monitoria. O acesso para lançamento das notas é pela tela Planos de Monitoria, o botão fica ao lado do componente. Será aprovado o estudante com nota de seleção igual ou superior ao estabelecido no edital como nota minima de seleção. Em caso de empate, será aprovado o estudante com maior nota no componente curricular em que pleiteia a monitoria e persistindo o empate ficará a decisão a cargo do docente-orientador do componente curricular que utilizará as setas para ordenar a classificação conforme na imagem 11. Os alunos são classificados de acordo com a escolha na hora da inscrição e na quantidade de vagas existentes para aquele componente. Abaixo, os testes de comportamento da funcionalidade:

```
Funcionalidade: Inserção de Notas
Esse cenario descreve a inserção das notas da selecao para a
monitoria
O professor acessa a listagem dos alunos inscritos no componente
curricular em que o mesmo eh o professor

Contexto:
    Dado que o usuario esta logado com perfil de professor
    E esteja na pagina de gerencia dos planos de monitoria
    E seleciona a opcao de inserir notas

Cenario: notas diferentes classificacao automatica
    Quando o professor inserir as notas de selecao e as medias e
    salvar
    Entao deve mostrar a classificacao atualizada de cada aluno

Cenario: aluno possui reprovacao
    Quando o professor indicar que aluno possui reprovacao e salvar
    Entao deve desclassificar o aluno
```

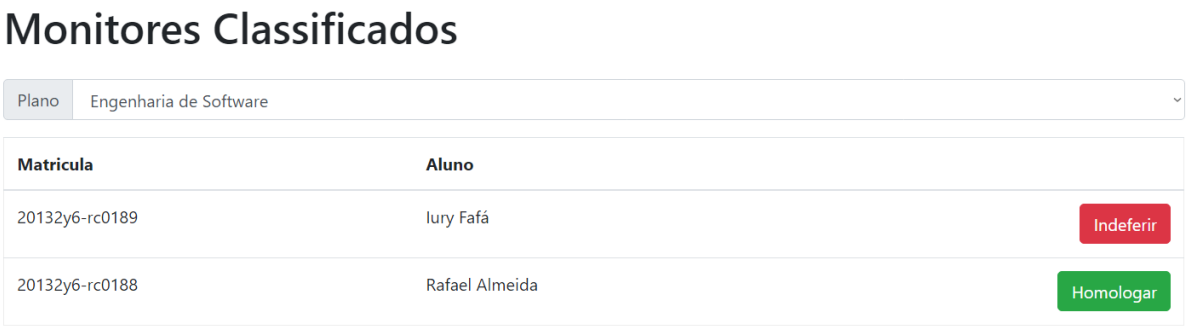
Figura 11 – Lançamento das notas

Fonte: Imagem elaborada pelos autores

4.3.8 Gerenciar Classificados

Através da Gerência de Classificados, a comissão pode homologar os candidatos que estão classificados e que entregaram toda a documentação exigida pelo edital. Após a homologação, o candidato é atualizado para "Selecionado"e está pronto para exercer as atividades de monitoria.

Figura 12 – Homologação dos Classificados



Fonte: Imagem elaborada pelos autores

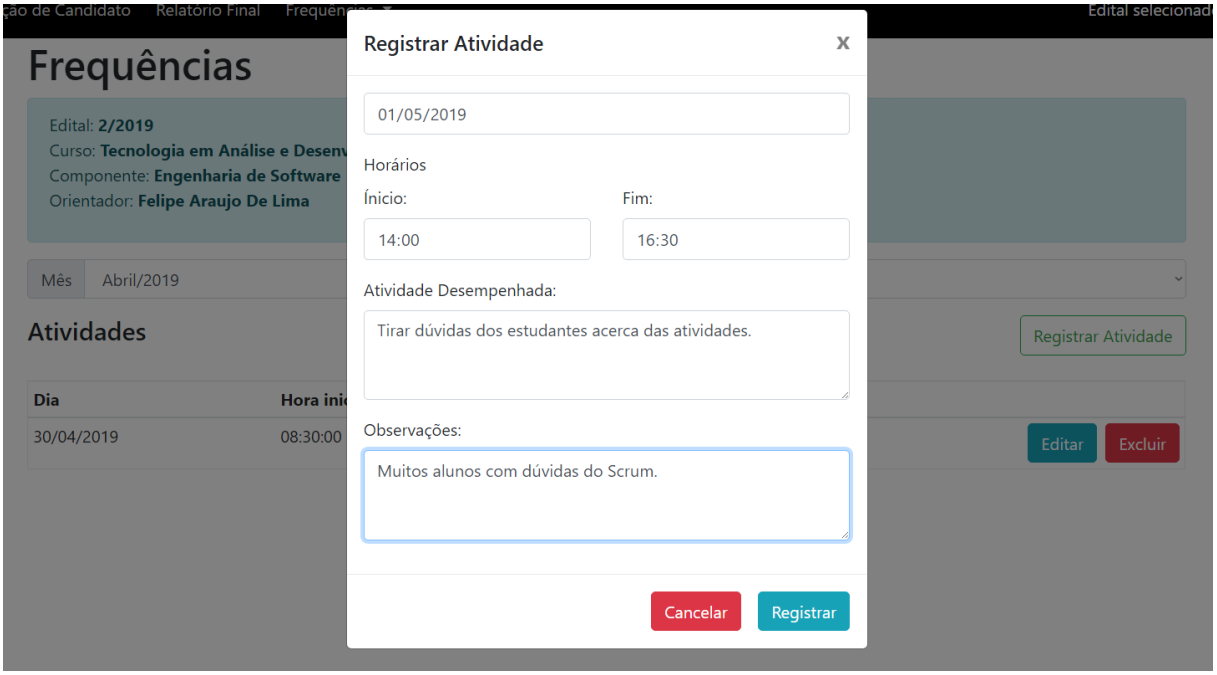
4.3.9 Submissão de ata mensal

Após selecionado, o estudante exercerá as atividades relacionadas a monitoria. O aluno deve submeter diariamente as atividades realizadas para gerar a ata mensal de monitoria. É possível selecionar a data, o periodo, a atividade desempenhada e qualquer observação que julgar interessante. Só é possivel adicionar atividades do dia corrente ou de dias anteriores, e apenas durante o periodo da monitoria. Abaixo, os testes de comportamento da funcionalidade:

```
Funcionalidade: Submeter uma atividade na ata mensal como um aluno
monitor

Contexto:
    Dado que o usuario esta logado como aluno monitor de um
    componente
    E esteja na pagina minha monitoria

Cenario: Registrar uma atividade valida
    Quando clicar em registrar atividade
    E preencher com as informacoes da atividade
    E clicar em registrar
    Entao o sistema deve registrar a atividade

Cenario: Registrar atividade em que a data eh fora do periodo de
monitoria
    Quando clicar em registrar atividade
    E preencher com uma data fora do periodo de monitoria
    E clicar em registrar
```

```
    Entao o sistema deve exibir a mensagem de data fora do periodo

Cenario: Registrar atividade em que a hora de inicio da atividade
eh antes do fim
    Quando clicar em registrar atividade
    E preencher com horas inconsistentes
    E clicar em registrar
    Entao o sistema deve exibir a mensagem de hora errada
```

Figura 13 – Adicionar atividades desempenhadas

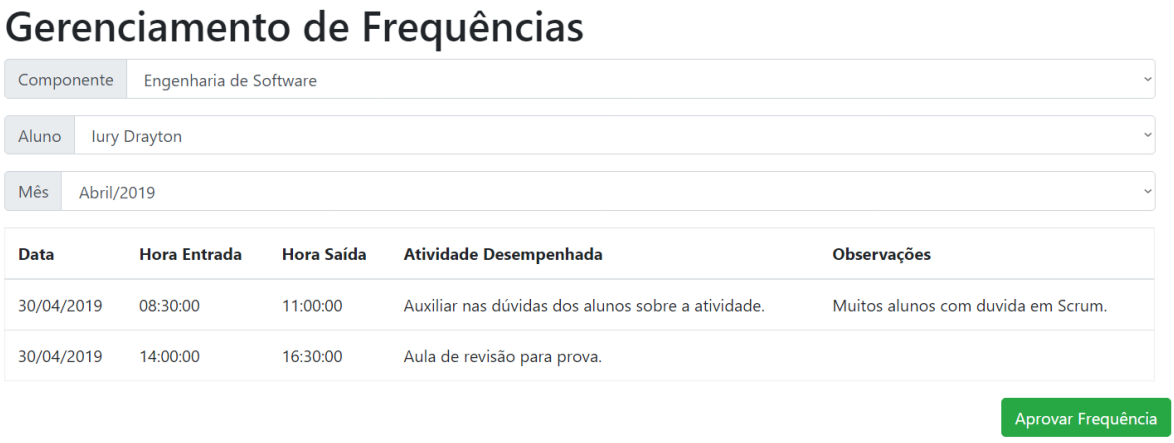


Fonte: Imagem elaborada pelos autores

4.3.10 Gerenciar Frequências

O professor-orientador pode acompanhar as atividades desempenhadas pelos monitores através da opção "Gerência de Frequências". Além de visualizar as atividades cadastradas, ao final de cada mês, ele deve aprovar ou não a frequência para que a comissão possa visualizar se o monitor exerceu as atividades naquele periodo. Após a aprovação, o aluno não pode mais editar as atividades daquele mês.

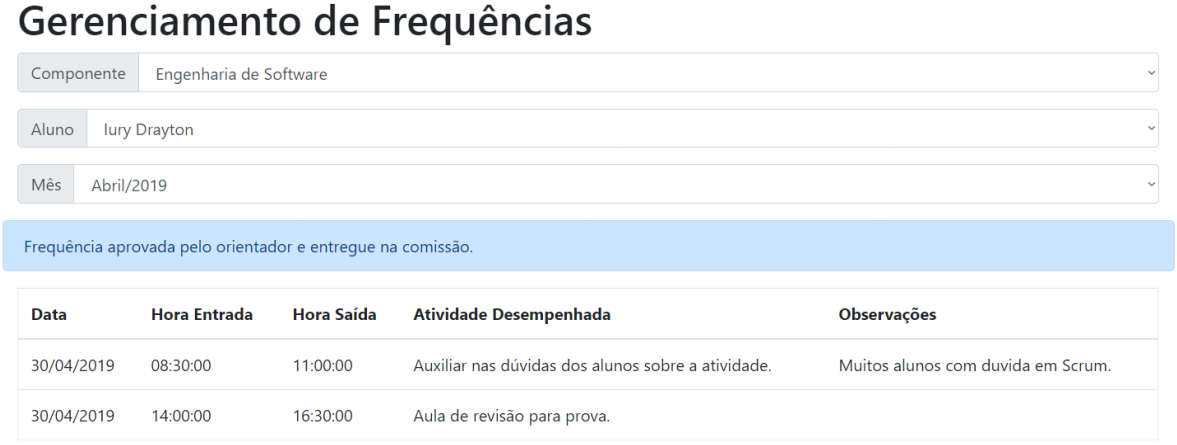
Figura 14 – Aprovação de frequência



Fonte: Imagem elaborada pelos autores

Após a aprovação do professor, a comissão pode conferir se está tudo certo, se clicar no botão "Receber Frequência". Caso seja necessário alguma correção, a comissão deve informar o professor para que possa "desaprovar"a frequência e assim liberar para o aluno realizar as correções necessárias.

Figura 15 – Frequência aprovada e recebida

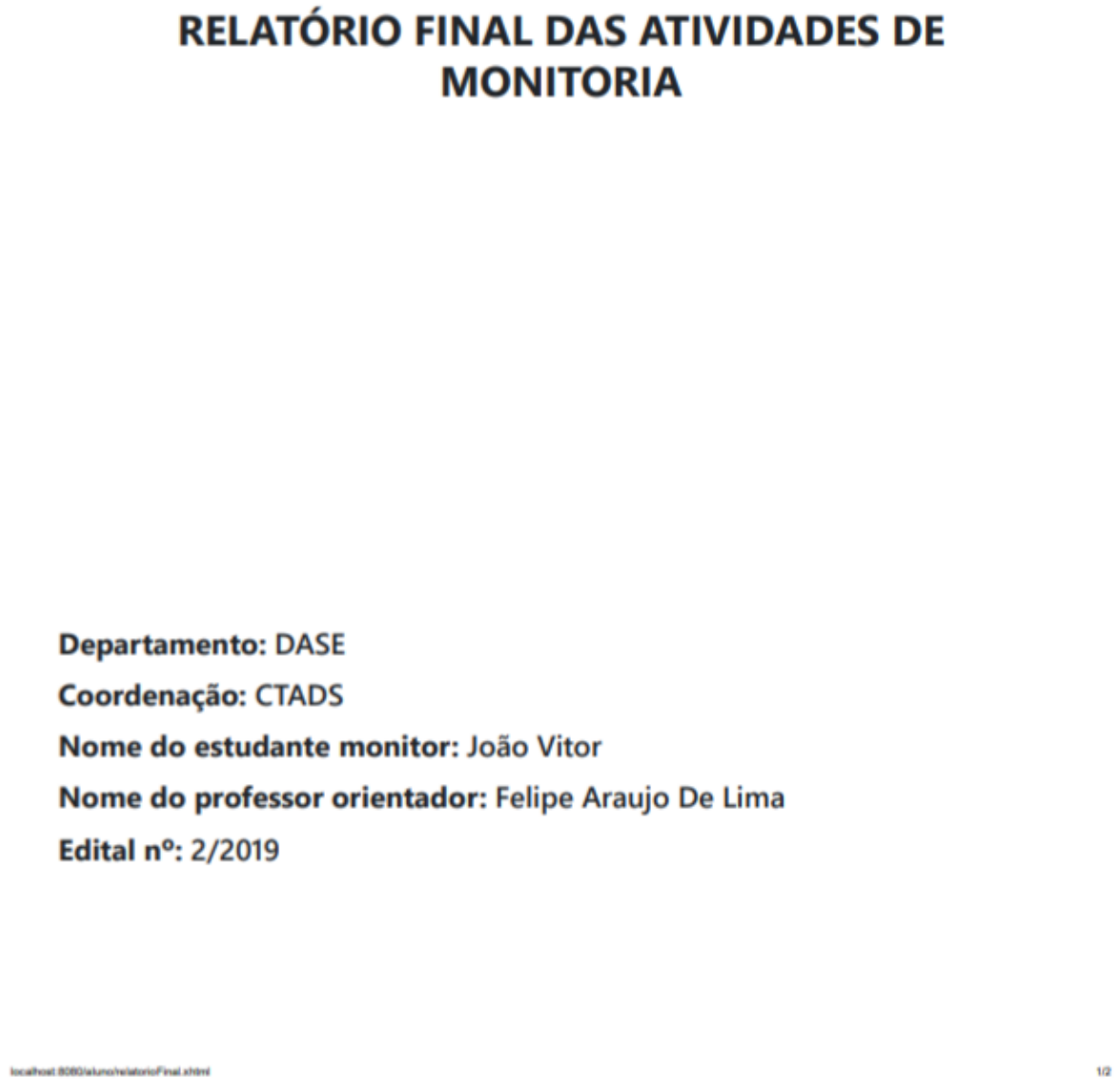


Fonte: Imagem elaborada pelos autores

4.3.11 Enviar relatório final

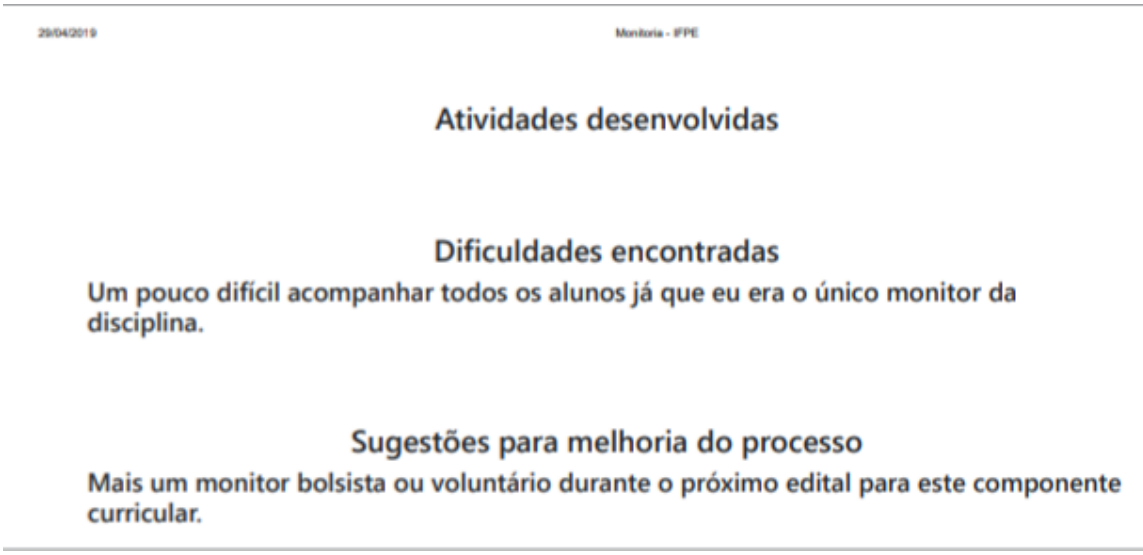
Na tela de relatório final, um aluno monitor descreve as atividades desenvolvidas, dificuldades encontradas, sugestões para melhoria do processo de monitoria além de avaliar o orientador com uma nota de 0 à 10. Esta tela foi feita para que o monitor pudesse ir incrementando seu relatório final ao longo da monitoria, então a cada vez que ele à acessa, a última versão salva do relatório é carregada e a edição por parte do monitor é livre, sempre persistindo suas alterações ao clicar no botão de salvar. Ao final do prazo da monitoria e quando o monitor precisar submeter a documentação para validação de sua monitoria, ele pode clicar no botão de impressão para gerar o seu relatório seguindo o modelo padrão do IFPE campus Recife.

Figura 16 – Exemplo de modelo de capa do relatório final formatado pelo sistema - Capa



Fonte: Imagem elaborada pelos autores

Figura 17 – Exemplo de modelo de conteúdo do relatório final formatado pelo sistema

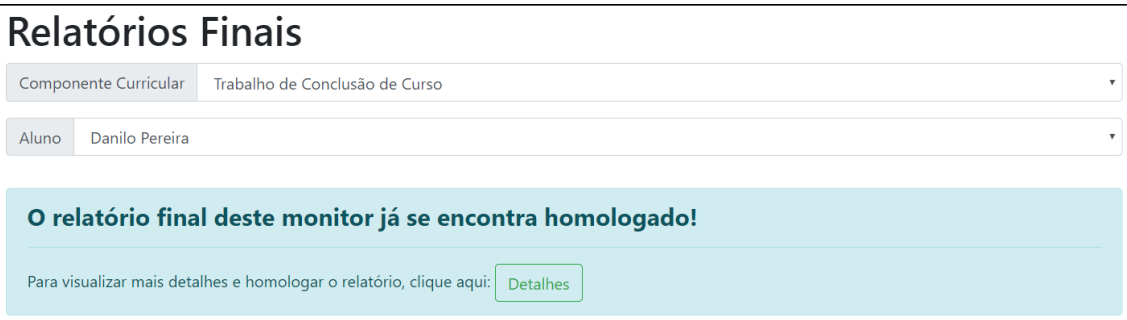


Fonte: Imagem elaborada pelos autores

4.3.12 Homologar relatório final

Na tela de gerência de relatórios, um professor pode visualizar detalhes de relatórios dos alunos de um determinado componente curricular limitando-se à apenas componentes onde ele é professor. Após revisar estes dados ao final do periodo de monitoria, o professor pode homologar aquele relatório. Podendo sempre consultar o relatório para ver os detalhes.

Figura 18 – Homologção de relatórios finais



Fonte: Imagem elaborada pelos autores

4.3.13 Visualizar relatórios finais recebidos

Na tela principal do sistema, professores e comissão podem visualizar a relação de alunos os quais tiveram seus relatórios finais recebidos pelos seus respectivos professores. Quando o usuário logado não faz parte da comissão, os componentes listados se limitam aos que o usuário é professor.

Figura 19 – Listagem de entrega de relatórios finais

Fonte: Imagem elaborada pelos autores

4.4 Testes automatizados

Para acompanhar a evolução do GEM foi desenvolvido uma série de testes unitários utilizando o framework jUnit e, com o intuito de documentar e validar suas funcionalidades, utilizamos o framework Cucumber e Selenium para criação de testes de comportamento.

4.5 Código

O código do GEM está disponível no link do repositório abaixo com licença de uso não comercial:

https://github.com/Masterfoni/ta-manager

5 Conclusão

Após o desenvolvimento do GEM, foi realizada uma implantação em um servidor do IFPE - Campus Recife, para uma simulação de uso em um edital. Foram convidados alguns professores e servidores que fazem parte da atual comissão de monitoria, além de atuais discentes, ex-discentes e até mesmo alunos de instituições de ensino superior diferentes. Foram simulados todos os fluxos disponíveis pelo GEM, desde a criação de editais até submissão de relatórios finais. Após a simulação, os participantes foram convidados a responder um questionário de acordo com seu perfil na aplicação.

Cem por cento dos alunos viram os fluxos de cadastro, candidatura e manutenção da monitoria como fáceis e intuitivos, podendo citar como possíveis melhorias: uma tela inicial condensando mais as principais funcionalidades disponíveis, além do fato de mostrar mensagens de confirmação ao final de algumas execuções.

A visão dos servidores que utilizaram o sistema também foi unanimamente positiva, com destaque a eficácia na criação e manutenção de um edital no sistema e a intuitividade dos menus e telas. Dentre as sugestões de melhoria por parte dos servidores, podemos destacar: um melhor redirecionamento quando uma sessão expirar e a possibilidade de suporte para pagamentos de bolsistas automaticamente.

Por fim, com base no que foi apresentado neste trabalho, podemos concluir que o GEM se mostrou uma eficiente ferramenta de suporte e gerenciamento para todo o ciclo de vida de um edital de monitoria, oferecendo visibilidade ao aluno e facilidade no gerenciamento e submissão de atas, visualização de resultados e inscrição. Além disso, é possível afirmar que os objetivos de compreender o funcionamento global de um edital de monitoria, aplicar os fundamentos do framework Scrum aliados à utilização de testes automatizados de comportamento das funcionalidades foram atingidos.

6 Trabalhos Futuros

- Integração com a gerência de pagamento de bolsas para monitores
- Envio de documentação diretamente pelo GEM
- Adição de relatório de alunos inscritos
- Integração com Q-Acadêmico

Referências

ANDERSON, David J. **Kanban: Mudança Evolucionária de Sucesso para seu Negócio de Tecnologia**. [S.l.: s.n.], 2011. 290 p. Citado 2 vezes nas páginas 27 e 29.

BEHAVIOUR-DRIVEN.ORG. **All You Need to Know About Behaviour-Driven Software**. 2016. Disponível em: <http://behaviour-driven.org/need-know-behaviour-driven-software.html>. Citado 2 vezes nas páginas 29 e 30.

CUCUMBER.IO. **Introduction : Cucumber**. 2018. Disponível em: <https://docs.cucumber.io/guides/overview/>. Citado 2 vezes nas páginas 30 e 31.

IFPE, (Campus Recife). EDITAL nº. 02/2018 - PROGRAMA INSTITUCIONAL DE MONITORIA DOS CURSOS SUPERIORES. 2017. Citado na página 21.

ROSE, Seb; WYNNE, Matt; HELLESØY, Aslak. **The cucumber for Java book: behaviour-driven development for testers and developers**. [S.l.: s.n.], 2015. 314 p. Citado na página 29.

SCHWABER, Ken; SUTHERLAND, Jeff. The Scrum Guide TM The Definitive Guide to Scrum: The Rules of the Game. 2016. Citado 2 vezes nas páginas 25 e 26.

SELENIUMHQ.ORG. **Selenium - Web Browser Automation**. 2018. Disponível em: <https://www.seleniumhq.org/>. Citado na página 31.