CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

SISTEMA PARA CONTROLE DO EVENTO SITES

GABRIEL LEANDRO JUNIOR DE SIQUEIRA

PEDRO VICTOR DE OLIVEIRA E SILVA

ANÁPOLIS - GO

2018

GABRIEL LEANDRO JÚNIOR SIQUEIRA

PEDRO VICTOR DE OLIVEIRA E SILVA

SISTEMA PARA CONTROLE DO EVENTO SITES

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientador: Prof. Me. Millys Fabrielle Araujo Carvalhaes

Anápolis-GO

GABRIEL LEANDRO JUNIOR SIQUEIRA
PEDRO VICTOR DE OLIVEIRA E SILVA

SISTEMA PARA CONTROLE DO EVENTO SITES

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em 12 de Dezembro de 2018, composta por:

Millys Fabrielle Araújo Carvalhaes
Presidente da Banca

Marcelo de Castro Cardoso
Prof(a). Convidado(a)

Natasha Sophie Pereira
Prof(a). Convidado(a)

RESUMO

O Seminário Interdisciplinar de Tecnologia e Sociedade (SITES) é um evento obrigatório para os discentes dos Bacharelados em Computação do Centro Universitário de Anápolis - UniEVANGÉLICA e aberto para a comunidade, e tem como propósito incentivar estudos que envolvem temas pertinentes ao desenvolvimento da Ciência e Tecnologia no contexto social contemporâneo. A ausência de automação nos processos de controle de frequência do SITES, aliada a dificuldade em obter o *feedback* dos participantes e a demora na geração de certificados, acaba sendo causa de aborrecimento e transtorno para os participantes do evento e funcionários dos cursos de computação. Devido a essa problemática, o presente trabalho tem como objetivo o desenvolvimento de um software para automatizar o gerenciamento do evento SITES e suas atividades. O sistema gerenciará a presença dos participantes de forma automatizada, será possível realizar a exportação dos relatórios, permitindo uma rápida consulta de dados e extração de métricas, além de substituir as atas de presença e questionários por formulários online, trazendo mais economia e sustentabilidade para a instituição, atendendo assim às expectativas da sociedade em termos de desenvolvimento tecnológico. Com a homologação do sistema pode-se comprovar na prática as vantagens propostas do sistema, bem como identificar novos pontos de melhorias.

Palavras-chave: Gerenciamento de Evento. Software Web.

ABSTRACT

The Interdisciplinary Seminar on Technology and Society (SITES) is a mandatory event for students of Computer Science of Centro Universitário de Anápolis - UniEVANGÉLICA and open to the community, and has an goal of the studies that involve the subjects to the development of Science and Technology. The absence of automation in the processes of frequency control of the SITES, ally with the receipt of feedback from the participants and the delay in issuance of the certificates, ends up being one of the causes of dissatisfaction and trouble for the participants of the event and the employees of the courses of computation. By the fact to this problem, the present work aims to develop software to automate and manage the SITES event and its activities. The system will manager the presence of the participants automatically, will be possible the reports export, allowing quick query of data and extraction of metrics, besides replacing the presence record and questionnaires by online forms, bringing more economy and sustainability to the institution, thus meeting the expectations of society in terms of technological development. With the homologation of the system it is possible to prove in practice the proposed advantages of the system, as well as to identify new points of improvement.

Key words: Event Management**.** Software Web.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BDD	*Behavior-Driven Development*
BSD	*Berkeley Software Distribution*
CSS	*Cascading Style Sheets*
DB	*Data Base*
DER	Diagrama Entidade Relacionamento
GCS	Gerência de Configuração de Software
HTML	*HyperText Markup Language*
JS	*Javascript*
MVC	*Model-View-Controler*
PGSQL	*PostgreSQL*
PHP	
PO	*Product Owner*
QR Code	*Quick Response Code*
SGBD	Sistema de Gerenciamento de Banco de Dados
SITES	Seminário Interdisciplinar de Tecnologia e Sociedade
SPA	*Single Page Applications*
SQL	*Structured Query Language*
TDD	*Test Driven Development*
TS	*Typescript*

SUMÁRIO

1 INTRODUÇÃO

O Centro Universitário de Anápolis – UniEVANGÉLICA por meio de seus cursos de graduação e outros departamentos, promove diversos eventos, tais como: seminários, palestras, semanas acadêmicas, acolhida de calouros. Dentre eles está o Seminário Interdisciplinar de Tecnologia e Sociedade (SITES). Este é um evento dos Bacharelados em Computação e tem como propósito incentivar estudos que envolvem temas pertinentes ao desenvolvimento da Ciência e a Tecnologia no contexto social contemporâneo.

O projeto SITES teve início no ano de 2008, com as professoras Adrielle Beze Peixoto e Inez Rodrigues Rosa que, na época, ministravam, respectivamente, as disciplinas de Sociedade da Informação e Leitura, Interpretação e Produção de Textos. Sob orientação destas, foram elaborados projetos que tinham como objetivo central a integração dos conhecimentos teórico-práticos e apresentados no formato de Seminário, denominado por SITES. Com o sucesso do projeto originaram-se as disciplinas de Projeto Interdisciplinar, que vão do I ao VII, cada uma responsável por um projeto específico.

O SITES torna-se o evento de apresentação dos trabalhos orientados nessas disciplinas, cada uma responsável pelo desenvolvimento de conhecimentos, habilidades e competências que são integradoras do perfil do egresso. A organização e exigências de cada projeto interdisciplinar são variáveis, mas o objetivo principal é a constituição de um acadêmico e profissional com formação integral, com capacidade para analisar contextos e propor soluções integradoras por meio de processos, técnicas e procedimentos inerentes à produção de software.

Em geral, a estrutura diária do SITES consiste em três partes. A abertura do evento, no qual alunos ou convidados podem fazer apresentações musicais e artísticas em um curto "momento cultural". A apresentação de trabalhos, em que são divulgados os resultados dos trabalhos desenvolvidos nas disciplinas de Projeto Interdisciplinar. E por fim, o encerramento do evento, quando os alunos devem preencher um formulário de avaliação das atividades do dia e assinar a ata de presença.

O SITES é o evento obrigatório para os discentes dos Bacharelados em Computação, ao mesmo tempo em que é aberto para a comunidade. Deste fato demanda grande esforço para o controle de frequência e extração do *feedback* dos participantes via formulário de avaliação, tendo em vista que são centenas de participantes envolvidos diariamente. Outra demanda que envolve um grande esforço manual é a geração dos certificados de participação os quais são emitidos para cada dia de evento.

De acordo com o documento de visão (APÊNDICE A) elaborado em conjunto com a *stakeholder* Adrielle Beze, a ausência de automação nos processos de controle de frequência do SITES acaba sendo causa de aborrecimento e transtorno para os participantes do evento e funcionários dos cursos de computação. O controle de frequência realizado por atas impressas, no qual cada discente deve procurar o professor responsável por sua disciplina naquele respectivo dia para assinar a ata de presença, traz como consequência filas de espera e confusão para encontrar o referido professor. Além disso, os questionários para obter o *feedback* dos participantes também contribuem com um ônus desnecessário para a instituição e o meio ambiente somado a isto, o uso de controle de processos manuais em um curso é de difícil compreensão e justificativa.

Finalizado o evento, a geração dos certificados de participação é outro procedimento extenso e realizado em tempo insatisfatório, pois depende da digitalização de atas para que os dados possam ser encaminhados ao setor institucional responsável. Outra etapa de difícil realização é a extração de métricas para a aplicação de melhorias no SITES. Informações como quantidades de participantes, por dia ou gênero, por exemplo, atualmente, são métricas inviáveis de serem produzidas devido a quantidade de dados gerados.

Devido a problemática apresentada, este trabalho tem como objetivo o desenvolvimento de um software para automatizar o gerenciamento do evento SITES e suas atividades, proporcionando ganho de tempo, qualidade, relatórios contínuos e em tempo real, maior interação com os participantes e geração de certificados com maior agilidade, atendendo assim às expectativas da sociedade em termos de desenvolvimento tecnológico.

O sistema gerenciará a presença dos participantes de forma automatizada, utilizando, como exemplo, o QR Code (*Quick Response Code*, código de resposta rápida). Será possível realizar a exportação dos relatórios para o Microsoft Office Excel - um editor de planilhas produzido pela Microsoft – permitindo uma rápida consulta de dados e extração de métricas, além de substituir as atas de presença e questionários por relatórios online, trazendo mais economia e sustentabilidade para a instituição. A seguir serão apresentadas as atividades desenvolvidas para a criação do sistema.

No capítulo 2 são apresentados os principais conceitos e temas pertinentes ao trabalho. Em seguida, no capitulo 3 é exposto o desenvolvimento do sistema, evidenciando o padrão utilizado para a realização das tarefas. Por fim, o capitulo 4 descreve os principais aprendizados e considerações relativas ao trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada a teoria utilizada para o desenvolvimento do presente trabalho, bem como os métodos e ferramentas para a aplicação da engenharia de software, o gerenciamento de projetos e tecnologias para o desenvolvimento.

2.1 Engenharia de software

Segundo Pressman e Maxim (2016, p. 14), “a engenharia de software abrange um processo, um conjunto de métodos (práticas) e um leque de ferramentas que possibilitam aos profissionais desenvolverem software de altíssima qualidade”. Os autores dizem ainda que a engenharia de software é direcionada por um conjunto de princípios que ajudam na produção de softwares eficazes. Sommerville (2011, p. 3) salienta-se que “quando falamos de engenharia de software, não se trata apenas do programa em si, mas de toda a documentação associada e dados de configurações necessários para fazer esse programa operar corretamente”.

Fornecer valor aos usuários e simplificar são alguns dos princípios que abrangem a engenharia de software (PRESSMAN; MAXIM, 2016). Assim, para este trabalho, a engenharia de software fornecerá processo, métodos, técnicas e ferramentas que contemplarão desde as fases iniciais do projeto, englobando levantamento e análise de requisitos, até as fases finais que se encerra com a utilização do software pelo cliente.

2.1.1 Modelos de Desenvolvimento

O processo de desenvolvimento de software é um conjunto de atividades a serem seguidas para o desenvolvimento de um software. Segundo Sommerville (2011, p. 18) um processo de software é definido como um “conjunto de atividades relacionadas que levam à produção de um produto de software”. Esse processo é importante “porque propicia estabilidade, controle e organização para uma atividade que pode, sem controle, tornar-se bastante caótica” (PRESSMAN; MAXIM, 2016, p. 52).

Existem diversos modelos de processos, dentre eles os modelos em cascata, incremental, evolucionário. Neste trabalho será utilizada uma metodologia de desenvolvimento ágil pertencente a categoria de processos iterativos incrementais, a qual permite entregar valor ao cliente de forma rápida, a fim de minimizar erros, sendo adaptativa a mudanças. Segundo

Pressman e Maxim (2016) os modelos incrementais oferecem grande vantagem em relação aos lineares, como cascata, permitindo a entrega de funcionalidades ao cliente em menor tempo.

Com a aplicação do modelo incremental e da metodologia ágil, se faz possível, a partir dos primeiros incrementos, entregar valor ao cliente. Dessa forma, é possível coletar *feedback* dos usuários, aplicando as possíveis melhorias nos próximos incrementos.

2.1.2 Levantamento de requisitos

De acordo com Sommerville (2011, p. 57), "os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que ele oferece e as restrições sobre seu funcionamento". Esses requisitos são classificados em requisitos funcionais e requisitos não funcionais. Sommerville (2011) descreve os requisitos funcionais como funções que o sistema deve fornecer e como deve se comportar em determinadas situações; e os requisitos não funcionais como requisitos que não estão relacionados às funções específicas do sistema, mas sim a qualidade do mesmo.

Pressman e Maxim (2016) classificam a engenharia de requisitos em três etapas: concepção, levantamento e elaboração.

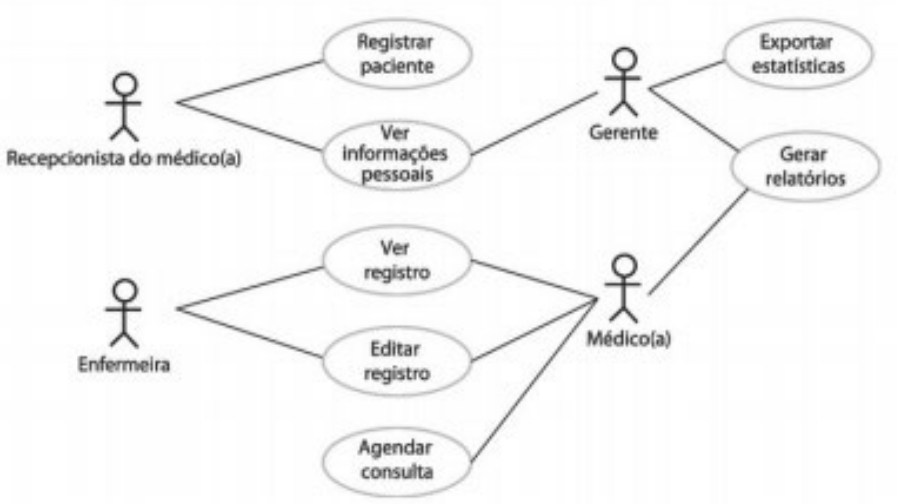
> A engenharia de requisitos começa com a concepção (uma tarefa que define a abrangência e a natureza do problema a ser resolvido). Ela prossegue para o levantamento (uma tarefa de investigação que ajuda os envolvidos a definir o que é necessário) e, então, para a elaboração (na qual os requisitos básicos são refinados e modificados). (PRESSMAN; MAXIM, 2016, p. 131);

Identificar quais os requisitos de um sistema é uma das partes mais difíceis de todo o processo de software e a que mais influência no sucesso ou fracasso do software final. Afinal, "projetar e construir um programa de computador elegante que resolva o problema errado não atende as necessidades de ninguém. " (PRESSMAN; MAXIM, 2016, p. 131).

Entre as diversas técnicas de levantamento de requisitos existentes destacam-se a entrevista informal e o *brainstorming*. Sommerville (2011, p. 72) afirma que "entrevistas são boas para obter uma compreensão global sobre o que os *stakeholders* fazem, como eles podem interagir com o novo sistema e as dificuldades que eles enfrentam com os sistemas atuais". Segundo Souza (2012) *brainstorming* significa "tempestade de ideias" e é uma técnica muito utilizada para incentivar a criatividade dos envolvidos. No *brainstorming* ideias são encorajadas, mesmo que a princípio não pareçam tão usuais, isso leva muitas vezes a soluções criativas.

Outra técnica utilizada para levantar requisitos é o Diagrama de caso de Uso (figura 1). O diagrama auxilia os analistas a terem uma visão macro do sistema, ajudando a identificar os atores (usuários) e suas responsabilidades (ações). Pressman e Maxim (2016) explica que o Diagrama de Caso de Uso é a visão geral das funcionalidades do sistema. O Diagrama de Caso de Uso auxilia na modelagem do sistema, podendo ser validado com o cliente a fim de confirmar que todas as funcionalidades foram mapeadas assim como seus atores. Pressman e Maxim (2016, p. 606) enfatiza que "um caso de uso descreve como um usuário interage com o sistema, definindo os passos necessários para atingir um objetivo específico" dessa forma facilitando o entendimento do domínio do problema do sistema, fornecendo um norte inicial para especificar os requisitos. Pressman e Maxim (2016) ainda explana que o diagrama de caso de uso "[…] é uma visão geral de todos os casos de uso e de como eles estão relacionados".

Figura 1 – Exemplo de Diagrama de Caso de Uso



Fonte: Sommerville (2011, p. 75)

Nestes diagramas os usuários são representados pelos atores e os casos de uso por elipses. Os atores são conectados por linhas com os casos de uso que eles executam (Pressman e Maxim, 2016). Como foi mencionado um caso de uso pode se relacionar com outro por inclusão ou extensão e os atores podem se generalizar (Pressman e Maxim, 2016).

Com isso a engenharia de requisitos vem para ajudar a garantir que os requisitos sejam identificados de forma clara, concisa, consistente e sem ambiguidade. Proporcionando assim, um software que atenda às reais necessidades do cliente.

2.2 Metodologias Ágeis

Existem diversas metodologias ágeis existentes, cada qual com objetivos e características divergentes. No entanto, elas possuem algumas familiaridades: documentar apenas o necessário, entregas incrementais, respostas rápidas a mudanças, participação ativa do cliente em todo processo, foco no cliente e na entrega de valor contínua. Pressman e Maxim (2016, p. 63) explicam que os princípios da metodologia ágil "[...] priorizam a entrega mais do que a análise e o projeto (embora essas atividades não sejam desencorajadas); também priorizam a comunicação ativa e continua entre desenvolvedores e clientes. "

Uma metodologia ágil deve seguir os princípios e valores do "Manifesto para o Desenvolvimento Ágil de Software", que é um documento assinado por 16 (dezesseis) renomados desenvolvedores com o objetivo de ajudar outros desenvolvedores de software com as melhores formas para se criar um software (BECK et al., 2001).

No processo convencional dirigido a planos, a entrega do software final costuma ser muito demorada. Especificar completamente os requisitos com uma documentação extensa e detalhada é uma das características deste processo das quais fazem com que ele seja custoso e prolongado. Sommerville (2011) destaca que em alguns sistemas críticos de controle de segurança que exigem maior atenção, uma abordagem orientada a planos pode ser a melhor opção. Porém, em softwares que estão em um ambiente de rápidas mudanças, isso pode trazer problemas. O autor frisa que neste caso, quando o software estiver pronto, pode ser que o motivo original da sua existência não exista mais ou tenha sido alterado.

Na metodologia ágil o foco é a entrega rápida de softwares úteis. Segundo Cohn (2004, p. 258) "o objetivo no desenvolvimento ágil é encontrar o equilíbrio certo entre documentação e discussão". A documentação não é totalmente descartada, mas o foco é voltado para o desenvolvimento do software. "O software não é desenvolvido como uma única unidade, mas como uma série de incrementos – cada incremento inclui uma nova funcionalidade do sistema" (SOMMERVILLE, 2011, p. 39).

2.2.1 Gerenciamento de projeto com Scrum

Schwaber e Sutherland (2017), desenvolvedores do *Scrum*, definem o *Scrum* como sendo um *framework* para gestão da resolução de problemas complexos e adaptativos, em que se visa agregar o mais alto valor possível ao produto. Sabbagh (2014) diz que entre os benefícios

do *Scrum* estão: entregas frequentes de retorno ao investimento dos clientes; redução dos riscos do projeto; maior qualidade no produto gerado; mudanças são bem-vindas; visibilidade do progresso do projeto; redução do desperdício e aumento da produtividade.

O *Scrum* é apoiado em três pilares: transparência, inspeção e adaptação. No qual segundo Schwaber e Sutherland (2017), a transparência requer que todos tenham um mesmo entendimento do que está sendo visto, a inspeção roga pela detecção de variações indesejadas nos artefatos e a adaptação permite fazer ajustes para minimizar desvios não previstos.

O Time Scrum consiste em um *Product Owner*, o Time de Desenvolvimento e um *Scrum Master* (SCHWABER; SUTHERLAND, 2017). De forma objetiva temos o *Product Owner*, também conhecido como P. O., como o responsável por gerenciar o *backlog* do Produto, o time de desenvolvimento composto pela equipe multidisciplinar encarregada pelo desenvolvimento do produto e o *Scrum Master* responsável pelo processo e as boas práticas do Scrum.

O *Scrum* é composto por vários pequenos ciclos de atividades denominados *Sprint*. "O coração do *Scrum* é a *Sprint*, um *time-boxed* de um mês ou menos, durante o qual um 'Pronto', incremento de produto potencialmente liberável é criado" (SCHWABER; SUTHERLAND, 2017, p.9).

Segundo Schwaber e Sutherland (2017) dentre os conceitos de inspeção e adaptação, temos quatro eventos formais no *Scrum*. O "planejamento da *Sprint"*, no qual é planejado o trabalho a ser realizado na iteração. A "reunião diária", que tem duração de 15 minutos e tem como objetivo planejar o trabalho das próximas 24 horas e inspecionar o trabalho realizado desde a última reunião passada. A "revisão da *sprint*", uma reunião informal realizada no final da *sprint* para verificar o que foi realizado e adaptar o *product backlog se* necessário. E por fim, a "retrospectiva da *sprint*", que é quando o Time tem a oportunidade de analisar os erros e acertos buscando melhorias no processo para a próxima *sprint*.

No decorrer de uma *Sprint*, pode acontecer uma revisão do *backlog*, denominada "reunião de *Backlog Grooming*". Bernardo (2015) diz que "o propósito das reuniões de *Backlog Grooming* (Refinamento do *Backlog*) é aprimorar o *Product Backlog*". O autor diz ainda que o tempo de uma reunião de *Backlog Grooming* é 5% a 10% da iteração.

O *Product Backlog*, de acordo com Schwaber e Sutherland (2017, p. 14) “é uma lista ordenada de tudo que é conhecido ser necessário no produto”. Ou seja, todos os requisitos a serem desenvolvidos para aquele produto.

2.2.2 BDD

O BDD (*Behivor Driven Development*) é uma metodologia ágil de desenvolvimento de software baseada no comportamento da aplicação, originalmente criada por Dan North, por volta dos anos 2000. Na metodologia contém as melhores práticas para desenvolver softwares entregando o máximo de valor no menor prazo com maior qualidade (SMART, 2014). Provendo uma linguagem ubíqua entre clientes e desenvolvedores de software, servindo assim como documentação para os requisitos do sistema de forma ativa e menos ambígua (NORTH, 2006;SMART, 2014).

O BDD é considerado por muitos uma evolução do TDD (*Test Driven Development*), no qual, gera especificações executáveis que podem ser utilizadas como testes de aceitação automatizados. As especificações do BDD são descritas por Histórias de Usuário e Cenários de Aceitação que descrevem o comportamento do sistema do ponto de vista do usuário. As especificações são escritas livres de detalhes de implementação e com linguagem orientada ao domínio do negócio (SOLÍS; WANG, 2011).

Algumas das vantagens do BDD conforme citado por Ruggiero (2017) são: priorização dos recursos críticos; linguagem comum, de fácil entendimento; suporte às necessidades atuais e futuras do projeto e código de qualidade.

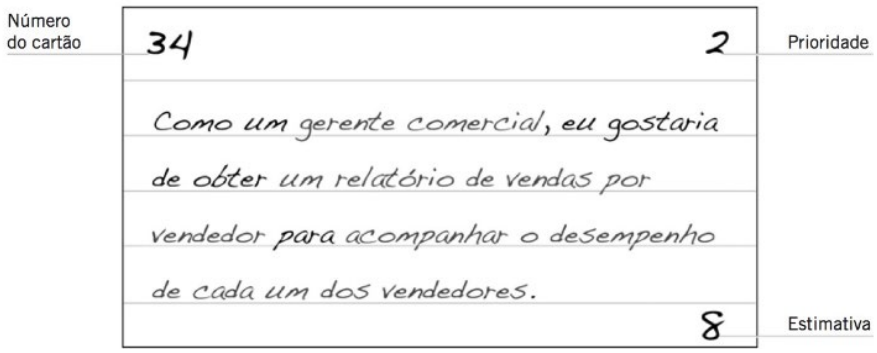
2.2.3 Histórias de usuário e cenários de aceitação

A aplicação do BDD é iniciada com o desenvolvimento das Histórias de Usurário e a descrição dos Cenários de Aceitação (o comportamento da aplicação diante de eventos). História de Usuário ou *User Story*, no qual, Wildt et al. (2015) afirmam que elas descrevem os requisitos do sistema de uma forma ágil e Pressman e Maxim (2016) completam que elas descrevem o resultado, características e funcionalidades no qual devem ser implementadas. As histórias de usuário são escritas sem detalhes de implementação, porém, devem conseguir transmitir a necessidade do negócio (PRIKLADNICKI et al; 2014).

Uma boa história de usuário deve ser independente, negociável, possuir valor, ser estimável, pequena e testável (COHN, 2004). Cohn (2004) explica ainda que a dependência entre as histórias de usuário pode causar problemas de priorização e estimativa, por isso devem ser independentes; devem ser negociáveis, pois, não são imutáveis e devem ser formuladas em consenso entre os clientes e a equipe de desenvolvimento; devem possuir valor, pois, em muitos projetos existem funcionalidades que não possuem utilidade para o negócio; devem ser estimáveis, pois, os desenvolvedores irão transformá-las em código executável; devem ser pequenas, visando que várias histórias pequenas irão compor histórias grandes (EPIC's); e por fim, testáveis, para que quando os testes forem executados, os mesmos possam demonstrar que a funcionalidade foi implementada com sucesso.

Conforme Prikladnicki et al. (2014, p. 44) normalmente é utilizado um formato para descrever histórias de usuário: "Como um **<tipo_de_usuário>**, eu gostaria de **<funcionalidade>** para **<benefício>**". A figura 2 representa um exemplo de uma história de usuário no formato citado.

Figura 2 - Exemplo de uma História de Usuário



Fonte: Prikladnicki et al (2014, p. 44)

Seguindo esse formato, a funcionalidade que o usuário deseja é escrita em linguagem natural, de fácil entendimento por ambas as partes interessadas. Além disso, podendo verificar seu valor para o negócio, sendo utilizada como documentação para os requisitos do sistema.

Com a história de usuário escrita no padrão citado, o próximo passo para a aplicação do BDD é a criação dos cenários de aceitação. Os critérios de aceitação são representados por um evento que ocorre no sistema dado uma condição que então representa um comportamento esperado. Na criação de cenários de aceitação é utilizada linguagem ubíqua com base em um formato de palavras-chave: dado que, quando e então (WILDT, 2014).

2.3 Gerência de configuração

Sistemas de software estão em constante evolução, a tecnologia é algo que está sempre a inovar. Acompanhar e manter-se atualizado se torna uma atividade muito complexa para ser realizada de forma manual, podendo, muitas vezes, se tornar um gargalo no desenvolvimento de software. A Primeira Lei da Engenharia de Sistemas diz que: “Não importa onde você esteja no ciclo de vida do sistema, o sistema mudará, e o desejo de alterá-lo persistirá por todo o ciclo de vida” (PRESSMAN; MAXIM, 2016, p. 624).

Realizar manutenção em sistemas de software é algo pressuposto dentro das metodologias de desenvolvimento ágil. Essas metodologias pregam que o software deve ser desenvolvido em incrementos, no qual, a cada ciclo de desenvolvimento novas funcionalidades serão adicionadas ao produto de software. Dessa forma, várias modificações serão realizadas ao longo do ciclo de vida de desenvolvimento de um produto de software, sendo necessário controlar tais modificações. Nesse contexto surge a Gerência de Configuração de Software (GCS) que, conforme Pressman e Maxim (2016, p. 623) explicam:

> [..] é um conjunto de atividades destinadas a gerenciar as alterações, identificando os artefatos que precisam ser alterados, estabelecendo relações entre eles, definindo mecanismos para gerenciar diferentes versões desses artefatos, controlando as alterações impostas e auditando e relatando as alterações feitas.

Pode-se definir a GCS como sendo um processo, no qual, todos os artefatos desenvolvidos no decorrer do projeto e suas relações, são armazenados, recuperados, modificados e identificados de maneira única (HUMBLE; FARLEY, 2014). Por fim, conforme explica Pádua Filho (2008), sem a gestão de configuração é impossível atingir um nível razoável de qualidade, podendo ocasionar diversos problemas no desenvolvimento do produto de software.

2.3.1 Controle de versão (Git)

Controle de Versão é uma atividade imprescindível no desenvolvimento de qualquer software e está contido dentro da gerência de configuração de software. “O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas” (GIT,2018). Conforme explica Humble e Farley (2014, p. 32), o controle de versão é um sistema que guarda várias versões de um ou vários arquivos, de modo que, quando algum deles é modificado ainda é possível acessar versões anteriores do mesmo.

Conforme dito por Humble e Farley (2014) qualquer artefato relacionado ao sistema pode e deve estar sob o controle de versão. Artefatos como o próprio código fonte, scripts de bancos de dados, binários, documentação, entre outros.

Atualmente existem diversos softwares *open source* que funcionam de forma local, centralizada ou distribuída, dentre eles, destaca-se o software Git.

Git é um sistema de controle de versão *open source* criado por Linus Torvalds, no qual, funciona de forma distribuída. Teve sua concepção no ano de 2005 e é um sistema fácil de se utilizar, leve e rápido. Seu controle de versão é feito por meio de *snapshots* (captura de algo em determinado momento) (GIT, 2018).

2.4 Projeto e desenvolvimento web

Para desenvolver um sistema diversas ferramentas e tecnologias são necessárias, ferramentas como de banco de dados, modelagem de requisitos, modelagem de processos, testes, entre outras. Também se faz necessário a utilização de linguagens de programação e *frameworks* que suportam as linguagens. Esses *frameworks* têm como objetivo minimizar esforços por parte do desenvolvedor durante a construção de algum sistema. A seguir são apresentadas algumas linguagens e *frameworks* para aplicações Web.

2.4.1 Javascript com Angular

Javascript (JS) é uma linguagem de *script* que permite implementar componentes complexos em páginas Web (MOZILLA, 2018). O interpretador do Javascript é vinculado com os navegadores Web, e executada do lado do cliente. Atualmente é a linguagem principal para desenvolvimento do lado cliente, mas também pode ser utilizada em outros ambientes fora do navegador. Uma linguagem multi-paradigma (orientado a objetos, funcional e imperativo) dinamicamente tipada baseada em protótipo (MOZILLA, 2018).

O JS pode ser incorporado a documentos HTML (*HyperText Markup Language*) ou Linguagem de Marcação de Hipertexto que é utilizada para desenvolver páginas web estáticas juntamente com o CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) que é responsável pela aparência da página HTML, seu estilo. O JS manipula o HTML e o CSS tornando as páginas que antes eram estáticas, passando a ser mais interativas e dinâmicas.

Angular é um *famework* JavaScript de código aberto, mantido pelo Google, que auxilia no desenvolvimento e execução de SPA (*Single Page Applications* ou Aplicações de uma Página). O desenvolvimento de aplicações Angular (na sua versão 6) é escrito não em Javascript, mas em Typescript.

Typescript (TS) é um super conjunto de JS que compila (reescreve) para Javascript simples. O TS fornece um conjunto de classes, *interfaces* e tipagem estática opcional, recursos esses não disponíveis no JS.

2.4.2 PHP com Laravel

Existem diversas linguagens de programação que dão suporte ao ambiente WEB como, por exemplo, JAVA, PHP e ASP. Dentre elas destaca-se o PHP que é uma linguagem de *script* de código aberto, no geral, utilizada para o desenvolvimento *web e t*eve sua concepção por volta dos anos 90.

PHP (*Hypertext Preprocessor*) é uma linguagem simples e que possui muitos recursos, tendo foco maior para *scripts* do lado servidor, que atendem as requisições de clientes, provendo dados e conteúdos solicitados. De acordo com a documentação do PHP (2018), ele pode ser utilizado na maioria dos sistemas operacionais (Linux, Windows, Mac OS X) e também suporta a maioria das aplicações de servidor web atuais (Apache, Nginx, Lighttpd). Além disso, também é possível desenvolver em dois paradigmas de programação: estrutural e orientado a objetos. Outro recurso disponível no PHP é sua ampla variedade de suporte a bancos de dados.

Laravel é um *framework* de código aberto de desenvolvimento web escrito em PHP, criado por Taylor B. Otwell, que tem se tornado cada vez mais popular no desenvolvimento web devido a sua ampla variedade de recursos. O Laravel é um *framework* que utiliza a arquitetura MVC (Model-View-Controler), com uma ampla documentação atualizada. O *framework* é de sintaxe simples e totalmente modular. O Laravel utiliza o Composer para gerenciar suas dependências. O Composer é uma ferramenta de gerenciamento de dependências para o PHP que permite declarar e instalar/atualizar as dependências de um projeto PHP.

2.5 Banco de dados

Um banco de dados, de acordo com Heuser (2009), é uma coleção de dados integradas que objetiva atender uma comunidade de usuários. A maioria dos bancos de dados possui seu próprio sistema de gerenciamento de banco de dados (SGBD). “Um sistema de gerenciamento de banco de dados (SGBD) é um software que incorpora as funções de recuperação e alteração de dados em um banco de dados” (HEUSER, 2009, p. 23).

Existem diversos *softwares* para gerenciar banco de dados (DB), neste trabalho aborda-se apenas o tipo relacional, que de acordo com Heuser (2009), é o que domina o mercado atual. Em um DB relacional, os dados estão distribuídos na forma de tabelas (HEUSER, 2009). Existem outros tipos de DB como, por exemplo, não relacionais (*NoSQL*), orientado a objetos.

Conforme explica Heuser (2009) um modelo de dados é a descrição dos tipos de informações que estão armazenadas no DB, sendo assim, a descrição formal de sua estrutura. Para se modelar DB utiliza-se de linguagens de modelagem de banco de dados que conforme Heuser (2009) podem ser classificadas em textuais e gráficas. Essa descrição formal de sua estrutura pode ser denominada de esquema de banco de dados.

Em geral, no desenvolvimento de software é comum projetar-se o DB antes do desenvolvimento. Heuser (2009) afirma que um projeto de banco de dados pode ser descrito em dois níveis de abstração o modelo conceitual e o modelo físico.

O modelo conceitual é a descrição do banco de dados de forma abstrata, independente do SGBD (HEUSER, 2009). O autor também explica que esse modelo conceitual é normalmente representado em linguagem gráfica por meio de diagrama que é denominado de diagrama-entidade-relacionamento (DER).

O modelo lógico é a descrição do banco de dados mais detalhada dependente do SGBD. “Um modelo lógico de um DB relacional deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas” (HUESER, 2009, p. 26).

No geral DB relacionais fornecem uma linguagem de definição de dados para realizar a especificação de dados e uma linguagem de manipulação de dados para realizar consultas e atualização de dados (KORTH et al, 2012). A linguagem SQL (*Structured Query Language*) é amplamente utilizada e suporta as linguagens de definição de dados e de manipulação (KORTH et al, 2012).

2.5.1 Postgresql

Conforme mencionado anteriormente o PHP suporta diversos bancos de dados, dentre eles o PostgreSQL, um banco de dados relacional de código aberto, gratuito e de fácil utilização. O PostgreSQL (PGSQL) pode ser executado em diversos sistemas operacionais e dispões de inúmeros recursos. Sendo compatível com a linguagem SQL oferecendo recursos complexos de manipulação de dados. O PGSQL suporta diversas linguagens de programação incluindo o PHP, sua licença BSD (*Berkeley Software Distribution*) dá direito a qualquer um para usar o SGBD de forma gratuita.

3 DESENVOLVIMENTO

Esse capítulo apresentará os passos realizados para o desenvolvimento do software utilizando o modelo interativo incremental integrado com o framework Scrum. Será apresentado o processo de desenvolvimento que inclui o processo de controle de versão do código, as arquiteturas desenvolvidas e os padrões de implementação dos requisitos.

3.1 Processo Metodológico

Inicialmente foi realizada a análise de requisitos, juntamente com os *stakeholders*, conforme Pressman e Maxim (2016, p.167) afirmam que ela "resulta na especificação das características operacionais do software, indica a interface do software com outros elementos do sistema e estabelece restrições a que o software deve atender". Para realizar essa atividade foi necessário identificar com os *stakeholders*, por meio de entrevistas informais e *brainstorming,* as funcionalidades (*features*) e os cenários do sistema. Essas funcionalidades e cenários foram mapeadas utilizando modelo baseado em cenários no formato de História de Usuário. Wildt et al. (2015) relata que as histórias de usuário descrevem os requisitos do sistema de uma forma ágil e de acordo com Pressman e Maxim (2016) representam o resultado, características e funcionalidades no qual devem ser implementadas.

Nas entrevistas realizadas com os *stakeholders* foram identificadas no Documento de Visão (APÊNDICE A) as histórias de usuário que posteriormente foram refinadas e adicionadas no *backlog* do produto. Pressman e Maxim (2016) definem o *backlog* como sendo uma lista com prioridades das funcionalidades do projeto que geram valor ao cliente.

Para realizar o gerenciamento do projeto foi empregado o *framework* ágil Scrum que de acordo com Prikladnicki et al. (2014, p. 22) "auxilia no gerenciamento de projetos complexos e no desenvolvimento de produtos". Para a auxiliar na sua aplicação foi utilizada a ferramenta Taiga (ferramenta *open source* para auxílio no gerenciamento de projetos) que contém os recursos para criação do *Backlog*, *Sprints*, *Kanban*, entre outros conceitos do *framework* Scrum.

Após a realização da análise de requisitos foi feita a priorização das histórias de usuário que estão contidas no *backlog* do produto. A priorização foi realizada utilizando a técnica "MoSCoW" conforme descrita no APÊNDICE A. Após a priorização das histórias foi executado o planejamento das *sprints* do projeto que consistem em "ciclos completos de desenvolvimento de duração fixa que, ao final, resultem em incrementos potencialmente entregáveis do produto" (PRIKLADNICKI et al., 2014, p. 31).

No desenvolvimento do sistema foi utilizada a metodologia *Behavior-Driven Development* (BDD), ou ainda Desenvolvimento Guiado por Comportamento que Humble e Farley (2014, p. 195) entendem como sendo os critérios de aceitação que devem ser escritos como sendo as expectativas do cliente em relação ao comportamento da aplicação e esses critérios devem ser utilizados para realizar testes de aplicação (aceitação) para verificar se o sistema atende aos seus requisitos. O sistema foi implementado utilizando os *frameworks* Laravel v5.6 e Angular v6, e os testes foram realizados por meio das ferramentas disponibilizadas pelos *frameworks*.

Após o desenvolvimento, a homologação do sistema foi realizada no XIX SITES. Para realizar a homologação do sistema será necessário preparar um servidor para a aplicação, realizar carga de dados, na qual, serão cadastradas as programações do evento, usuários entre outras informações. Por fim, após a homologação, os clientes assinarão o Termo de Aceite da Entrega (APÊNDICE N) para que seja concluído o projeto.

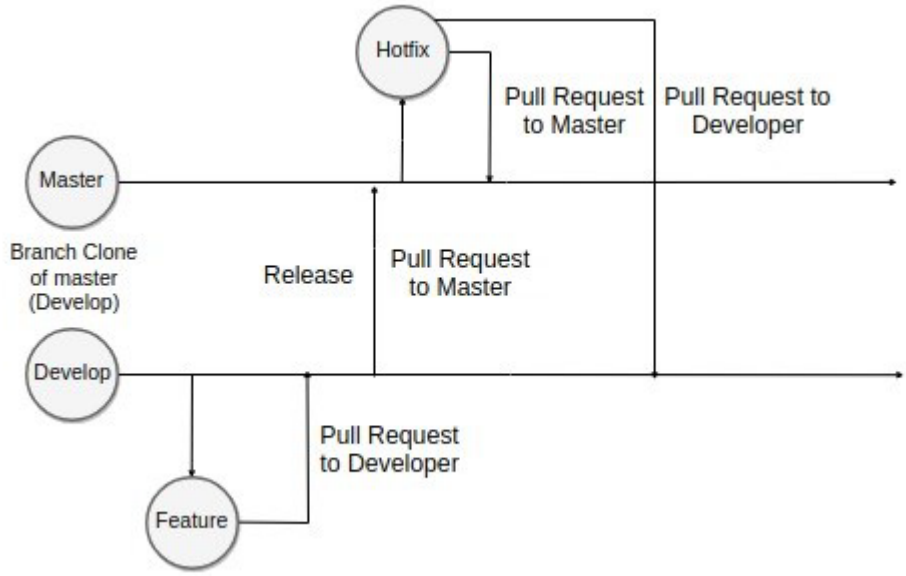
3.1.1 Interativo Incremental

A aplicação do modelo incremental auxiliará na entrega contínua do software desenvolvido, entregando valor para os *stakeholders*, possibilitando a coleta de feedbacks dos mesmos, levando para o backlog as possíveis melhorias e validando os requisitos.

3.1.2 Política de Versionamento

Antes de iniciar o desenvolvimento foi decidido pela equipe que todo e qualquer artefato do sistema seria colocado sobre o controle de versão utilizando o sistema Git. A escolha do software Git deu-se pelo fato de que ele é gratuito e de fácil utilização no controle de versão. Após definir o software de controle de versão a equipe de desenvolvimento, com intuito de padronizar o gerenciamento de versão do código, elaborou a política de versionamento descrita na figura 3. Política esta que norteou o processo de desenvolvimento evitando problemas como: conflitos de código, versões duplicadas e/ou desatualizadas e a liberação do código. A seguir explana-se cada elemento da política de versionamento elaborada pela equipe de desenvolvimento.

Figura 3 - Política de Versionamento



Fonte: Os Autores

Inicialmente decidiu-se que a política de versionamento conteria duas *branchs*, além da *branch* auxiliar (*Feature*). A *branch* principal, denominada *Master* na qual conterá todo o código estável do sistema, e uma *branch* secundária denominada *Develop*, que será responsável por armazenar o código em desenvolvimento. Ambas têm duração até o final do projeto.

Na seleção de tarefas, foi definido que ao selecionar uma tarefa para ser implementada, o responsável deve criar uma *branch* auxiliar (*feature*) com base no código da *Develop*. Após o desenvolvimento da *feature* o responsável deve realizar uma junção (*merge*) para a *branch Develop* e a remoção da *branch* auxiliar (feature). Posteriormente, pode ser feita a liberação do código para a *branch Master* por meio de merge.

Durante a utilização do sistema, caso seja encontrado algum erro ou falha no mesmo, deve-se criar uma *branch* denominada "*Hotfix*", com base no código da *Master*. Após a correção são feitos dois merges, um para *Master* e outro para *Develop*, a fim de manter ambas as versões atualizadas no que se diz respeito a correção da falha.

Desta forma foi possível padronizar o versionamento do sistema, separando o código em desenvolvimento do código estável, consequentemente proporcionando mais segurança durante o processo e evitando envios de funcionalidades não finalizadas para o código liberado na versão estável do sistema.

3.1.3 Scrum

Com a aplicação do Scrum foi possível planejar as *sprints* selecionando o backlog de acordo com a prioridade dos requisitos, proporcionando organização durante o decorrer do projeto e o controle sobre as atividades a serem desenvolvidas. Por meio das revisões das *sprints* foi possível obter a transparência e feedbacks durante o decorrer do projeto, evitando que os stakeholders se percam durante o andamento do mesmo.

3.1.3.1 Planejamento da *Sprint*

Foi definido pela equipe de desenvolvimento que todas as histórias de usuário seriam fragmentadas em três tarefas visando uma fácil organização. A primeira tarefa chamada de back-end, trata-se do desenvolvimento de artefatos relacionados aos serviços (API) da aplicação, como o banco de dados. A segunda tarefa, denominada front-end, refere-se as atividades de desenvolvimento das telas da aplicação e integração com a API desenvolvida na tarefa anterior. Por fim, a terceira tarefa, "atualizar documentação", para, se necessário, realizar alguma alteração na documentação objetivando a conformidade entre as histórias de usuário e os requisitos implementados.

3.1.4 *Sprint* 01 – Levantamento de Requisitos e BDD

A primeira *Sprint* do projeto teve por objetivo a criação dos artefatos do projeto (documento de visão, histórias de usuário, diagrama de caso de uso), os esforços foram concentrados na modelagem do sistema, visto que ela irá nortear todo o desenvolvimento do mesmo.

A princípio foram realizadas as primeiras entrevistas com os *stakeholders* nas quais foram expostos seus interesses por meio da técnica de levantamento de requisitos *brainstorming*. Como resultado foi construído o Documento de Visão (APÊNDICE A), no qual está mapeado o problema que o projeto irá resolver, uma visão macro dos seus requisitos e a estrutura do projeto.

Com base no Documento de Visão, foi modelado o Diagrama de Caso de Uso (APÊNDICE B), que demonstra os atores que interagem com o sistema e suas

responsabilidades, dessa forma foi possível validar os requisitos com os *stakeholders* quanto ao seu entendimento.

Com base no Diagrama de Caso de Uso (APÊNDICE B) foi possível iniciar a escrita das Histórias de Usuário nas quais foram mapeadas as primeiras funcionalidades, conforme a metodologia do BDD (*Behavior Driven Development*) sugere, juntamente com os cenários de aceitação iniciais.

Foram levantadas as Histórias de usuário:

1. US001 – Autenticar Usuário
2. US002 – Registrar Usuário
3. US003 – Programação
4. US004 – Manter Programação
5. US005 – Manter Formulário de Avaliação
6. US006 – Autorizar Check-in
7. US007 – Realizar Sorteio
8. US008 – Editar Usuário
9. US009 – Notificar Programação
10. US010 – Visualizar Participantes

Essas histórias de usuário (APÊNDICE C até L) nortearam o desenvolvimento do sistema. Todas as histórias abordadas foram adicionadas ao *backlog* do projeto na ferramenta Taiga (APÊNDICE M) que foi priorizado utilizando a técnica "MoSCoW".

3.1.5 *Sprint* 02 – Preparação do Ambiente e Desenvolvimento US001 e US002

A segunda *Sprint* teve por objetivo a preparação do ambiente de desenvolvimento e também a implementação das histórias US001 – Autenticar Usuário e US002 – Registrar Usuário. A seguir explana-se como foi realizada a preparação do ambiente de desenvolvimento.

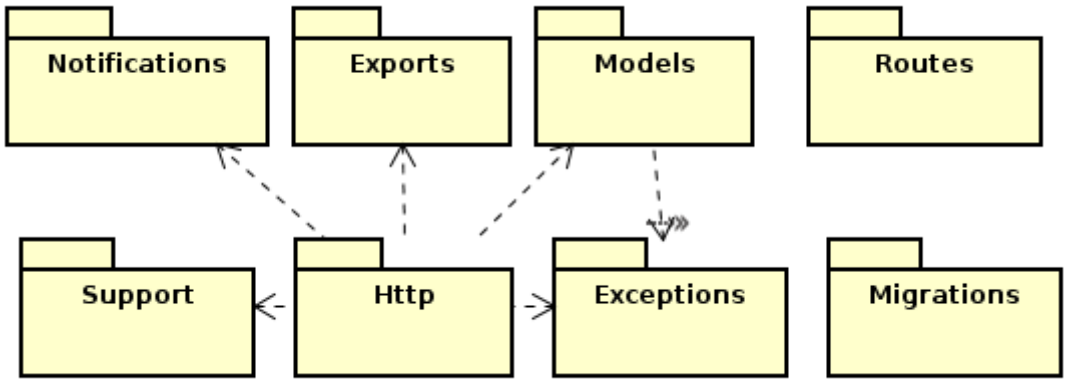
A preparação do ambiente de desenvolvimento se deu por meio da instalação de todas as dependências necessárias para a implementação do sistema, tais como: Composer, PHP, Node, NPM, Angular CLI, PostgreSQL, entre outras. Com o ambiente devidamente configurado, foi realizada a instalação padrão dos *frameworks* Laravel e Angular, responsáveis, respectivamente, pelo back-end e front-end, e a configuração da conexão com o banco de dados

PostgreSQL, possibilitando assim realizar a configuração da arquitetura de software para ambos os *framework's*. A seguir explana-se os componentes principais das arquiteturas desenvolvidas.

3.1.5.1 Arquitetura Laravel

O *framework* Laravel traz consigo, por padrão, diversas funcionalidades que aceleram o processo de desenvolvimento, como, por exemplo, envio de e-mail, criptografia, notificações, entre outras. Assim, o foco fica apenas no domínio do sistema, tendo em vista que as funcionalidades básicas já estão presentes no próprio *framework*. A arquitetura utilizada no projeto (figura 4) foi uma customização da arquitetura padrão do Laravel pela equipe de desenvolvimento, objetivando a organização da estrutura de pastas e melhor separação de interesses, melhorando assim a manutenibilidade e legibilidade do software.

Figura 4 - Diagrama de Arquitetura Back-end



Fonte: Os Autores

O pacote denominado Migrações (*Migrations)* contém todas as migrações do sistema, que são classes responsáveis pela criação das tabelas e campos no banco de dados. Por padrão, o Laravel utiliza as migrações para controlar e versionar o banco de dados disponibilizando comandos para criar, gerar, atualizar ou deletar as tabelas do banco de dados.

O pacote Modelos (Models) compreende as classes responsáveis por mapear as tabelas do banco de dados em objetos do sistema, também chamados de Object Relational Mapper (ORM). Esses objetos são responsáveis por consultar, escrever ou atualizar informações no banco de dados, neles também estão inclusas a maioria das regras de negócio do sistema.

O pacote HTTP engloba os controladores, requisições, interceptadores (*middlewares*), e recursos (*resources*), objetos responsáveis por converter os dados do pacote modelos para os

dados exibidos na tela da aplicação. Os controladores são responsáveis por receber as requisições dos clientes e realizar a validação dos dados executando ações nos modelos e retornando dados para a aplicação front-end.

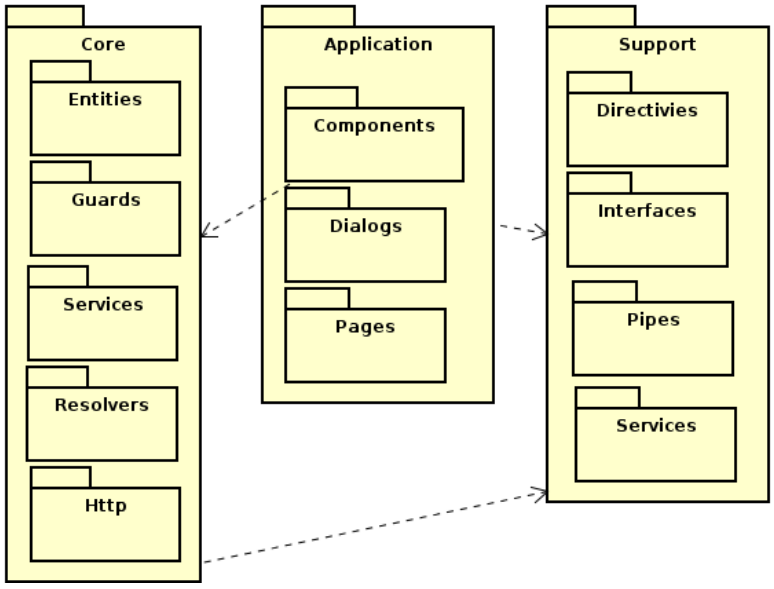
O pacote Suporte (*Support*) compreende classes genéricas que podem ser úteis em todo o sistema. Classes de validação, exceções, recursos, dentre outras. Já o pacote Notificações (*Notifications*) são objetos que notificam diretamente o usuário, como, por exemplo, e-mails e alertas.

O pacote Exceções (Exceptions), como o próprio nome sugere, armazena as exceções do sistema, como mensagens de erro que poderão vir a ser apresentadas. Já o pacote Exportações (Exports) são classes responsáveis por realizar a exportação de dados dos modelos para planilhas do Excel. Por fim o pacote Rotas (Routes) contém a definição das rotas do sistema, os pontos de entrada para os controladores do sistema.

3.1.5.2 Arquitetura Angular

Após realizar a instalação padrão do Angular, foi definida a arquitetura e a estrutura de pasta a ser seguida durante o desenvolvimento, objetivando a melhor separação de interesses, melhora na manutenibilidade e qualidade de software. A figura 5 demonstra todos os pacotes contidos na arquitetura definida pela equipe de desenvolvimento.

Figura 5 - Diagrama de Arquitetura Front-end



Fonte: Os Autores

Foram definidas três camadas para englobar as classes e objetos do sistema. A camada Suporte (*Support*), uma camada de infraestrutura, que contém as subcamadas *Directivies*, *Interfaces*, *Pipes* e *Services* e tem como responsabilidade servir as demais com seus respectivos conteúdos. A camada Aplicação (*Application*), que compreende todas as páginas, estilos, caixas de diálogos e componentes de tela do sistema. E por fim, a camada Núcleo (*Core*), que abrange as classes e serviços de integração entre o front-end e o back-end, e contém entidades de mapeamento de dados.

A seguir explana-se os padrões de desenvolvimento back-end e front-end aplicados nas arquiteturas apresentadas.

3.1.5.3 Padrão de desenvolvimento

Após as arquiteturas devidamente configuradas se torna possível realizar a implementação da primeira funcionalidade do sistema. Conforme elucidado anteriormente as histórias foram divididas em três tarefas (back-end, front-end, documentação). A seguir é explanado o processo de desenvolvimento padrão do back-end tendo como referência a história de usuário US002 – Registrar Usuário na qual será utilizado como modelo padrão de desenvolvimento para os demais requisitos.

3.1.5.3.1 Padrão de desenvolvimento Back-end

Ao iniciar o desenvolvimento da funcionalidade, primeiramente é criado a migração de banco de dados. A figura 6 retrata a migração da funcionalidade registrar usuário (US002).

Figura 6- Migração da Tabela de Usuários

```php

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('cellphone')->nullable();
            $table->string('password')->nullable();
            $table->date('birthday')->nullable();
            $table->boolean('verified')->default(false);
            $table->enum('type', [
                \UniEventos\Models\User::TYPE_STUDENT,
                \UniEventos\Models\User::TYPE_SERVANT,
                \UniEventos\Models\User::TYPE_COMMUNITY
            ])->nullable();
            $table->enum('gender', [
                \UniEventos\Models\User::GENDER_MALE,
                \UniEventos\Models\User::GENDER_FEMALE
            ])->nullable();
            $table->string('registration')->nullable();
            $table->string('role')->nullable();
            $table->rememberToken();
            $table->timestamps();
            $table->softDeletes();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Fonte: Os Autores

Na migração estão mapeados o nome da tabela (*users*), tal como seus campos e respectivos tipos de dados a serem armazenados. Uma das vantagens da utilização das migrações do Laravel é o controle do versionamento do banco de dados realizado pelo próprio *framework*, bem como a conversão para a linguagem do banco de dados. Em seguida é realizada a criação do objeto modelo que estruturará essa tabela no banco de dados e armazenará também,

as regras de negócio referentes ao usuário dentro do sistema. A figura 7 demonstra o modelo da tabela de usuários.

Figura 7 - Modelos de Usuários

Fonte: Os Autores

Consequentemente é criada a classe de teste do modelo para verificar se o mapeamento está funcionando corretamente, realizar testes na criação, atualização, exclusão e consulta no banco de dados, a fim de evitar erros durante a execução do software. Também são executados testes nas demais funcionalidades referentes as regras de negócio específicas do modelo de usuário. A figura 8 representa a classe de teste para o modelo de usuário.

Figura 8 - Teste do Modelo de Usuário

```php
class UserTest extends TestCase
{
    use RefreshDatabase;

    /**
     * A basic test example.
     *
     * @return void
     */
    public function testCreateUser()
    {
        $user = factory(User::class)->make();
        $this->assertTrue($user->save());
        $this->assertNotEmpty($user->id);
    }

    /**
     * A basic test example.
     *
     * @return void
     */
    public function testUpdateUser()
    {
        $user = factory(User::class)->create();
        $this->assertTrue($user->update([
            'name' => 'Test Updated'
        ]));
    }

    /**
     * A basic test example.
     *
     * @return void
     */
    public function testFindUser()
    {
        $user = factory(User::class)->create();
        $found = User::query()->find($user->id);
        $this->assertNotEmpty($found);
        $this->assertEquals($user->id, $found->id);
    }

    /**
     * A basic test example.
     *
     * @return void
     */
    public function testDeleteUser()
    {
        $user = factory(User::class)->create();
        $this->assertTrue($user->delete());
        $this->assertEmpty(User::query()->find($user->id));
    }
```

Fonte: Os Autores

Realizados os devidos testes então são gerados os controladores responsáveis por processar as requisições da funcionalidade de registro de usuário. A figura 9 demonstra o controlador responsável por registrar o usuário no sistema. Adiante é realizada a configuração das rotas da API (figura 10).

Figura 9 - Controlador de Registo de Usuário

```php
class RegisterController extends Controller
{
    /**
     * Handle a registration request for the application.
     *
     * @param  RegisterUserRequest $request
     * @return \Illuminate\Http\Response|mixed
     */
    public function register(RegisterUserRequest $request)
    {
        event(new Registered($user = $this->create($request->validated())));
        return new RegisteredUserResource($user);
    }

    /**
     * @return array
     */
    public function verifyUniqueEmail()
    {
        return [
            'available' => User::isEmailAvailable(
                request('email'),
                request('ignoreId', null)
            )
        ];
    }

    /**
     * @return array
     */
    public function verifyUniqueCellphone()
    {
        return [
            'available' => User::isCellphoneAvailable(
                preg_replace('/[^0-9]/', '', request('cellphone')),
                request('ignoreId', null)
            )
        ];
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array $data
     * @return \UniEventos\Models\User|mixed
     */
    protected function create(array $data)
    {
        $user = User::query()->create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),
            'gender' => $data['gender'],
            'type' => $data['type'],
            'birthday' => Carbon::createFromFormat('d/m/Y', $data['birthday']),
```

Fonte: Os Autores

Figura 10 - Rotas do Registo de Usuário

```php
/**
 * Public Api
 */
Route::group(['prefix' => '/auth'], function () {
    /**
     * Register User End Point's
     */
    Route::post('available/email', 'Auth\RegisterController@verifyUniqueEmail');
    Route::post('available/cellphone', 'Auth\RegisterController@verifyUniqueCellphone');
    Route::post('register', 'Auth\RegisterController@register');
```

Fonte: Os Autores

Por fim, é realizado o teste de funcionalidade que verifica a funcionalidade por completo, testando todas as rotas criadas e suas validações e respostas com o intuito de garantir

que o requisito tenha sido implementado corretamente, evitando assim futuras falhas e refatorações no código. A figura 11 demonstra o teste de funcionalidade para a funcionalidade de registro de usuário.

Figura 11 - Teste da Funcionalidade de Registro de Usuário

```php
class RegisterUserApiTest extends TestCase
{
    use RefreshDatabase;

    /**
     * @test
     * @return void
     */
    public function it_should_create_a_user()
    {
        $response = $this->post('/api/auth/register', [
            'name' => 'Test User',
            'gender' => 'M',
            'email' => 'test@email.com.br',
            'password' => '123321',
            'password_confirmation' => '123321',
            'type' => 0,
            'registration' => '1234567',
            'birthday' => '29/01/2000',
            'cellphone' => '(62) 99999-9999'
        ]);
        $response->assertStatus(201);
        $response->assertJsonStructure([
            'success',
            'data' => [
                'message'
            ]
        ]);
    }

    /**
     * @test
     * @return void
     */
    public function it_should_not_create_a_user()
    {
        $response = $this->post('/api/auth/register', []);
        $response->assertStatus(422);
        $response->assertJsonStructure([
            'data',
            'error',
            'message'
        ]);
    }
```

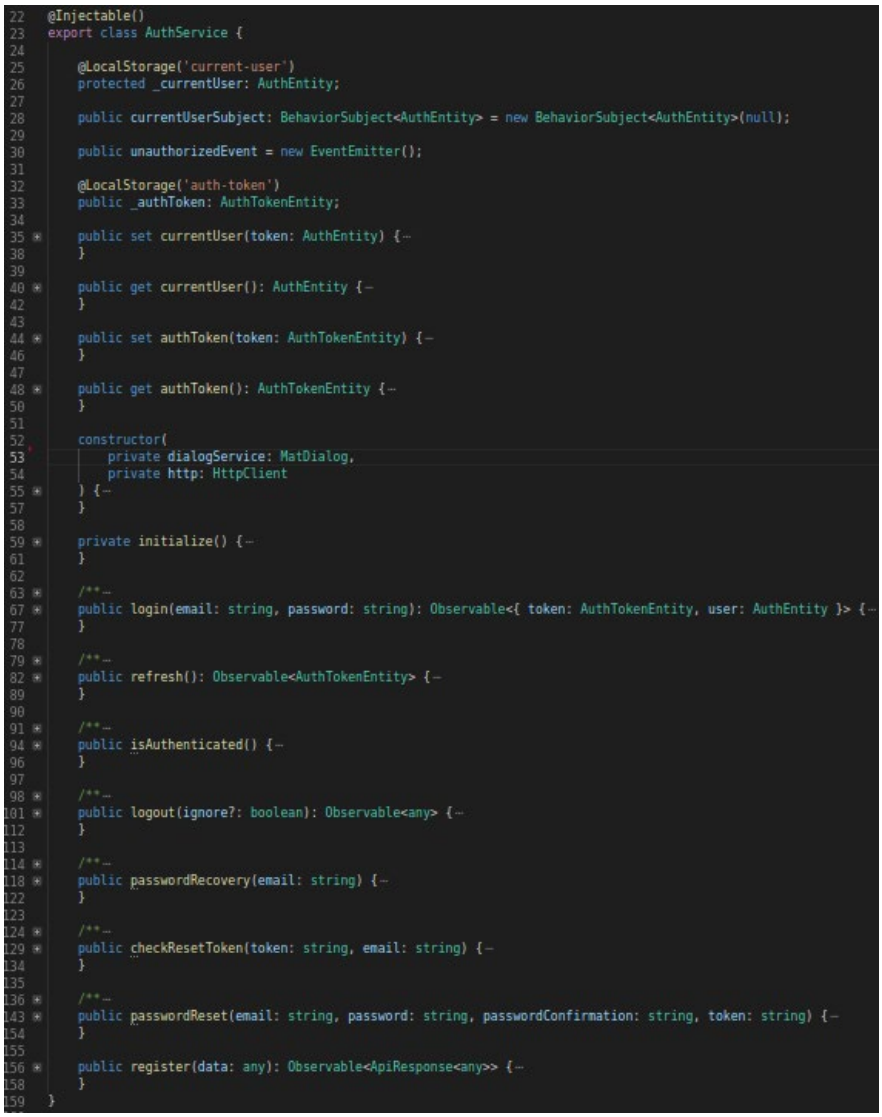
Fonte: Os Autores

Deste modo encerra-se o desenvolvimento da tarefa do back-end, podendo então, dar início ao desenvolvimento das telas (front-end) bem como a integração com o back-end. A seguir explana-se as etapas necessárias para a realização da tarefa do front-end, que também será utilizada como referência para o desenvolvimento dos demais requisitos.

3.1.5.3.2 Padrão de desenvolvimento Front-end

No desenvolvimento da tarefa front-end, a princípio é criado o serviço no qual a funcionalidade irá consumir os dados do back-end, onde é feita a integração. A figura 12 demonstra o serviço que engloba a funcionalidade de registro de usuário no sistema. Após realizar a criação do serviço ele é devidamente registrado e exportado no módulo para assim poder ser utilizado por outros elementos do sistema.

Figura 12 - Serviço Responsável por Registrar o Usuário



Fonte: Os Autores

Realizada a criação do serviço então é produzida a página da funcionalidade, página essa onde o usuário irá interagir através de formulários, botões entre outros elementos visuais. Essa página é descrita na forma de um componente dentro do Angular conforme a figura 13.

Esse componente também inclui o template e o estilo do componente que são responsáveis pelos elementos HTML e CSS que serão renderizados pelo navegador do usuário. Serviço Responsável por Registrar o Usuário.

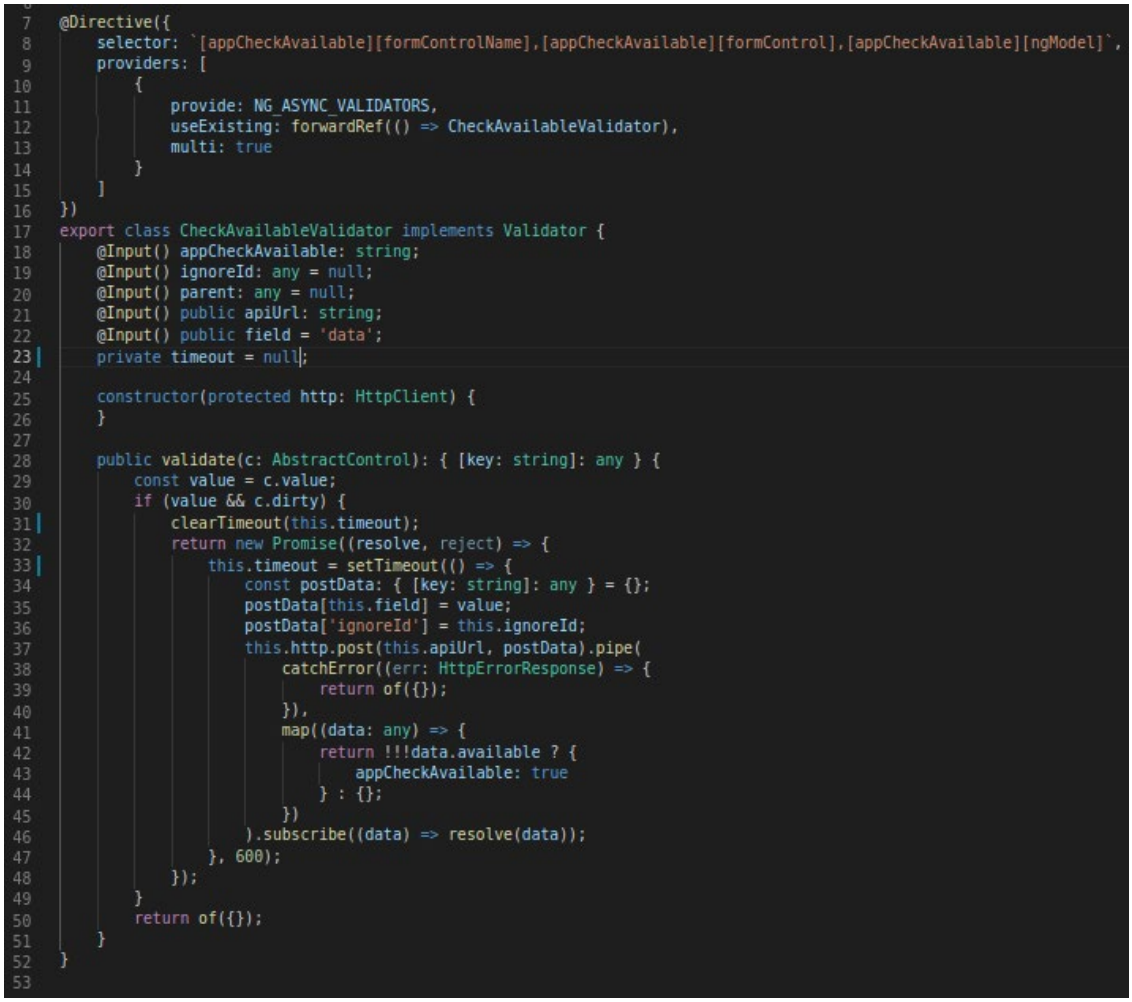
Figura 13 - Componente da Página de Registrar Usuário

```
7
8   @Component({
9       selector: 'app-register-page',
10      templateUrl: './register-page.component.html',
11      styleUrls: [
12          './register-page.component.less'
13      ]
14  })
15  export class RegisterPageComponent {
16      public signupForm: FormGroup;
17      public loading = false;
18      public sexos = [
19          {label: 'Masculino', value: 'M'},
20          {label: 'Feminino', value: 'F'}
21      ];
22      public vinculos = [
23          {label: 'Aluno', value: 0},
24          {label: 'Servidor', value: 1},
25          {label: 'Comunidade', value: 2}
26      ];
27
28      constructor(
29          private route: Router,
30          fb: FormBuilder,
31          private auth:
32              AuthService,
33          public toastr: ToastService
34      ) {…
50      }
51
52      onSubmit(value: any) {…
78      }
79  }
80
```

Fonte: Os Autores

Durante o desenvolvimento da página, caso se faça necessário a criação de diretivas de validação e outros componentes dentro da arquitetura, então será realizada a criação desses componentes complementares. A figura 14 ilustra uma diretiva de validação de e-mail, diretiva essa responsável por verificar se um e-mail está disponível para a utilização.

Figura 14 - Diretiva de Validação de E-mail Disponível



```
@Directive({
    selector: `[appCheckAvailable][formControlName],[appCheckAvailable][formControl],[appCheckAvailable][ngModel]`,
    providers: [
        {
            provide: NG_ASYNC_VALIDATORS,
            useExisting: forwardRef(() => CheckAvailableValidator),
            multi: true
        }
    ]
})
export class CheckAvailableValidator implements Validator {
    @Input() appCheckAvailable: string;
    @Input() ignoreId: any = null;
    @Input() parent: any = null;
    @Input() public apiUrl: string;
    @Input() public field = 'data';
    private timeout = null;

    constructor(protected http: HttpClient) {
    }

    public validate(c: AbstractControl): { [key: string]: any } {
        const value = c.value;
        if (value && c.dirty) {
            clearTimeout(this.timeout);
            return new Promise((resolve, reject) => {
                this.timeout = setTimeout(() => {
                    const postData: { [key: string]: any } = {};
                    postData[this.field] = value;
                    postData['ignoreId'] = this.ignoreId;
                    this.http.post(this.apiUrl, postData).pipe(
                        catchError((err: HttpErrorResponse) => {
                            return of({});
                        }),
                        map((data: any) => {
                            return !!!data.available ? {
                                appCheckAvailable: true
                            } : {};
                        })
                    ).subscribe((data) => resolve(data));
                }, 600);
            });
        }
        return of({});
    }
}
```

Fonte: Os Autores

Finalizando o desenvolvimento da página e seus elementos auxiliares é realizada a configuração das rotas do front-end na qual o usuário acessará para utilizar a funcionalidade. A figura 15 demonstra a classe responsável pelo registro de rotas da aplicação.

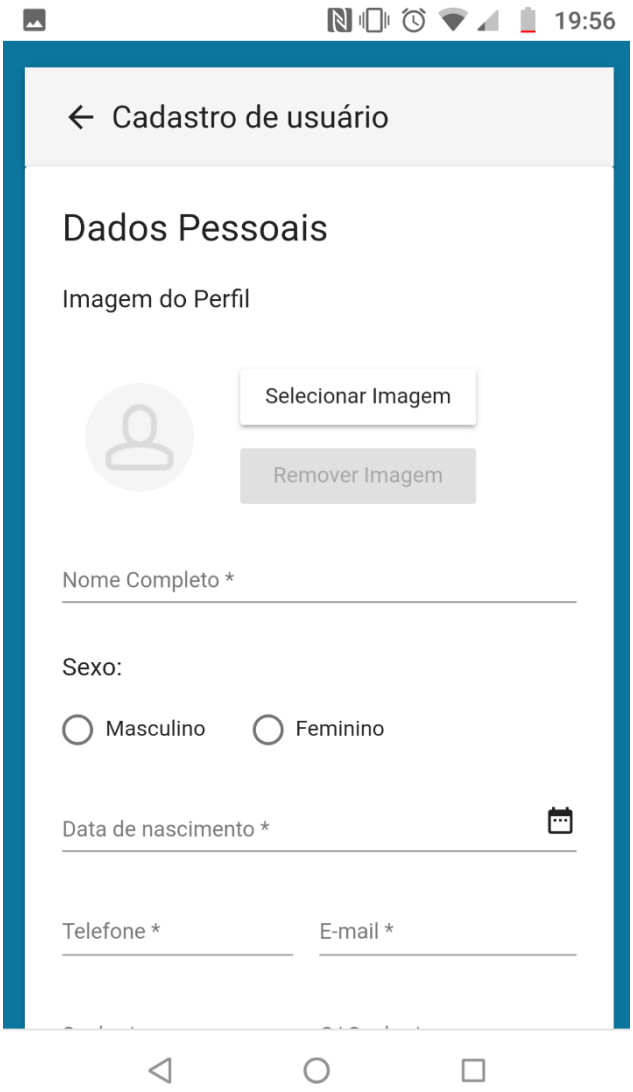
Por fim são realizados testes manuais na funcionalidade a fim de garantir a qualidade da mesma, concluindo o desenvolvimento da tarefa de front-end. A figura 16 demonstra o resultado da tela de registro de usuário.

Figura 15 - Configuração de Rotas da Funcionalidade Registrar Usuário

```
20  const generateBreadTitleForEntity = (prefix, field) => {…
24  };
25
26  const routes: Routes = [
27      {
28          path: 'auth',
29          data: {…
31          },
32          children: [
33              {…
39              },
40              {
41                  path: 'cadastro',
42                  component: RegisterPageComponent,
43                  data: {
44                      breadcrumb: 'Cadastro'
45                  }
46              },
47              {…
53              },
54              {…
60              }
61          ]
62      },
63      {…
177     }
178 ];
179
180 @NgModule({
181     imports: [RouterModule.forChild(routes)],
182     exports: [RouterModule]
183 })
184 export class ApplicationRoutingModule {
185 }
186
```

Fonte: Os Autores

Figura 16 - Tela da Funcionalidade Registrar Usuário



Fonte: Os Autores

3.1.6 *Sprint* 03 – Desenvolvimento US003 e US004 e US010

A *Sprint* 03 deu-se início na reunião de planejamento na qual foram selecionados os seguintes requisitos para o desenvolvimento, de acordo com as suas prioridades: Programação (US003), Manter Programação (US004) e Visualizar os Participantes (US010). Os requisitos são de prioridade essencial e foram desenvolvidos conforme o modelo padrão apresentado nas seções 3.1.3.3.4 e 3.1.3.3.5 cujo o resultado foram os apêndices E, F e L. Posteriormente foi realizada a revisão da *Sprint* juntamente com os *stakeholders*. A *Sprint* fechou com a retrospectiva.

3.1.7 *Sprint* 04 – Desenvolvimento US005 e US006 e US008

Na reunião de planejamento da *Sprint* 04 os requisitos selecionados foram: Manter Formulário de Avaliação (US005), Autorizar *Check-in* (US006), Editar Usuário (US008). Os três requisitos são de prioridade essencial. Assim como na *Sprint* 03, o desenvolvimento seguiu os modelos apresentado nas seções 3.1.3.3.4 e 3.1.3.3.5, resultando nos apêndices G, H e J. Após, foi realizada a revisão da *Sprint* e sucessivamente a retrospectiva.

3.1.8 *Sprint* 05 – Desenvolvimento US007 e US009

A *Sprint* 05 foi desenvolvida com base nos requisitos Realizar Sorteio (US007) e Notificar Programação (US009), selecionados na reunião de planejamento. O requisito US007 é de prioridade importante e o US009 de prioridade poderia[1]. Também seguindo os modelos das seções 3.1.3.3.4 e 3.1.3.3.5 para o desenvolvimento, foram gerados os apêndices I e K. Posteriormente, foi realizada a revisão da *Sprint* e em seguida a retrospectiva, finalizando assim a *Sprint* 05.

3.1.9 Homologação e resultados

A seguinte sessão apresenta os passos realizados na homologação do sistema assim como os defeitos detectados e as correções realizadas. Além disso, contém os resultados obtidos com a aplicação do sistema durante o evento SITES.

3.1.9.1 Homologação

Para realizar a homologação do sistema foi separado um ambiente de homologação denominado *staging*. O mesmo foi configurado no Heroku (plataforma em nuvem como um serviço que suporta várias linguagens de programação permitindo a hospedagem de aplicações web). A figura 17 representa as configurações de servidores disponíveis para utilização no Heroku.

[1] São os requisitos que poderiam ser implementados no sistema. Não alteram o funcionamento caso não estejam implementados. O sistema pode funcionar de forma satisfatória sem eles.

Figura 17 - Configurações dos Servidores

Dyno Type	Memory (RAM)	CPU Share	Compute	Dedicated	Sleeps
free	512 MB	1x	1x-4x	no	yes
hobby	512 MB	1x	1x-4x	no	no
standard-1x	512 MB	1x	1x-4x	no	no
standard-2x	1024 MB	2x	4x-8x	no	no
performance-m	2.5 GB	100%	11x	yes	no
performance-l	14 GB	100%	46x	yes	no

Fonte: Documentação Heroku (2018).

A princípio foi utilizado um servidor *free* (gratuito)para a configuração do ambiente de homologação. Em seguida o sistema foi liberado para os stakeholders realizarem o cadastro das programações e questionários de avaliação da XIX edição do SITES. O evento foi realizado nos dias 27 a 29 de outubro de 2018 no Centro Universitário de Anápolis – UniEVANGÉLICA, com início às 19 horas e término às 22 horas.

No decorrer do evento foi realizado o monitoramento do sistema e do ambiente de homologação por parte da equipe de desenvolvimento, objetivando a detecção de possíveis falhas na aplicação do sistema.

Na terça-feira (27/11/2018), primeiro dia do evento, foram identificados os seguintes problemas:

1. A controle do check-in não estava disponível nos dispositivos da plataforma IOS devido a um problema de incompatibilidade com a câmera dos mencionados dispositivos;
2. Problema na alteração do usuário, especificamente no campo de papel do usuário (realizar a remoção do papel de auxiliar e/ou administrador do sistema);
3. Dificuldade dos usuários ao responder os questionários de avaliação no que se diz respeito a opção “Salvar”. A mesma ficava desabilitada enquanto o usuário não respondesse todos os campos obrigatórios do questionário. O problema aqui se deu devido à falta de clareza de quais questões eram obrigatórias;

Já no segundo dia do evento, quarta-feira (28/11/2018), os problemas levantados foram:

1. Durante o cadastro do formulário de avaliação notou-se que a responsividade da tela não estava com barra de rolagem sendo necessário diminuir o zoom da tela para realizar o cadastro do mesmo.
2. Foi verificado que apenas uma pequena parte dos usuários responderam o questionário de avaliação do dia anterior em consequência de que por padrão o sistema desabilitava a funcionalidade de responder o questionário caso o dia da programação fosse diferente do dia atual, visto isso só era possível responder o questionário no dia da programação.

Os problemas detectados no primeiro dia geraram hotfix's[2] no backlog da sprint. O primeiro problema da terça-feira (27/11), devido ao fato da equipe de desenvolvimento não possuir um dispositivo da plataforma IOS para realização dos testes, não foi possível realizar a sua avaliação e correção, gerando assim um trabalho futuro para sistema. Já os problemas 2 e 3, bem como o problema 1 do segundo dia, quarta-feira (28/11), foram problemas pontuais no código fonte aos quais foram aplicadas as devidas correções. O segundo problema ocorrido na quarta-feira foi resolvido realizando uma alteração na regra da funcionalidade, permitindo a submissão de formulários de avaliação no dia da programação ou após o dia da mesma.

No terceiro e último dia, quinta-feira (29/11/2018), foi detectado apenas um problema no sistema. Problema esse que resultou na indisponibilidade do sistema por 20 minutos. Foi verificado que durante o início do evento, por volta das 19 horas, quando se dá a realização dos check-ins, ocorreu um pico muito alto de requisições no sistema, superior a capacidade de processamento do servidor. Os usuários, antes de chegarem na mesa de check-in, abriam a tela do QRCode, com o intuito de otimizar o tempo do check-in. O problema, porém, se deu devido ao tempo de requisição do sistema, que a cada 2 segundos realizava uma requisição para verificar se o check-in do usuário em questão tinha sido confirmado. Com isso de acordo com que o número de usuários aumentava as requisições do sistema também cresciam. Com a grande carga de requisições no ambiente *free*, descrito na figura 17, sobrecarregou o servidor fazendo com que o ambiente do Heroku rejeitasse as requisições da maioria dos usuários, deixando assim o tempo de resposta muito lento.

Após a detecção do problema foi iniciado por parte da equipe de desenvolvimento os procedimentos necessários para a correção do mesmo. A princípio foi aumentado o intervalo de requisições de 2 para 5 segundos, diminuindo assim consideravelmente a taxa de requisições

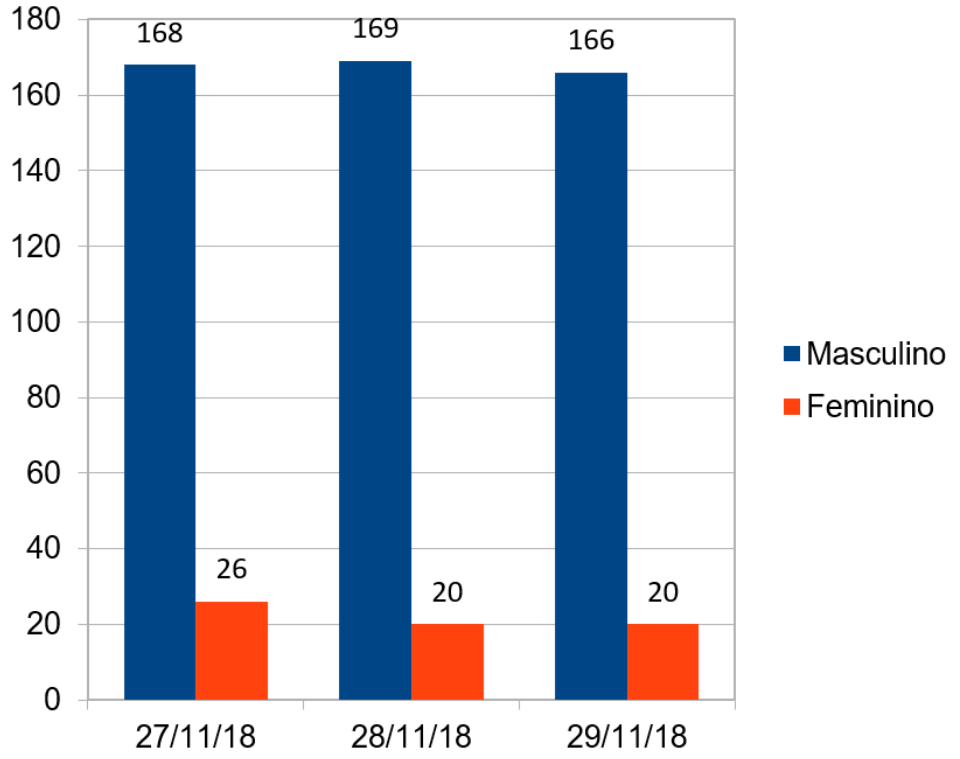
[2] Reparo do software feito com o sistema em atividade.

por minuto no sistema. Porém ainda não foi o suficiente para que o sistema voltasse a operar normalmente, apresentando ainda um tempo de resposta muito elevado, entre 25 a 30 segundos, devido à grande quantidade de usuários tentando realizar check-in. Com isso o próximo passo foi realizar a escalação do sistema alterando sua capacidade de processamento do servidor *free* para o servidor do tipo *standard-x1*. Com configurações semelhantes ao servidor free, o servidor *standard*-x1 não foi suficiente para correção do problema, sendo necessário uma segunda escalação, dessa vez para o tipo *standard-x2,* com uma capacidade de processamento superior aos anteriores. Mesmo com a segunda escalação, ainda assim o sistema apresentava certa lentidão. Isto posto, a equipe de desenvolvimento resolveu aumentar ainda mais a capacidade de processamento, no qual foram disponibilizados 4 servidores do tipo standard-x2 durante o momento de pico do sistema, resultando dessa vez na solução do problema exposto. Conforme os check-ins foram sendo finalizados foi possível regredir o servidor gradativamente para as versões anteriores, já que após o momento de pico não era mais necessário tamanho poder de processamento. Consequentemente foi constatado que o "laço" (*loop*) de requisições com uma grande quantidade de usuários pode causar problemas de instabilidade no sistema gerando assim outro ponto de melhoria. A aplicação da tecnologia de *websocket* nessa parte em específico do sistema, substituindo o loop de requisições, solucionaria este problema.

3.1.9.2 Resultados

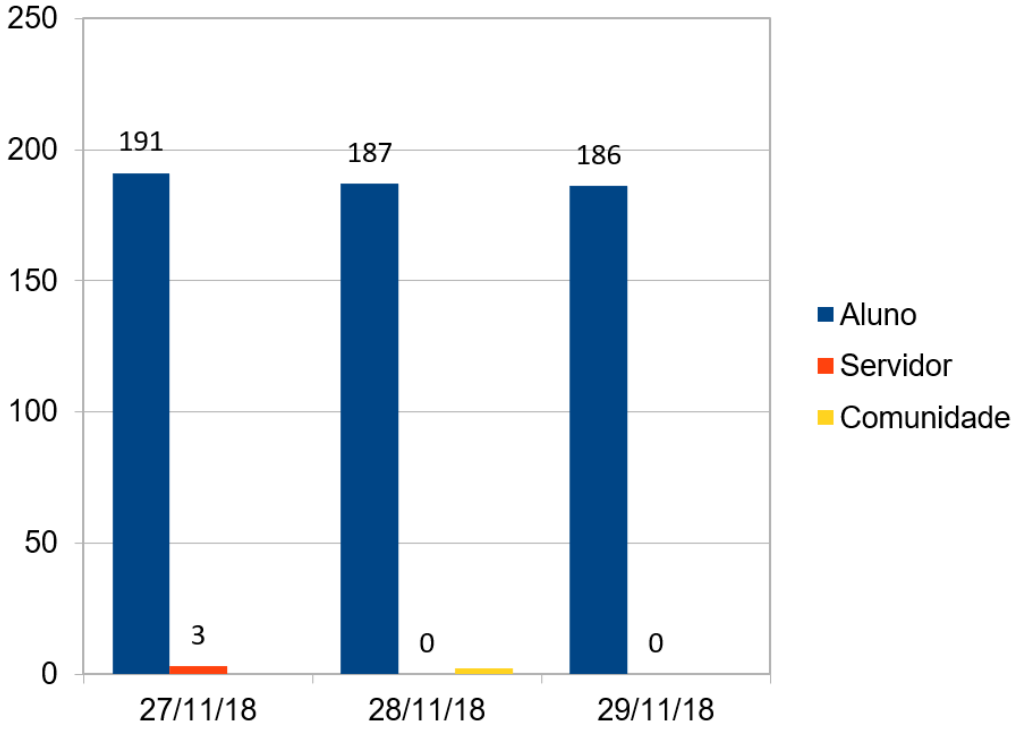
Por meio da homologação do sistema se fez possível confirmar a eficiência do software na extração dos dados dos participantes do evento, como por exemplo: nome, matrícula, vínculo com a instituição, gênero, se o participante foi sorteado, entre outros. Após a compilação dos dados extraídos do sistema pode-se elaborar diversos gráficos para melhor representação dos mesmos. A imagem 1 representa a quantidade de check-ins por gênero em cada dia do evento. Na imagem 2 observa-se a quantidade de check-ins por vínculo (aluno, servidor, comunidade). Já a imagem 3 mostra a relação entre a quantidade de participantes e a quantidade de feedbacks recolhidos, bem como o número de questões nos questionários.

Figura 18 - Gráfico de gêneros



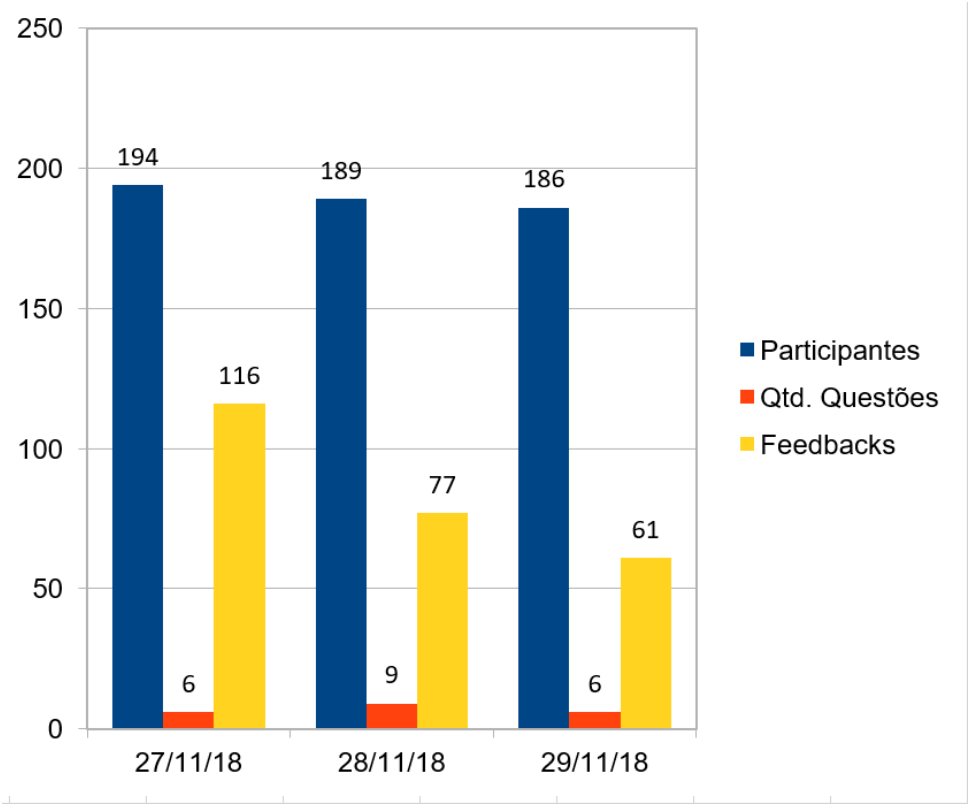
Fonte: Os autores.

Figura 19 - Gráfico de vínculos



Fonte: Os autores.

Figura 20 - Gráfico de *feedbacks*



Fonte: Os autores.

4 CONSIDERAÇÔES FINAIS

A realização desse projeto auxiliou no crescimento pessoal dos autores, possibilitando aplicar na prática toda teoria adquirida no decorrer da graduação, realizando pesquisas e aplicando ferramentas e métodos para resolução do problema proposto.

Com a aplicação do BDD foi possível desenvolver o software em uma linguagem não ambígua de forma que todos os envolvidos entendam com clareza os requisitos. Dessa forma foi possível realizar a validação das mesmas de forma fácil quanto ao seu entendimento, tanto pela equipe de desenvolvimento, quanto pelas demais partes interessadas.

Outro ganho notável com a metodologia BDD foi a possibilidade de mapear diversos cenários para as funcionalidades do software resultando assim em um software mais abrangente aos diversos contextos possíveis. O BDD auxiliou também na criação de histórias de usuário curtas e viáveis a testes. Mesmo com a aplicação da metodologia ao decorrer do desenvolvimento se fez necessário uma divisão adicional de uma história de usuário em outras menores. Histórias de Usuários curtas são de mais fácil validação e entendimento.

A aplicação do Scrum no projeto auxiliou no gerenciamento das tarefas, obtenção de *feeedbacks* contínuos e na transparência do mesmo. Auxiliou também no controle do backlog e na divisão de tarefas. Conforme as *sprints* foram realizadas e os incrementos de valor adicionados ao produto final, por meio das revisões das *sprints,* foi possível obter *feedback's* dos clientes a fim de realizar ajustes e aplicar a melhorias no produto para que o resultado final atendesse as expectativas dos clientes. Com isso, este trabalho ganhou não apenas agilidade no seu desenvolvimento, mas também um produto com maior qualidade, implementado em menos tempo e com riscos calculados.

Com a utilização dos frameworks Laravel e Angular foi possível obter maior produtividade durante o desenvolvimento do software devido ao fato de que os *frameworks* disponibilizam uma gama de funcionalidades já implementadas fazendo-se necessário apenas utilizá-las, com isso o foco fica apenas no domínio de informação do sistema e nas suas regras de negócio.

Como possíveis trabalhos futuros, pode-se apontar:

- Aplicação da tecnologia *websocket* no requisito Autorizar Check-in, para melhorar a interação do usuário e obter um resultado em tempo real da confirmação do mesmo;
- Criação de relatórios customizáveis;

- Melhorias no suporte da plataforma IOS;
- Configuração da plataforma em um ambiente disponibilizado pela instituição;
- Extensão do sistema para todos os eventos da instituição;

REFERÊNCIAS BIBLIOGRÁFICAS

BECK, K. et al. **Manifesto para o desenvolvimento ágil de software**. 2001. Disponível em: < http://www.manifestoagil.com.br/> Acesso em: 12 abr. 2018.

BERNARDO, K. **Backlog Grooming**: refinando o backlog. jul. 2015. Disponível em: < https://www.culturaagil.com.br/backlog-grooming-refinando-o-backlog/> Acesso em: 21 maio 2018.

COHN, M. **User Stories Applied For Agile Software Development**. [S.l.]: Addison-Wesley Professional, 2004.

GIT. **Primeiros passos – Sobre Controle de Versão**. Disponível em: <https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o/> Acesso em: 21 abr. 2018.

HEUSER, C.A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

HEROKU. **Dyno Types**. Disponível em: < https://devcenter.heroku.com/articles/dyno-types> Acesso em: 05 dez. 2018.

HUMBLE, J.; FARLEY, D. **Entrega contínua**: como entregar software de forma rápida e confiável. 8. Ed. Porto Alegre: Bookman, 2014.

KORTH, H.F; SILBERSCHATZ, A; SUDARSHAN, S; **Sistemas de Bancos de Dados**. 6. ed. Eslavier, 2012.

MOZILLA. **Javascript**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/> Acesso em: 13 abr. 2018.

NORTH, D. **Introducing BDD**. [S.l.]; Better Software magazine, 2006.

NORTH, D. **What's in a Story?** [2007]. Disponível em: <https://dannorth.net/whats-in-a-story>. Acesso em: 12 abr. 2018.

PADUA FILHO, W. P**. Engenharia de Software**: fundamentos, métodos e padrões. 3. ed. Rio de Janeiro: LTC, 2008.

PHP. **Manual do PHP**. Disponível em: < http://php.net/manual/pt_BR/> Acesso em: 11 abr. 2018.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software**: Uma Abordagem Profissional. 8. Ed. Porto Alegre: McGraw-Hill, 2016.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos Ágeis para Desenvolvimento de Software**. 3. ed. Porto Alegre: Bookman, 2014.

RUGGIERO, LUCAS. **Desenvolvimento Guiado por Comportamento (BDD)**. 2017. Disponível em <https://medium.com/@partnerfusionbrasil/desenvolvimento-guiado-por-comportamento-bdd-9006d69e97e7> Acesso em: 12 abr. 2018.

SABBAGH, RAFAEL. **Gestão Ágil para Projetos de Sucesso**. 2014. Disponível em: <https://docgo.net/philosophy-of-money.html?utm_source=pdf-scrum-gestao-agil-para-projetos-de-sucesso-de-rafael-sabbagh-baixar-livros&utm_campaign=download> Acesso em: 12 abr. 2018.

SCHWABER, K; SUTHERLAND, J. **Guia do ScrumMR**. 2017. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf> Acesso em: 12 abr. 2018.

SMART, J. F. **BDD in Action**. [S.l.]; Manning Publications, 2014.

SOLÍS, C; WANG, X. **A Study of the Characteristics of Behaviour Driven Development**. 2011. Disponível em: <http://damiantgordon.com/Methodologies/Papers/A%20Study%20of%20the%20Characteristics%20of%20Behaviour%20Driven%20Development.pdf>. Acesso em: 21 abr. 2018.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, W. **Brainstorm**: o que é e quais são suas vantagens. set. 2012. Disponível em: < http://www.blogcmmi.com.br/o-que-e/brainstorm-o-que-e-e-quais-sao-suas-vantagens/> Acesso em: 21 maio 2018.

WILDT, D. et al. **Extreme programming**: práticas para o dia a dia no desenvolvimento ágil de software. [S.l.]; Casa do Código, 2015.

APÊNDICES

APÊNDICE A – DOCUMENTO DE VISÃO

DOCUMENTO DE VISÃO

Sistema para controle do evento SITES

VERSÃO: 0.6

Autores:
Gabriel Leandro Júnior Siqueira
Pedro Victor de Oliveira e Silva

Anápolis – GO
2018

HISTÓRICO DE REVISÃO

Versão	Data	Responsável	Descrição
0.1	15/02/2018	Gabriel Leandro J. Siqueira	Início do documento de visão.
0.2	04/04/2018	Pedro Victor de O. e Silva	Revisão do documento de visão. Alterações na Visão do Produto.
0.3	21/05/2018	Gabriel Leandro J. Siqueira e Pedro Victor de O. e Silva	Adicionar etapas do processo.
0.4	19/06/2018	Pedro Victor de O. e Silva	Alteração no tempo da Sprint e correção de ortografia.
0.5	10/09/2018	Pedro Victor de O. e Silva	Alteração do problema, visão do produto e inclusão de nova abreviatura.
0.6	09/11/2018	Gabriel Leandro J. Siqueira e Pedro Victor de O. e Silva	Atualização dos requisitos funcionais e da sprint, e revisão do documento em si

SUMÁRIO

1 STAKEHOLDERS

JANEIRO - 2018/1		
Nome	**Cargo**	**Responsabilidade**
Gabriel Leandro Júnior Siqueira	Desenvolvedor e Scrum Master	Responsável pela implementação das funcionalidades do software e garantir as boas práticas do Scrum
Adrielle Beze Peixoto	Cliente	
Millys Fabrielle Araujo Carvalhaes	Cliente	
Pedro Victor de Oliveira e Silva	Product Owner	Responsável pelo Product Backlog

2 ETAPAS DO PROCESSO

2.1 Visão

- Análise do escopo do projeto;
- Definição do escopo do projeto;
- Definição das partes interessadas.

2.2 Desenvolvimento PB

- Levantamento de Requisitos (Histórias de Usuários);
- Elaboração do Diagrama de Casos de Uso.

2.3 Reuniões de Planejamento da Sprint

- Definição do nível de prioridade dos requisitos;
- Levantamento dos requisitos essenciais para atingir os resultados esperados.

2.4 Desenvolvimentos da Sprint

- Reunião diária;
- Será utilizado a metodologia BDD;
- Será aplicado a metodologia ágil Scrum para otimizar o desenvolvimento do software;
- Sprint de 25 dias para obter melhores resultados;
- Desenvolver a arquitetura do projeto;
- Realizar Testes para verificar e validar os artefatos e funcionalidades implementadas do software;
- Desenvolver o banco de dados.

2.5 Revisão

- Duração de 1 hora para analisar os artefatos desenvolvidos, dificuldades enfrentadas e solução adquiridas na *sprint*.

2.6 Retrospectiva

- Duração de 1 hora;
- Análise dos pontos positivos e negativos das atividades/integrantes durante a *Sprint;*
- Definição de melhorias para as próximas *sprints* procurando sempre otimizar os resultados do projeto.

3 PROBLEMA

O problema de demora para geração de certificados, dificuldade no controle de presença e realização de sorteios para os participantes, e geração de relatórios do evento SITES (Seminário Interdisciplinar de Tecnologia e Sociedade) da UniEVANGÉLICA **afeta** os funcionários e alunos da instituição. **O impacto do problema é** grande, pois além de fazer com que os funcionários da instituição tenham que conferir dezenas de atas de forma manual para gerar os certificados, faz com que os alunos tenham que ir atrás da ata de presença após o término do evento, trazendo transtornos para os mesmos e para os professores. **Uma solução bem-sucedida** incluiria um sistema que gerenciasse as presenças dos participantes de forma segura e automatizada, realizasse os sorteios entre os alunos que fizeram check-in no respectivo dia, centralizasse os dados em um só local e gerasse relatórios online de cada evento.

4 VISÃO DO PRODUTO

Para alunos e funcionários dos Cursos Superiores de Computação da UniEVANGÉLICA **que necessitam** gerenciar o evento SITES. **É um** software com plataforma web para gestão do evento SITES. **Possibilita** que alunos e funcionários realizem atividades relacionadas ao processo de controle de presença, extração de métricas através de formulários e sorteios para os participantes do evento SITES. **O diferencial** do produto é a eficiência com que o software consegue operar e principalmente a segurança dos dados informados. **Nosso produto** proporcionará uma modernização e principalmente agilidade nas atividades que são desempenhadas no controle do SITES da UniEVANGÉLICA.

5 ABREVIATURAS, CONVENÇÕES E SIGLAS

Todas as siglas, abreviações, convenções e estrangeirismos existentes nos documentos do projeto estarão relatados na tabela abaixo.

Siglas	Descrição
Manter	Desempenha as seguintes ações no sistema: Inserir, Alterar, Visualizar, Ativar e Inativar.
Controlar	Desempenha as seguintes ações no sistema: Inserir, Alterar, Visualizar.
SITES	Seminário Interdisciplinar de Tecnologia e Sociedade

5.1 Prioridade dos Requisitos

Para estabelecer a prioridade dos requisitos, foi adotada a técnica **MOSCOW**, onde:

- **Essencial: "MUST"**, referem-se aos requisitos que necessitam ser implementados prioritariamente, sem eles não há o funcionamento do sistema;
- **Importante: "SHOULD"**, são referidos aos requisitos que deveriam ser implementados no sistema, ele pode funcionar sem eles ainda que de forma não satisfatória;
- **Poderia: "COULD"**, são os requisitos que poderiam ser implementados no sistema, não alteram o funcionamento caso não estejam implementados, o sistema pode funcionar de forma satisfatória sem eles;
- **Interessante: "WANT"**, são os requisitos que seriam interessantes haver no sistema.

6 REQUISITOS FUNCIONAIS

6.1 Autenticar Usuário

US.	Descrição	Status	Prioridade
US001	Autenticar Usuário	Concluído	Essencial
Detalhamento: Este requisito permite que o acesso as funcionalidades fiquem restritos apenas aos usuários do sistema, focando assim a segurança das informações cadastradas.			

6.2 Registrar Usuário

US.	Descrição	Status	Prioridade
US002	Registrar Usuário	Concluído	Essencial
Detalhamento: Este requisito permite aos usuários cadastrarem-se no sistema.			

6.3 Programação

US.	Descrição	Status	Prioridade
US003	Programação	Concluído	Essencial
Detalhamento: Este requisito permite aos usuários visualizarem a programação do SITES.			

6.4 Manter Programação

US.	Descrição	Status	Prioridade
US004	Manter Programação	Concluído	Essencial
Detalhamento: Este requisito permite ao ator realizar todas as ações de manter na programação do evento no sistema.			

6.5 Manter Formulário de Avaliação

US.	Descrição	Status	Prioridade
US005	Manter Formulário de Avaliação	Concluído	Essencial
Detalhamento: Este requisito permite ao ator realizar todas as ações de manter um formulário de avaliação de um evento no sistema.			

6.6 Autorizar Check-in

US.	Descrição	Status	Prioridade
US006	Autorizar Check-in	Concluído	Essencial
Detalhamento: Este requisito permite ao ator autorizar um check-in em um evento no sistema.			

6.7 Realizar Sorteio

US.	Descrição	Status	Prioridade
US007	Realizar Sorteio	Concluído	Importante
Detalhamento: Este requisito permite ao ator realizar sorteios para eventos no sistema.			

6.8 Editar Usuário

US.	Descrição	Status	Prioridade
US008	Editar Usuário	Concluído	Essencial
Detalhamento: Este requisito permite ao ator editar usuários e poder atribuir permissões no sistema.			

6.9 Notificar Programação

US.	Descrição	Status	Prioridade
US009	Notificar Programação	Concluído	Poderia
Detalhamento: Este requisito permite que o sistema notifique por e-mail os usuários cadastrados no sistema sobre a programação do evento.			

6.10 Visualizar Participantes

US.	Descrição	Status	Prioridade
US0010	Visualizar Participantes	Concluído	Essencial
Detalhamento: Este requisito permite visualizar os participantes de uma programação no sistema.			

7 REQUISITOS NÃO FUNCIONAIS

7.1 Funcionalidade

Descrição	Status	Prioridade
Usabilidade	Concluído	Essencial
Subcaracterística	**Detalhamento**	
Adequação	Esta subcaracterística permite verificar o quanto as funcionalidades do software é adequando para o ambiente do usuário final.	
Segurança	Esta subcaracterística permite que o software proteja as informações do usuário e fornecê-las somente a pessoas autorizadas.	

7.2 Confiabilidade

Descrição	Status	Prioridade
Confiabilidade	Concluído	Essencial
Subcaracterística	**Detalhamento**	
Maturidade	Esta subcaracterística permite que o software evite falhas decorrentes de defeitos no software.	
Recuperabilidade	Esta subcaracterística permite que o software se recupere após uma	

	falha, restabelecendo seus níveis de desempenho e recuperando seus dados.

7.3 Usabilidade

Descrição	Status	Prioridade
Usabilidade	Concluído	Essencial
Subcaracterística	**Detalhamento**	
Apreensibilidade	Esta subcaracterística representa a capacidade de facilitar o aprendizado do sistema para os usuários finais.	
Operacionalidade	Esta subcaracterística representa a capacidade de facilitar a operação das funcionalidades por parte do usuário.	

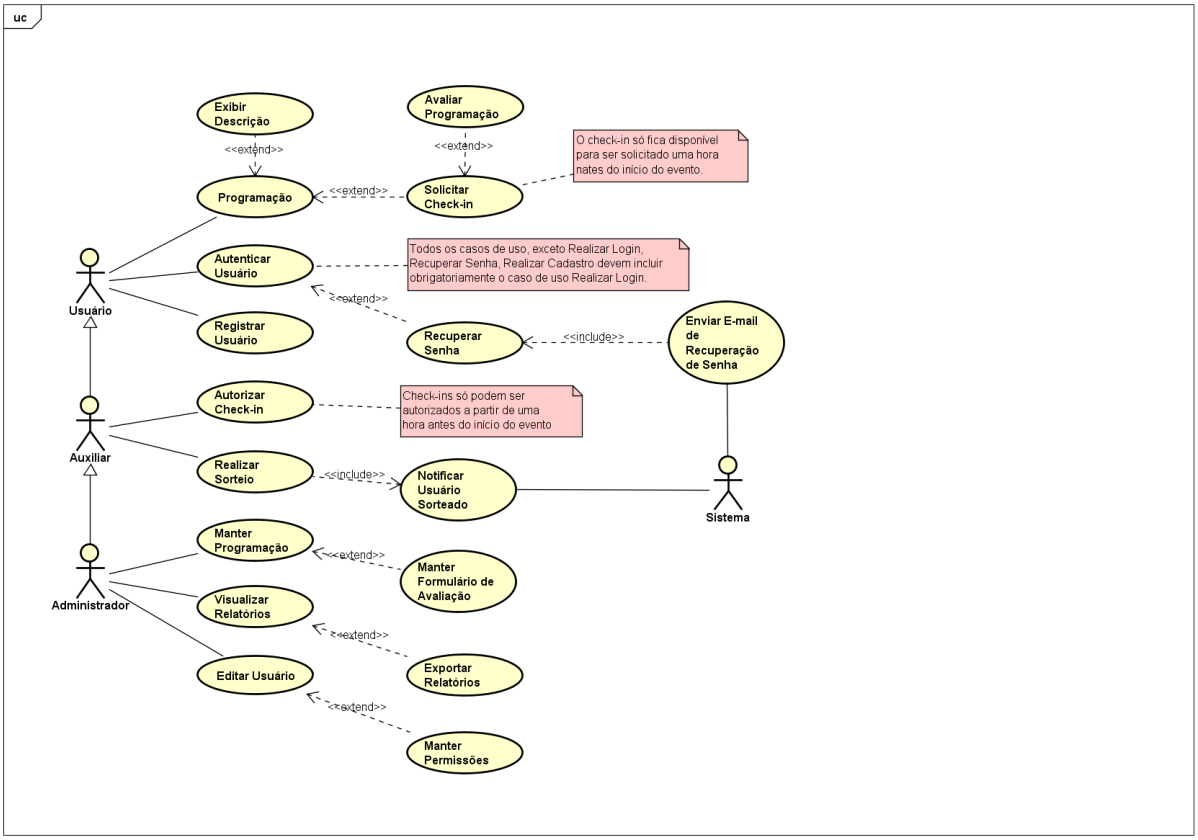
7.4 Manutenibilidade

Descrição	Status	Prioridade
Manutenibilidade	Concluído	Essencial
Subcaracterística	**Detalhamento**	
Estabilidade	Esta subcaracterística representa a capacidade do software de evitar efeitos colaterais decorrentes de modificações introduzidas.	
Testabilidade	Esta subcaracterística representa a capacidade de se testar o sistema atualizado, novas funcionalidades e funcionalidades sem vínculo com a modificação.	

7.5 Portabilidade

Descrição	Status	Prioridade
Portabilidade	Concluído	Essencial
Subcaracterística	**Detalhamento**	
Adaptabilidade	Esta subcaracterística representa a capacidade do software de se adaptar a diferentes ambientes sem necessidade de configurações.	

APÊNDICE B – DIAGRAMA DE CASO DE USO



APÊNDICE C – US001 AUTENTICAR USUÁRIO

```
#language: pt

Funcionalidade: Autenticar Usuário
  Com o objetivo de autenticar o usuário no sistema
  Como um Usuário
  Eu quero poder informar minhas credenciais de acesso para poder visualizar a programação do
sites
  e demais informações disponíveis no sistema

  @positivo
  Cenário: Realizar login com informações válidas
    Dado que meu e-mail <e-mail> e senha <senha> já estejam cadastrados no sistema
    Quando eu informar meu e-mail <e-mail> e senha <senha>
    E clicar no botão "Acessar"
    Então o sistema deve me redirecionar para a "Página de Agenda de Programação"

  Exemplos:
  | e-mail | senha |
  | gabrielleandrojunior@live.com | senhavalida |

  @negativo
  Cenário: Realizar login com informações inválidas
    Dado que meu <e-mail valido> e <senha valida> já estejam cadastrados no sistema
    Quando eu informar um e-mail válido <e-mail>
    E uma senha inválida <senha invalida>
    OU eu informar um e-mail inválido <e-mail invalido>
    E uma senha válida <senha valida>
    E clicar no botão "Acessar"
    Então o sistema deve apresentar a mensagem "Ops! Usuário ou senha incorretos."

  Exemplos:
  | e-mail valido | e-mail invalido | senha valida | senha invalida |
  | gabrielleandrojunior@live.com | gabrielleandro@teste.com | senhavalida | senhainvalida |

  @positivo
  Cenário: Recuperar senha
    Dado que eu já tenha realizado o meu cadastro e não lembre a minha senha
    Quando eu clicar no link "Esqueceu sua senha?"
    Então o sistema deve me redirecionar para "Página de Recuperação de Senha"

  @positivo
  Cenário: Solicitar recuperação de senha com e-mail cadastrado
    Dado que eu esteja na tela de recuperação de senha
    Quando eu informar um <e-mail> cadastrado no sistema
    E clicar no botão "Enviar solicitação de recuperação"
    Então o sistema deve me redirecionar para a "Página de Login"
    E apresentar a mensagem "O link para redefinição de senha foi enviado para o seu e-mail."
    E o sistema deve gerar um token de acordo com as regras <regras>
    E enviar para o e-mail cadastrado o e-mail de recuperação de senha de acordo com o modelo:
        """
            Olá <nome do usuário>!
            ======================

            Você está recebendo este e-mail
            porque recebemos uma solicitação
            de recuperação de senha para sua conta.

            [Redefinir Senha]

            Se você não solicitou uma recuperação
            de senha, nenhuma ação adicional será
            necessária, não se preocupe!

            Atenciosamente,
            SITES

            ========================

            Se você tiver problemas para clicar no botão
```

```
            "Redefinir Senha", copie e cole o URL abaixo no
            seu navegador: [url]
        """
  Exemplos:
  | regras |
  | O token deve ser único |
  | O token pode ser utilizado apenas uma vez |
  | Após a utilização do token o mesmo deve ser invalidado e deletado |

  @negativo
  Cenário: Solicitar recuperação de senha com e-mail não cadastrado
    Dado que eu esteja na tela de recuperação de senha
    Quando eu informar um <e-mail> diferente do e-mail cadastrado no sistema
    E clicar no botão "Enviar solicitação de recuperação"
    Então o sistema deve apresentar a mensagem "Não encontramos nenhum usuário com esse endereço
de e-mail."

  @negativo
  Cenario: Resetar Senha - Token Inválido
    Dado que eu realizei o cenário "Solicitar recuperação de senha com e-mail cadastrado"
    E acessei o link de redefinição de senha com um token inválido
    Quando preencher os campo <campos> de acordo com suas regras <regras>
    E clicar no botão "Resetar Senha"
    Então o sistema deve apresentar a mensagem "Este token de redefinição de senha é inválido."
    E me redirecionar para a "Página de Login"

  @positivo
  Cenário: Resetar senha
    Dado que eu realizei o cenário "Solicitar recuperação de senha com e-mail cadastrado"
    E acessei o link de redefinição de senha com um token válido
    Quando preencher os campo <campos> de acordo com suas regras <regras>
    E clicar no botão "Resetar Senha"
    Então o sistema deve apresentar a mensagem "Sua senha foi alterada!"
    E me autenticar
    E me redirecionar para a "Página de Programação"

  Exemplos:
  | campos | regras |
  | Senha | Obrigatório, Mínimo 6 dígitos, Confirmado |
  | C/Senha | Obrigatório |

  @negativo
  Cenário: Resetar senha - informações inválidas
    Dado que eu realizei o cenário "Solicitar recuperação de senha com e-mail cadastrado"
    E acessei o link de redefinição de senha com um token válido
    Quando preencher algum dos campo <campos> divergente de suas regras <regras>
    Então o sistema deve desativar o botão "Resetar Senha"
    E apresentar as mensagem <mensagem> de acordo com a regra <regra> abaixo do campo

  Exemplos:
  | Rega | Mensagem  |
  | Obrigatório | O campo <nome do campo> é obrigatório. |
  | Confirmada | As/Os <nome do campo> não conferem. |
  | Mínimo 6 dígitos | A/O <nome do campo> deve conter no mínimo 6 caracteres. |
```

APÊNDICE D – US002 REGISTRAR USUÁRIO

```
#language: pt

Funcionalidade: Registrar Usuário
  Com o objetivo de me cadastrar no sistema
  Como um usuário
  Eu quero poder me cadastrar no sistema para poder visualizar
  as programações dos dias do evento SITES
  e demais informações disponíveis no sistema

  Regras e Campos do Registro:
  | Campos | Regas |
  | Nome Completo  | Obrigatório, Máximo 50 caracteres |
  | E-mail | Obrigatório, Validação de e-mail, Deve ser único |
  | Telefone Celular | Obrigatório, Validação de telefone, Deve ser único |
  | Senha | Obrigatório, Confirmada, Mínimo 6 dígitos |
  | Imagem do Perfil | Opicional |
  | Sexo | Obrigatório |
  | Data de Nascimento | Obrigatório, Validação de data |
  | Vínculo | Obrigatório, (Aluno, Servidor, Comunidade) |
  | Matrícula | Obrigatório se o vínculo for diferente de Comunidade, Validação de Matrícula (7
dígitos numéricos) |

  @positivo
  Cenário: Registrar usuário com informações válidas
    Dado que preenchi todos os campos <campos> corretamente de acordo com sua regas <regas>
    Quando clicar no botão "Cadastrar"
    Então o sistema deve me cadastrar
    E me autenticar em seguida
    E me redirecionar para a "Página de Programação"
    E apresentar a mensagem "Seja bem-vindo ao Sites <nome do usuário>!"

  @negativo
  Cenário: Registrar usuário com informações inválidas
    Dado que preenchi todos algum dos campos <campos> divergente de alguma das suas regras
<regras>
    Quando eu tiver preenchido
    Então o sistema deve apresentar abaixo do campo a mensagem <mensagem> referente a regra
<regra>
    E desativar o botão "Cadastrar"

  Exemplos:
  | Rega | Mensagem  |
  | Obrigatório | O campo <nome do campo> é obrigatório. |
  | Validação de data | Entre com uma data válida. |
  | Validação de telefone | Entre com um telefone de e-mail válido. |
  | Validação de e-mail | Entre com um endereço de e-mail válido.  |
  | Validação de matrícula | Entre com a matrícula válida.  |
  | Confirmada | As/Os <nome do campo> não conferem. |
  | Mínimo 6 dígitos | A/O <nome do campo> deve conter no mínimo 6 caracteres. |

  @negativo
  Cenário: Registrar usuário com e-mail já cadastrado
    Dado que preenchi todos os campos <campos> corretamente com suas regras <regras>
    Quando eu preencher no campo "E-mail" um e-mail <e-mail> já cadastrado no sistema
    Então o sistema deve apresentar abaixo do campo a mensagem "E-mail já cadastrado no sistema."
    E desativar o botão "Cadastrar"

  Exemplos:
  | e-mail |
  | gabrielleandrojunior@live.com |

  @negativo
  Cenário: Registrar usuário com telefone já cadastrado
    Dado que preenchi todos os campos <campos> corretamente com suas regras <regras>
    Quando eu preencher no campo "Telefone" um telefone <telefone> já cadastrado no sistema
    Então o sistema deve apresentar abaixo do campo a mensagem "Telefone já cadastrado no
sistema."
    E desativar o botão "Cadastrar"

  Exemplos:
  | telefone |
  | (62) 99999-9999 |
```

APÊNDICE E – US003 PROGRAMAÇÃO

```
#language: pt

Funcionalidade: Programação
  Com o objetivo de listar a programação que irá acontecer no sites na instituição
  Como um usuário
  Eu quero visualizar a programação que irá acontecer no sites para que eu visualizar os detalhes
dos dias

  @positivo
  Cenário: Programação - Dias Disponíveis
    Dado que eu esteja autenticado
    Quando o sistema me redirecionar para "Página de Programação"
    Então eu posso visualizar a programação que irá acontecer o evento sites na instituição

  Regras de Apresentação:
  - Serão exibidas as Programações agrupadas por edições em forma de lista
  - Serão exibidas apenas edições que possuem programações vinculadas
  - Serão ordenadas por data de cadastro do maior para o menor

  @positivo
  Cenário: Programação - Nenhuma programação cadastrada
    Dado que eu esteja autenticado e não existe nenhum dia Cadastrado no sistema
    Quando o sistema me redirecionar para "Página de Programação"
    Então o sistema deve representar uma lista vazia com a mensagem "Nenhuma programação
disponível."

  @positivo
  Cenário: Solicitar Check-in
    Dado que eu esteja na página de "Programação"
    E que o dia de hoje seja o mesmo dia cadastrado
    Quando eu clicar no botão "Solicitar Check-in"
    Então o sistema deve apresentar o QRCode do token referente ao Check-in da programação

  Regras:
  - Deve ser gerado um token unico na solicitação do check-in
  - O token deve expirar em 30 segundos

  @positivo
  Cenário: Solicitar Check-in - Atualizar Token
    Dado que eu tenha realizado o cenário "Solicitar Check-in"
    E no exato momento em que o token expirar (30 segundos após sua geração)
    Então o sistema deve solicitar um novo token
    E apresentar o QRCode do token referente ao Check-in da programação

  @positivo
  Cenário: Avaliar Dia
    Dado que eu esteja na página de "Programação"
    E tenha realizado check-in no dia
    E a avaliação esteja disponível
    Quando eu clicar no botão "Avaliar Evento"
    Então o sistema deve apresentar o formulário de "Avaliação do dia do Evento"

  Regras da avaliação:
  - A avaliação do evento será por dia (Ex: Evento começa dia 1 e vai até dia 3, poderá ter 3
avaliações, uma para cada dia)
  - O dia do evento só tera avaliação se no cadastro do mesmo tiver sido cadastrado o formulário
de avaliação do dia do evento
  - A avaliação do evento só podera ser realizada por usuários que participaram do evento e
realizaram o Check-in do dia
  - Só é possível avaliar uma vez cada dia do evento

  @positivo
  Cenário: Avaliar Dia - Submeter Formulário
    Dado que eu esteja no formulário de "Avaliação do dia do Evento"
    E tenha preenchido todos campos da avaliação
    Quando eu clicar no botão "Avaliar"
    Então o sistema deve salvar minha avaliação para o dia do evento
    E apresentar a mensagem "Avaliação realizada, obrigado!"
```

APÊNDICE F – US004 MANTER PROGRAMAÇÃO

```
#language: pt

Funcionalidade: Manter Programação
  Com o objetivo de inserir, alterar, visualziar e inativar uma Programação no sistema
  Como um "Administrador" do sistema
  Eu quero poder inserir, alterar, visualziar e inativar uma Programação dentro do sistema

  @positivo
  Cenário: Gerenciar Programação
    Dado eu esteja autenticado
    E tenha permissão de "Administrador"
    Quando clicar no botão "Gerenciar Programação" no menu principal
    Então o sistema deve me redirecionar para página de "Gerenciar Programação"
    E deve ser apresentada a lista das Programações cadastradas

  Regras de Apresentação:
  - Serão exibidas as Programações agrupadas por edições em forma de lista
  - Serão exibidas apenas edições que possuem programações vinculadas
  - Serão ordenadas por data de cadastro do maior para o menor

  @positivo
  Cenário: Cadastrar Programação
    Dado que eu tenha realizado o cenário "Gerenciar Programação"
    Quando eu clicar no botão "Cadastrar Programação"
    Então o sistema deve me apresentar o formulário de "Cadastro de Programação"

  @positivo
  Cenário: Cadastrar Programação - Salvar Programação
    Dado que eu tenha realizado o cenário "Cadastrar Programação"
    E tenha preenchido todos os campos do Programação corretamente
    Quando eu clicar no botão "Salvar"
    Então o sistema deve me redirecionar para página de "Gerenciar Programação"
    E apresentar a mensagem "Programação cadastrada com sucesso!"

  @positivo
  Cenário: Editar Programação
    Dado que eu tenha realizado o cenário "Gerenciar Programação"
    Quando eu clicar no botão "Editar Programação" de algum Programação
    Então o sistema deve me apresentar o formulário de "Editar Programação"

  @positivo
  Cenário: Editar Programação - Salvar Programação
    Dado que eu tenha realizado o cenário "Editar Programação"
    E tenha preenchido todos os campos da Programação corretamente
    Quando eu clicar no botão "Salvar"
    Então o sistema deve me redirecionar para página de "Gerenciar Programação"
    E apresentar a mensagem "Programação atualizada com sucesso!"

  @positivo
  Cenário: Deletar Programação
    Dado que eu tenha realizado o cenário "Gerenciar Programação"
    Quando eu clicar no botão (icone delete) "Deletar" de alguma Programação
    Então o sistema deve apresentar uma caixa de dialogo com a mensagem "Tem certeza que deseja
deletar a programação?"
    E os botões "Cancelar" e "Confirmar"
    E caso o usuário clique no botão "Confirmar" o sistema deve deletar a programação
    E apresentar a mensagem "Programação deletada com sucesso!"
```

APÊNDICE G – US005 MANTER FORMULÁRIO DE AVALIAÇÃO

```
#language: pt

Funcionalidade: Manter Formulário de Avaliação de Eventos
  Com o objetivo de receber avaliações dos dias dos Eventos
  Como um "Administrador"
  Eu quero poder Inserir, Visualizar e Remover formulários de avaliação de eventos

  @positivo
  Cenário: Visualizar Formulário de Avaliação
    Dado que eu tenha realizado o cenário "Gerenciar Programação (US004)"
    Quando eu clicar no botão (icone feedback) "Formulário de Avaliação"
    Então o sisteam deve me redirecionar para a "Página de Formulário de Avaliação"
    E apresentar a tabela de formulários de acordo com o exemplo

  Exemplos:
  | Título | Cadastrado Por | Qtd. Questões | Ações |
  | Formulário | Gabriel Siqueira | (icone cloud_download) Exportar (icone edit) Editar (icone
delete) Deletar |

  @positivo
  Cenário: Cadastrar Formulário de Avaliação
    Dado que eu tenha realizado o cenário "Visualizar Formulário de Avaliação"
    Quando eu clicar no botão "Cadastrar Formulário de Avaliação"
    Então o sistema deve me apresentar o formulário de "Cadastro de Formulário de Avaliação"

  @positivo
  Cenário: Cadastrar Formulário de Avaliação - Salvar Formulário de Avaliação
    Dado que eu tenha realizado o cenário "Cadastrar Formulário de Avaliação"
    E tenha preenchido todos os campos do Formulário de Avaliação corretamente
    Quando eu clicar no botão "Salvar"
    Então o sistema deve me redirecionar para página de "Gerenciar Formulário de Avaliação"
    E apresentar a mensagem "Formulário de Avaliação cadastrado com sucesso!"

  @positivo
  Cenário: Editar Formulário de Avaliação
    Dado que eu tenha realizado o cenário "Gerenciar Formulário de Avaliação"
    Quando eu clicar no botão "Editar Formulário de Avaliação" de algum Formulário de Avaliação
    Então o sistema deve me apresentar o formulário de "Editar Formulário de Avaliação"

  @positivo
  Cenário: Editar Formulário de Avaliação - Salvar Formulário de Avaliação
    Dado que eu tenha realizado o cenário "Editar Formulário de Avaliação"
    E tenha preenchido todos os campos da Formulário de Avaliação corretamente
    Quando eu clicar no botão "Salvar"
    Então o sistema deve me redirecionar para página de "Gerenciar Formulário de Avaliação"
    E apresentar a mensagem "Formulário de Avaliação atualizado com sucesso!"

  @positivo
  Cenário: Deletar Formulário de Avaliação
    Dado que eu tenha realizado o cenário "Gerenciar Formulário de Avaliação"
    Quando eu clicar no botão (icone delete) "Deletar" de alguma Formulário de Avaliação
    Então o sistema deve apresentar uma caixa de dialogo com a mensagem "Tem certeza que deseja
deletar a programação?"
    E os botões "Cancelar" e "Confirmar"
    E caso o usuário clique no botão "Confirmar" o sistema deve deletar a programação
    E apresentar a mensagem "Formulário de Avaliação deletado com sucesso!"
```

APÊNDICE H – US006 AUTORIZAR CHECK-IN

```
#language: pt

Funcionalidade: Autorizar Check-in
  Com o objetivo de poder confirmar a presença de um usuário em um evento
  Como um "Auxiliar" ou "Administrador" do evento
  Eu quero poder confirmar a presença de um usuário em um evento

  Regras da autorização do check-in:
  - O check-in do dia de um evento só pode ser realizado uma vez por usuário no dia do evento

  @positivo
  Cenário: Autorizar Check-in
    Dado que esteja autenticado e tenha permissão de "Auxiliar" ou "Administrador" do evento
    Quando eu clicar no botão "Autorizar Check-in" no menu principal
    Então o sistema deve abrir a câmera do meu dispositivo para que eu possa ler um QRCode

  @negativo
  Cenário: Autorizar Check-in - Dispositivo sem camera
    Dado que eu esteja autenticado e tenha permissão de "Auxiliar" ou "Administrador" do evento
    Quando eu clicar no botão "Autorizar Check-in" no menu principal
    Então o sistema deve apresentar a mensagem "Ops! A camera não está disponível."

  @positivo
  Cenário: Autorizar Check-in - Ler QRCode
    Dado que eu tenha realizado o cenário "Autorizar Check-in"
    Quando a camera do dispositivo ler um QRCode com o Token do check-in
    Então o sistema deve buscar as informações do check-in
    E apresentar na tela as seguintes informações:
  - Foto do Usuário (Se disponível)
  - Nome do Usuário
  - Edição do sites
  - Data do Check-in

  @positivo
  Cenário: Autorizar Check-in - Confirmar check-in
    Dado que eu tenha realizado o cenário "Autorizar Check-in - Ler QRCode"
    Quando eu clicar no botao "Confirmar"
    Então o sistema deve apresentar a mensagem "Check-in confirmado com sucesso."
    E em tempo real no dispositivo do usuário a tela do QRCode deve fechar
    E apresentar a mensagem "Check-in confirmado com sucesso, bom evento!"

  @negativo
  Cenário: Autorizar Check-in - Programação não disponível para check in
    Dado que eu tenha realizado o cenário "Autorizar Check-in"
    Quando a camera do dispositivo ler um QRCode com o Token do check-in
    E o check-in do evento não tiver atendido as condições para realizar check-in
    Então o sistema deve apresentar a mensagem "Check-in fora de data."

  @negativo
  Cenário: Autorizar Check-in - Check-in já realizado
    Dado que eu tenha realizado o cenário "Autorizar Check-in"
    Quando a camera do dispositivo ler um QRCode com o Token do check-in
    E o check-in do usuário já tiver sido confirmado
    Então o sistema deve apresentar a mensagem "Check-in já realizado."

  @negativo
  Cenário: Autorizar Check-in - Token expirado
    Dado que eu tenha realizado o cenário "Autorizar Check-in"
    Quando a camera do dispositivo ler um QRCode com o Token do check-in
    E o token do check-in do usuário estiver expirado
    Então o sistema deve apresentar a mensagem "Check-in expirado."
```

APÊNDICE I – US007 REALIZAR SORTEIO

```
#language: pt

Funcionalidade: Realizar Sorteio
    Com o objetivo de sortear um premio durante meus eventos
    Como um "Auxiliar" ou "Administrador" do evento
    Eu quero que o sistema sorteie um usuário para mim

    Regras o sorteio:
     - Só será possível realizar sorteio de eventos que estão acontecendo e possuem no mínimo 1
usuário confirmado
     - Um usuário só poderar ser sorteado uma vez por cada check-in

    @positivo
    Cenário: Sortear Usuário
        Dado que eu esteja autenticado e tenha permissão de "Auxiliar" ou "Administrador" do
evento
        Quando clicar no botão "Realizar Sorteio" no menu principal
        Então o sistema deve solicitar que eu escolha o evento
        E me redirecionar para página de "Sorteio do Evento"

    @positivo
    Cenário: Sortear
        Dado que eu tenha realizado o cenário "Sortear Usuário"
        Quando eu clicar no botão "Sortear"
        Então o sistema deve selecionar um usuário aleatoriamente
        E apresentar suas informações
         - Nome Completo
         - Telefone
         - E-mail
        E notificar o dispositivo do usuário caso ele esteja com o sistema aberto com a seguinte
mensagem "Parabéns! Você foi sorteado, procure os organizadores do evento para retirar seu
prêmio."
        E enviar um e-mail informando que ele foi sorteado no seguinte modelo:

        """
            Parabéns!!! <nome do usuário>,

            Você foi sorteado por estar participando do evento <nome do evento>,
            Caso esteja no local do evento, procure os organizadores do evento para retirar seu
prêmio.

            Caso não esteja mais no local do evento infelizmente você não poderá retirar o prêmio
pois,
            os prêmios são apenas para os participantes que ficaram até o final.

            Atenciosamente,
            Uni Eventos
        """
```

APÊNDICE J – US008 EDITAR USUÁRIO

```
#language: pt

Funcionalidade: Editar Usuário
    Com o objetivo de editar um usuário e poder atribuir permissões para os usuários
    Como um "Administrador"
    Eu quero poder editar as informações do usuário e poder atribuir/remover permissões para os
mesmos

    @positivo
    Cenário: Listar Usuário
        Dado que eu esteja autenticado e possua permissão de "Administrador"
        Quando clicar no botão "Gerenciar Usuários" no menu principal
        Então o sistema deve me redirecionar para página de "Gerenciar Usuários"
        E deve ser apresentada a lista de todos os usuários cadastrados paginados

    @positivo
    Cenário: Listar Usuário - Pesquisar
        Dado que eu tenha realizado o cenário "Listar Usuário"
        Quando eu digitar algum texto no campo de pesquisa da página
        Então o sistema deve filtar a lista Usuários
        E mostrar os usuários correspondentes

    @positivo
    Cenário: Editar Usuário
        Dado que eu tenha realizado o cenário "Listar Usuário"
        Quando eu clicar no botão "Editar Usuário" no usuário que desejo editar
        Então o sistema deve apresentar o "Formulário de Edição de Usuário" com as informações do
usuário

    Regras e Campos do formulário:
        | Campos | Regas |
        | Nome Completo  | Obrigatório, Máximo 50 caracteres |
        | Imagem do Perfil | Opicional |
        | Sexo | Obrigatório |
        | Data de Nascimento | Obrigatório, Validação de data |
        | E-mail | Obrigatório, Validação de e-mail, Deve ser único |
        | Senha | Obrigatório, Confirmada, Mínimo 6 dígitos |
        | Vínculo | Obrigatório, (Aluno, Servidor, Comunidade) |
        | Matrícula | Obrigatório se o vínculo for diferente de Comunidade |
        | Papel | Opicional (Administrador, Auxiliar, Nenhum) |

    @positivo
    Cenário: Atualizar Usuário
        Dado que eu tenha realizado o cenário "Editar Usuário"
        E tenha preenchido todos os campos do Usuário corretamente
        Quando eu clicar no botão "Salvar"
        Então o sistema deve me redirecionar para página de "Gerenciar Usuários"
        E apresentar a mensagem "Usuário atualizado com sucesso!"
```

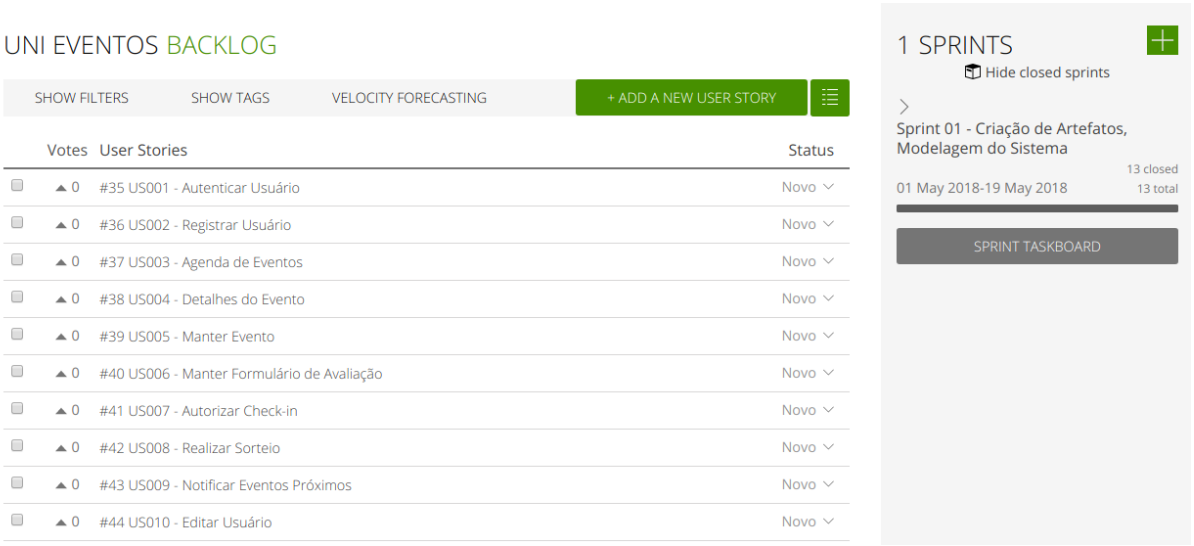
APÊNDICE K – US009 NOTIFICAR PROGRAMAÇÃO

```
#language: pt

Funcionalidade: Notificar Programação
    Com o objetivo de notificar por e-mail os usuários cadastrados no sistema sobre as
programações que irão acontecer
    Como um usuário
    Eu quero que o sistema notifique-me por e-mail caso tenha
    alguma programação acontecendo no dia

    @positivo
    Cenário: Notificar Programação - Resumo do dia
        Dado que eu esteja cadastrado no sistema
        E tenha autorizado o sistema a me enviar e-mail sobre a programação na minha tela de
perfil
        Quando for 08h da manhã de todos dias
        E existir alguma programação cadastrada para o dia
        Então o sistema deve enviar um e-mail notificando-me com os detalhes da programção do dia

    """
        Resumo do dia
        ===============
        Olá <nome do usuário>,

        Hoje temos a seguinte programação acontecendo no evento SITES da instituição:

        [detalhe da programação cadastrada no sistema]
        ----------------------------------------------------------------------------------------------
----------------------

        Fique atento para não chegar atrasado,
        Atenciosamente,
        SITES
    """
```

APÊNDICE L – US010 VISUALIZAR PARTICIPANTES

```
#language: pt

Funcionalidade: Visualizar Participantes
  Com o objetivo de visualizar os participantes de uma Programação no sistema
  Como um "Administrador" do sistema
  Eu quero poder visualziar a lista de participantes de uma Programação dentro do sistema

  @positivo
  Cenário: Visualizar Participantes
    Dado que eu tenha realizado o cenário "Visualizar Programação (US004)"
    Quando eu clicar no botão (icone people) "Participantes"
    Então o sisteam deve me redirecionar para a "Página de Participantes"
    E apresentar a tabela de participantes de acordo com o exemplo

  Exemplos:
  | Matrícula | Nome Completo | Vínculo | Sexo | Avaliou o Programação? | Horário do Check-in |
Confirmado por |
  | 1410732 | Gabriel Siqueira | Aluno | Masculino | Sim | 07/07/2018 10:10:10 | Administrador |

  @positivo
  Cenário: Visualizar Participantes - Ordenar dados
    Dado que eu tenha realizado o cenário "Visualizar Participantes"
    Quando eu clicar uma vez em algum cabeçalho da tabela
    Então o sisteam deve ordenar os dados da tabela de acordo com as regras

  Regras:
  - 1º clique deve ordernar crescente
  - 2º clique deve ordernar decrescente
  - 3º clique deve remover a ordenação (deixar a padrão)
  - ordenação padrão por ID crescente


  @positivo
  Cenário: Visualizar Participantes - Pesquisar
    Dado que eu tenha realizado o cenário "Visualizar Participantes"
    Quando eu digitar algum texto no campo "Pesquisar"
    Então o sisteam deve realizar a pesquisa nos campos

  Campos:
  - Matrícula
  - Nome
  - Vinculo
  - Sexo
  - Horário do Check-in
  - Confirmado por

  @positivo
  Cenário: Visualizar Participantes - Exportar Dados
    Dado que eu tenha realizado o cenário "Visualizar Participantes"
    Quando eu clicar no botão (icone cloud_download) "Exportar para Excel"
    Então o sistema deve exportar todos os participantes para o formato XLS

  Exemplos:
  | Matrícula | Nome Completo | Vínculo | Sexo | Horário do Check-in | Confirmado por |
  | 1410732 | Gabriel Siqueira | Aluno | Masculino | 07/07/2018 10:10:10 | Administrador |
```

APÊNDICE M – TAIGA PRODUTO BACKLOG



UNI EVENTOS BACKLOG

SHOW FILTERS | SHOW TAGS | VELOCITY FORECASTING | + ADD A NEW USER STORY

Votes	User Stories	Status
▲ 0	#35 US001 - Autenticar Usuário	Novo
▲ 0	#36 US002 - Registrar Usuário	Novo
▲ 0	#37 US003 - Agenda de Eventos	Novo
▲ 0	#38 US004 - Detalhes do Evento	Novo
▲ 0	#39 US005 - Manter Evento	Novo
▲ 0	#40 US006 - Manter Formulário de Avaliação	Novo
▲ 0	#41 US007 - Autorizar Check-in	Novo
▲ 0	#42 US008 - Realizar Sorteio	Novo
▲ 0	#43 US009 - Notificar Eventos Próximos	Novo
▲ 0	#44 US010 - Editar Usuário	Novo

1 SPRINTS

Hide closed sprints

Sprint 01 - Criação de Artefatos, Modelagem do Sistema

13 closed

01 May 2018-19 May 2018 13 total

SPRINT TASKBOARD

APÊNDICE N – TERMO DE ACEITE DE ENTREGA

TERMO DE ACEITE DA ENTREGA

Sistema Para Controle Do Evento Sites

VERSÃO: 0.1

Anápolis – GO

2018

1. OBJETIVO DESTE DOCUMENTO

Este documento formaliza o aceite da entrega considerando-a em conformidade com os requisitos e os critérios de aceitação definidos nas histórias de usuário identificadas com os *stakeholders*.

2. ENTREGA

O sistema deve atender as histórias de usuário abaixo e seus critérios de aceitação. As histórias foram priorizadas utilizando a técnica "MOSCOW", na qual será utilizada como premissa para realização da entrega dos requisitos de forma que o sistema atenda obrigatoriamente todas as histórias e cenários que foram priorizados como sendo "essencial".

1. US001 - Autenticar Usuário;
2. US002 - Registrar Usuário;
3. US003 - Programação;
4. US004 – Manter Programação;
5. US005 - Manter Formulário de Avaliação;
6. US006 – Autorizar Check-in;
7. US007 – Realizar Sorteio;
8. US008 – Editar Usuário;
9. US009 – Notificar Programação;
10. US010 – Visualizar Participantes;

3. REQUISITOS FUNCIONAIS

ACEITE DA ENTREGA		
Os participantes abaixo atestam o cumprimento dos requisitos e dos critérios de aceitação da entrega.		
Participante	**Assinatura**	**Data**
ADRIELLE BEZE PEIXOTO		
GABRIEL LEANDRO J. SIQUEIRA		
MILLYS FABRIELLE A. CARVALHAES		
PEDRO VICTOR DE OLIVEIRA E SILVA		

APÊNDICE O – TELAS DO SISTEMA