SISEVENT: uma proposta de sistema para apoio à gestão de eventos em instituições públicas de ensino

Kaio Mitsuharu Lino Aida[1], Vinícius Espíndola Ferreira[1], Awdren de Lima Fontão (*orientador*)[1]

Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

{Aida, Kaio} kaiomudkt@gmail.com, awdren.fontao@ufms.brf

{Espindola, Vinicius} vinicius_espindola@ufms.br, awdren.fontao@ufms.br

Resumo. As restrições financeiras para investir em software pago de gestão de eventos e a não utilização de todos os seus recursos faz com que possa existir uma ineficiência nos processos da gestão de eventos realizados em instituições públicas de ensino. A grande quantidade de participantes em eventos torna complexo o processo de gerenciamento e controle, antes, durante e depois da realização, sendo trabalhoso e cansativo. Desta maneira, podendo ocasionar a ocorrência de erros humanos por trabalho repetitivo. Por meio deste sistema web, será implementado uma solução para sistematizar os processos, automatizando etapas e assim melhorando o fluxo e andamento dos eventos, principalmente na geração de certificados.

Palavras-chave: Credenciamento, Gestão de evento, Instituições de ensino, software.

Abstract. The low budget for investing in event management software and the underutilization of all its features can lead to inefficiencies in the processes of event management carried out in educational institutions. The large number of participants in events complicates the management and control process throughout all stages, making it laborious and tiring. Consequently, this can result in human errors due to repetitive work. Through this web-based system, a solution will be implemented to systematize processes, automate steps, and thereby improve the flow and management of events, especially in terms of certificate generation.

Keywords: *Check-in,* Credenciamento, Event management, Educational institution, software.

Agradecimentos

Agradecemos a Deus e aos nossos familiares que nos apoiaram todo o tempo, desde o começo da graduação até a finalização deste projeto.

Agradecemos aos professores Renato Ishii e Sérgio Carvalho de Araújo, pelas contribuições. Agradecemos em especial nosso professor orientador Awdren de Lima Fontão, pela orientação, paciência e compreensão. Aos membros do LEDES e faculdade em todos seus âmbitos.

Agradecemos ao nosso grupo de amigos Diego Bulhões, Carlos Pereira, Mateus Ragazzi, Lucas Sandim que acompanharam o trajeto de concepção e desenvolvimento deste projeto.

Eu, Kaio Mitsuharu Lino Aida, particularmente agradeço ao meu companheiro de projeto Vinícius Espíndola Ferreira pela paciência e profissionalismo e também agradeço aos incentivos da minha família e da minha namorada Mariana Vidotti.

Lista de abreviaturas e siglas

CSV	*Comma-separated values*
DB	Banco de Dados (do inglês, *Data Base*)
SQL	*Standard Query Language*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
HTTP	*Hypertext Transfer Protocol*
FDD	*Feature Driven Development*
PET	Programa de Educação Tutorial
CRUD	*Create, Read, Update, Delete*
SISCERT	Sistema de Certificados
SISCAD	Sistema Acadêmico
SIGPROJ	Sistema de Informação e Gestão de Projetos
LGPD	Lei geral de proteção de dados
PROECE	Pró-Reitoria de Extensão, Cultura e Esporte
CI	*Continuous Integration*
CD	*Continuous Delivery*
REST	*Representational State Transfer*
ACID	*atomicity, consistency, isolation, durability*
API	*Application Programming Interface*
ORM	*Object Relational Mapper*

Lista de tabelas

Lista de figuras

Sumário

1 Introdução

Está monografia tem como objetivo apresentar o projeto SISEVENT, que foi uma proposta de sistema web para apoiar a gestão de eventos em instituições de ensino, com ênfase em instituições públicas. Desde sua necessidade até sua solução tende a se especializar em ferramentas que agregam para as particularidades deste contexto.

Ao decorrer do texto será apresentado desde a concepção, processo de desenvolvimento e decisões até o planejamento futuro. Tudo para que seja possível viabilizar sua construção em tempo hábil, sendo necessário tomar diversas decisões tanto na parte de negócio e requisitos quanto na parte técnica, para que seu desenvolvimento comece com uma base sólida, mas que permita possíveis mudanças e adaptações para que o sistema vingue e tenha adesão real da comunidade e uma longa vida útil.

1.1 Contexto

Em 2022, a Universidade Federal do Mato Grosso do Sul (UFMS) contava com 24,8 mil estudantes e 3,2 mil servidores, distribuídos em 22 cidades do estado do Mato Grosso do Sul[1]. sendo eles Aquidauana, Chapadão do Sul, Campo Grande, Corumbá, Coxim, Naviraí, Nova Andradina, Três Lagoas, Paranaíba e Ponta Porã [UFMS, 2023].

Os eventos realizados dentro da UFMS abrangem uma ampla diversidade de atividades acadêmicas, culturais e científicas. Esses eventos são de fundamental importância, pois proporcionam oportunidades de aprendizado, interação e disseminação de conhecimento para a comunidade universitária e a sociedade.

1.2 Motivação e justificativa

Com o grande volume da comunidade interna da UFMS e organizando eventos, surge o problema de gestão de eventos, que pode incluir lidar com formas manuais de controle de eventos. Como, por exemplo, a relatório de certificado, criação e gerenciamento da equipe organizadora e a dificuldade de realizar eventos pagos.

Além de particularidades que um evento científico possui, por exemplo, submissões e avaliações de artigos de pesquisa, apresentação e premiação para esses trabalhos. Outra motivação é a ausência de um local atualizado e centralizado na UFMS

que permita aos professores divulgar e alterar facilmente informações sobre os eventos diretamente na plataforma.

1.3 Contexto do projeto

Nos eventos promovidos pelos grupos PET, identificou-se a demanda por um sistema destinado ao controle de eventos e ao registro em larga escala de presenças, visando a eficiente geração de relatório de certificados. Diante desse cenário, o PET Sistemas empreendeu uma iniciativa que resultou na implementação de um sistema de registro de presenças baseado em QR Code. Contudo, constatou-se que essa solução carecia de flexibilidade, sobretudo no que se refere à geração em massa de relatórios de certificados.

1.4 Descrição do problema

1.4.1 Plataformas pagas

As plataformas de gestão de eventos disponíveis no mercado, como o Sympla[1] e o Even3[2], foram desenvolvidas por empresas privadas visando o lucro, com venda de ingressos, como para shows musicais, espetáculos de teatro e eventos esportivos. Com o principal objetivo do credenciamento nessas plataformas é validar o pagamento, e não para emissão de certificado, resulta com que as particularidades e necessidades das instituições públicas de ensino podem não ser totalmente supridas, tanto de funcionalidades quanto de forma financeira, por exemplo, em levar em consideração a carga horária e cálculo de nota para emissão de relatório de certificado sem custo.

1.4.2 Registro de presença

Com base em conversas com potenciais usuários a abordagem convencional para registrar a presença é a coleta de assinaturas de cada participante em uma lista de presença em papel. Outra prática comum envolve a organização do evento utilizando um sistema pago, ainda que sua versão gratuita geralmente seja limitada, para controlar a entrada de participantes em cada sessão.

[1] Disponivel em https://www.sympla.com.br/
[2] Disponivel em https://www.even3.com.br/

1.4.3 Geração de relatório de certificados em massa

A utilização de listas de presença em papel ou em plataformas em suas versões limitadas, faz com que a geração de relatórios de certificados se torne trabalhosa. O SISCERT[3] oferece uma ferramenta para emissão em massa de certificados, exigindo apenas um arquivo na extensão CSV (*Comma-separated value,* do inglês, arquivo separado por vírgula) com o formato especificado. Para gerar os certificados sem utilizá-lo é necessário manipular grandes planilhas de excel para ajustar os dados no formato que o SISCERT espera. Tornando o trabalho repetitivo e aumentando as chances de erro humano e o tempo de espera na emissão dos certificados.

1.4.4 Gestão da equipe organizadora

Com base nas conversas iniciais, foi descoberta a relevância de auxiliar na gestão da equipe organizadora, assim como a dificuldade de se comunicar com os alunos para convidá-los a compor a equipe organizadora de forma voluntária. E em seguida organizar a equipe de forma em que todas as tarefas sejam distribuídas sem sobrecarregar os membros e respeitando a suas respectivas disponibilidades.

Foi constatado por meio das entrevistas que quando o credenciamento é realizado de forma manual, é comum que essa organização seja feita em planilhas e por meio de mensagens de textos em grupos de aplicativo de mensagem, como por exemplo Whatsapp e Telegram. Cada membro da equipe recebe suas tarefas, enquanto o responsável pelo evento ou pela equipe acompanha o andamento das atividades, verificando de que estão sendo executadas de acordo com o esperado.

1.4.5 Divulgação de evento

A UFMS[4] dispõe de uma área centralizada e dedicada onde é possível obter informações sobre os eventos, onde é gerenciada por um órgão administrativo, porém somente servidores deste setor podem atualizar informações dos eventos, gerando burocracia na atualização das informações em tempo hábil que dificulta a adesão da comunidade. O que resulta em uma lacuna na divulgação de eventos dentro da UFMS.

[3] Sistema usado na UFMS para emissão, download e visualização dos certificados, está disponivel no link https://certificados.ufms.br/

[4] Disponível em https://www.ufms.br/eventos/

Essa falta de acesso fácil às informações atualizadas e detalhes da programação dos eventos pode levar a uma desconexão com a comunidade acadêmica e as oportunidades.

Apesar de normalmente as instituições terem órgãos administrativos, cada instituição pode decidir o quanto de autonomia pode dar aos professores para criar e publicar eventos, como por exemplo na UFMS, é possível ter uma regra com que o evento possa alterar o status da visibilidade para pública, mas antes deve estar aprovado no SIGPROJ, ou somente o servidor administrativo pode alterar manualmente a visibilidade. Mas em outras instituições, pode ocorrer que o próprio professor, que tem responsabilidade sobre suas ações, possa ter autonomia para publicar eventos. Desta forma o sistema é capaz de se adaptar às necessidades de cada instituição.

1.4.6 Avaliação e melhoria dos eventos

Os eventos frequentemente solicitam apenas as informações essenciais para a emissão dos certificados. No entanto, a falta de coleta e armazenamento de dados dos eventos, especialmente de forma estruturada para permitir análises posteriores, representa uma limitação significativa. A falta desses dados de forma estruturada dificulta visualizar se os objetivos do evento foram alcançados e em caso de falha, apoiar o dono do evento a entender onde estão os possíveis pontos de melhoria.

Desta forma, seria viável gerar avaliações dos eventos, atividades, sessões, tarefas e demais, com sugestões de melhorias, exibindo de forma gráfica os pontos fracos e fortes.Comunicação com os participantes

A dificuldade do participante de entrar em contato com a equipe organizadora do evento pode trazer diversos problemas, por exemplo a necessidade de trazer itens específicos ou mudança de local para realização da atividade. Pois a comunicação efetiva é essencial para garantir que todas as partes envolvidas estejam alinhadas.

Quando há uma lacuna na comunicação, torna-se difícil transmitir mudanças de planos, lembrar aspectos importantes e enviar e-mails em massa para grupos específicos de participantes, por exemplo, inscritos em uma atividade específica, presentes na sessão ou aptos a receber certificados.

1.4.7 Reserva de locais e cautela de materiais

Dentro das instituições existem diversos recursos de materiais e locais, que podem ser reservados para serem utilizados durante o evento. Exemplo de recursos de materiais são microfone, caixa de som e projetor. Exemplo de recursos de locais são laboratório de informática, auditórios e salas de aulas.

A concorrência para reserva de recursos é normalmente feita de forma descentralizada, no qual cada unidade da instituição tem seu próprio controle. Desta forma, podem existir recursos que acabam sendo subutilizados por desconhecimento de sua existência e até por simplesmente inacessibilidade ao responsável do recurso para poder solicitar a reserva. E até podendo haver possíveis alocações simultâneas desses recursos em mais de um evento, gerando conflito.

1.5 Objetivo

Portanto, o objetivo é a criação de um sistema web para facilitar a gestão de eventos e exportação do relatório de certificados aceito pelo SISCERT. O sistema de gestão de eventos tem como primeiro foco o registro de presença dos participantes auxiliando na geração de certificados, buscando fazer o relacionamento e controle de forma dinâmica e simples. O segundo objetivo é auxiliar no processo de estruturação do evento e sua equipe organizadora auxiliando na montagem e distribuição de responsabilidades, com isso também auxiliando na divulgação dos eventos para aumentar o engajamento da comunidade acadêmica.

2 Referencial Teórico

2.1 Conceito

2.1.1 Evento

É uma ocorrência planejada e organizada com um propósito específico. Eventos podem ocorrer em uma ampla gama de contextos, desde eventos sociais e culturais até eventos empresariais e acadêmicos [Ivo, 2014]. Para o contexto deste projeto refere-se a reunião de pessoas em um local específico ou virtualmente para participar de atividades, experiências ou celebrações. Eventos podem ter diferentes formatos e objetivos, e podem variar em escala e complexidade. Alguns exemplos comuns de eventos incluem conferências, seminários e até cursos.

2.1.2 Atividade

Como conceito do sistema, uma atividade refere-se a uma ação específica que ocorre como parte do programa ou agenda do evento. As atividades podem variar de acordo com o tipo do evento, objetivos e público-alvo. Elas são planejadas e organizadas para um propósito e gerar uma experiência específica aos participantes assim agregando valor e promovendo os objetivos do evento. As atividades podem incluir palestras, workshops, cursos e painéis de discussão.

2.1.3 Sessão

Como conceito do sistema, a atividade pode ser separada em diversas sessões, sendo que cada sessão poderá ter seu início e fim ser síncrona ou assíncrona, presencial ou remota, sendo flexível para atender diferentes necessidades, assim viabilizando a realização da mesma. Como por exemplo, realização de uma minicurso com quatro sessões, sendo as duas primeiras sessões presenciais e síncronas, e as duas segundas remotas e assíncrona, trazendo flexibilidade para a execução da atividade.

2.1.4 Registro de presença

Como conceito do sistema, o registro de presença, comumente denominado credenciamento ou *check-in*, constitui o procedimento autorizado e controlado que atesta o êxito da participação de um indivíduo em uma sessão específica [Hu, 2020].

2.1.5 *Framework*

Framework é uma estrutura de trabalho que fornece uma base para facilitar o desenvolvimento de aplicativos. Segundo Ramirez *et al*. [2019], Ele oferece um conjunto de ferramentas, bibliotecas (subseção 3.8), componentes e padrões de projeto que ajudam os desenvolvedores a criar aplicativos de forma mais eficiente, padronizada e escalável. Os *frameworks* são projetados para resolver problemas comuns enfrentados pelos desenvolvedores, oferecendo abstrações de alto nível e fornecendo funcionalidades básicas que podem ser reutilizadas em diferentes projetos.

2.1.6 Desenvolvimento orientado por funcionalidade (FDD)

O FDD (Feature Driven Development, do inglês Desenvolvimento Orientado por Funcionalidades), é uma metodologia ágil de desenvolvimento de software que se concentra na entrega incremental de funcionalidades por prioridade e na colaboração eficiente entre membros da equipe [Coad, 1999]. A motivação da escolha do FDD é devido ao seu enfoque na funcionalidade e à simplicidade na elaboração da lista de requisitos e funcionalidades do sistema. O FDD foi aplicado de forma parcial, tendo os requisitos mapeados e inseridos em uma planilha Excel, com base nas entrevistas feitas durante o levantamento de requisitos e os classificando com suas prioridades.

2.1.7 Bibliotecas

Bibliotecas consistem em uma coleção de código pré-escrito, classes, procedimentos, scripts, dados de configuração, por exemplo [Mili, 1998]. Um desenvolvedor pode adicionar manualmente uma biblioteca de software para automatizar um processo sem escrever código para ele. Por exemplo, no desenvolvimento de uma aplicação, o desenvolvedor pode adicionar bibliotecas ao projeto para evitar a escrita de funções que realizam conversão de data e fuso horário. Em outras palavras, podem ser reaproveitados trechos de códigos e funcionalidades já desenvolvidas.

De acordo com *Batory et al*. [1993], bibliotecas de software são coleções de código conhecidas por impulsionar programadores na produtividade e reduzir o tempo e preço

para o desenvolvimento de software. Para o desenvolvimento deste trabalho, bibliotecas foram utilizadas, tanto na parte do *frontend* quanto no *backend* da aplicação.

2.1.8 Pirâmide de testes

A Pirâmide de Testes é um conceito utilizado no desenvolvimento de software para orientar a estratégia de testes, enfatizando a distribuição adequada dos diferentes tipos de testes [Contan, 2018]. Ela sugere que os testes devem ser organizados em diferentes camadas, representando diferentes níveis de abstração e escopo [Contan, 2018], com o objetivo de obter uma cobertura de acordo com os padrões definidos no projeto, resultando na diminuição de recursos, assim estreitando a pirâmide.

Na implementação dos testes das três aplicações, inicialmente nos baseamos na estrutura da pirâmide de testes. Devido a restrições de tempo e recursos, concentramos nossos esforços principalmente na realização de testes de integração no *backend*, visando garantir a qualidade de maneira eficiente.

2.2 Ferramentas

2.2.1 Kanban

O Kanban é uma metodologia ágil para gerenciamento de projetos e fluxo de trabalho. Originado no setor de manufatura japonesa, o Kanban foi adaptado para o desenvolvimento de software e outras áreas de trabalho [Nakazawa, 2016]. Ele se baseia em um sistema visual de quadro de tarefas, cartões e limites de trabalho em progresso para ajudar as equipes a organizar, priorizar e controlar seu trabalho de forma eficiente.

A escolha dele foi feita considerando a necessidade de adaptação ao modo de trabalho e às restrições de tempo, tendo em vista o desempenho de tarefas adicionais além das atividades acadêmicas.

2.2.2 Discord

O Discord é uma plataforma de comunicação por voz, vídeo e texto voltada principalmente para comunidades online e gamers. Lançado em 2015 [Raglianti, 2022], o Discord rapidamente ganhou popularidade devido à sua facilidade de uso, recursos robustos e grande ênfase na interação social. Ele foi usado para comunicação entre as

partes do projeto e também para registrar decisões e documentar mudanças e eventuais problemas que ocorreram durante o projeto.

2.2.3 Versionamento

2.2.3.1 Sistema de Controle de Versão

Já no Sistema Distribuído de Controle de Versão (DVCSs, do inglês Distributed Version Control Systems), os usuários podem clonar todo o repositório em suas máquinas, desta maneira, se dados forem danificados em um servidor, podem ser restabelecidos (Figura 2). Exemplos de DVCSs: Git, Mercurial, Bazaar ou Darcs.

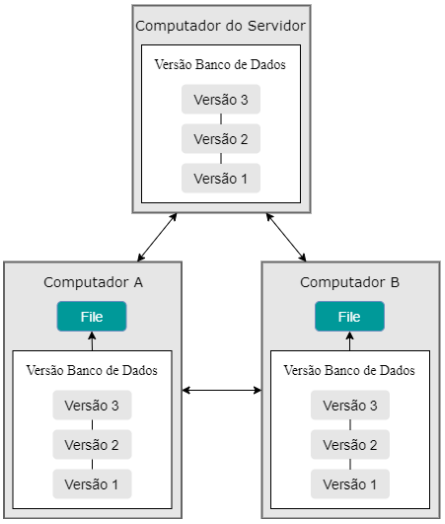

Figura 1. Controle de versionamento distribuído [Chacon *et al*., 2014].

2.2.4 Github

Utilizou-se do DVCS Git, que é rápido, distribuído, escalável e permite manter e navegar entre várias versões do desenvolvimento, contendo todos os registros de modificações. Como complemento ao Git, foi utilizado o GitHub, uma plataforma para armazenar repositórios. De maneira simplória, Git é a ferramenta onde o trabalho é realizado, localmente em seu computador, e Github um servidor público onde seu código pode ser armazenado e compartilhado [Mudholkar, 2017].

2.2.5 Laravel

Laravel é um popular *framework* de desenvolvimento de aplicativos web em PHP. Ele foi criado por Taylor Otwell em 2011 [Laravel *Development Team*, 2023], e desde então se tornou uma das escolhas mais populares para o desenvolvimento web com PHP sendo considerado elegante, eficiente e rico em ofertas de recursos. Amplamente utilizado pela comunidade e construído para um propósito geral, é de fácil aprendizado e tendo diversos conteúdos, oque poderá facilitar sua evolução e manutenção. Este foi escolhido por se encaixar no domínio utilizado. Dentro do nicho de aplicações voltadas para gestão de eventos, existem algumas empresas que utilizam o Laravel, por exemplo Eventbrite e a Grenadine Event Software.

2.2.6 Vue.js

Vue.js é um *framework* de Javascript de código aberto e progressivo para a construção de interfaces de usuário [Vue *Development Team*, 2023]. Ele foi criado por Evan You em 2014 e rapidamente ganhou popularidade devido à sua abordagem simples, flexível e fácil de aprender. ele foi criado para gerar HTML, CSS e Javascript provendo de forma declarativa baseada em componentes de se criar uma aplicação de forma a facilitar o seu desenvolvimento em projetos simples e complexos.

2.2.7 Vuetify

Vuetify é uma biblioteca de componentes de interface de usuário baseada no Vue.js que segue as diretrizes de Material Design do Google [Vuetify *Development Team*, 2023]. Ela oferece uma gama de componentes prontos para uso que podem ser facilmente personalizados para se adequarem ao design do aplicativo, como botões, barras de navegação, formulários, cartões, listas e mais.

2.2.8 Axios

O Axios é uma biblioteca Javascript de código aberto que permite fazer requisições HTTP a partir de navegadores ou de servidores que usam Node.js [Axios *Development Team*, 2023]. Ele fornece uma interface simples e intuitiva para lidar com solicitações e respostas HTTP.

3 Ferramentas relacionadas

3.1 Sympla

O Sympla[5] oferece uma solução abrangente para organizadores de eventos, permitindo que os donos dos eventos criem, promovam, vendam ingressos e gerenciem os eventos. Fornecendo funcionalidades como, exportar dados de presença em diferentes formatos. A plataforma não possui funcionalidades que um evento científico cediado em uma universidade federal necessita, por exemplo, gerenciamento de materiais e laboratórios, a emissão de relatórios de presença no formato utilizado pelo SISCERT e no plano gratuito não contempla as necessidades descritas na na subseção 1.3.

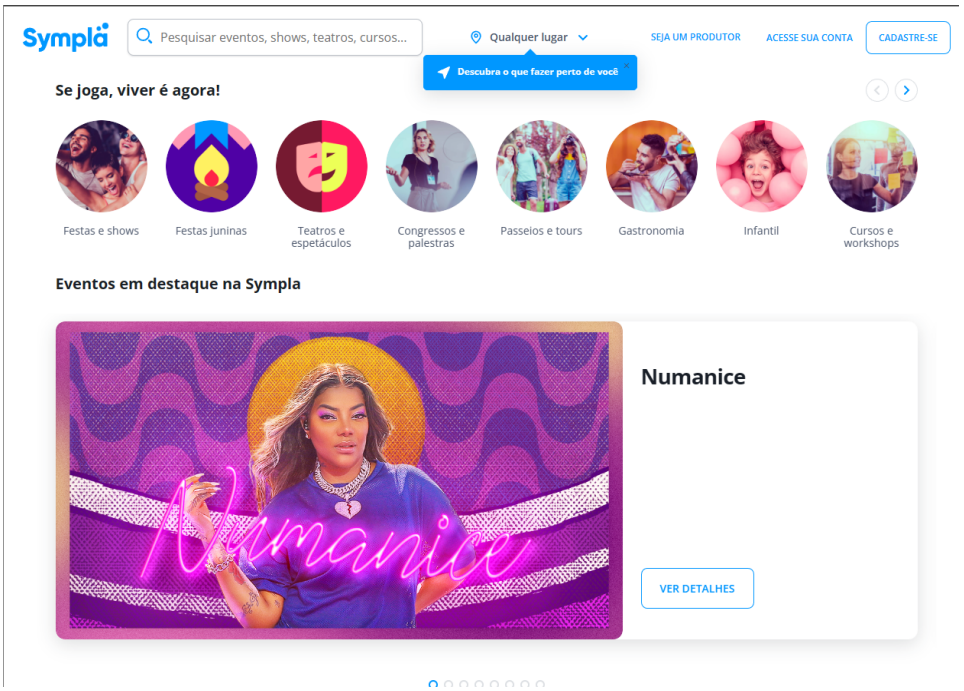

Figura 2. Plataforma Sympla

3.2 Whova

Plataforma de gestão de eventos tudo-em-um projetada para simplificar e melhorar a experiência de organizadores e participantes de eventos. O Whova[6] oferece uma gama de recursos e ferramentas para ajudar na organização, promoção e execução

[5] Disponível em https://www.sympla.com.br/
[6] Disponível em https://whova.com/

de eventos, sejam eles conferências, feiras, simpósios, reuniões corporativas, entre outros.

O Whova disponibiliza funcionalidades para gestão de eventos presenciais com em híbridos e remotos, onde conseguem criar sessões de transmissão de vídeo, apresentação das atividades e fazer gerenciamento das presenças. Porém, o Whova contempla parcialmente as necessidades por ser um software feito para vários nichos como, universitário, governamental e empresarial.

Dessa forma, as funcionalidades disponíveis, como registro de presença e geração de relatório de presença no formato compatível com a ferramenta de geração de certificados em massa do SISCERT, resolvem parcialmente os problemas previamente citados na subseção 1.3, mas ele não consegue auxiliar nas particularidades de um evento no contexto de uma instituição públicas de ensino.

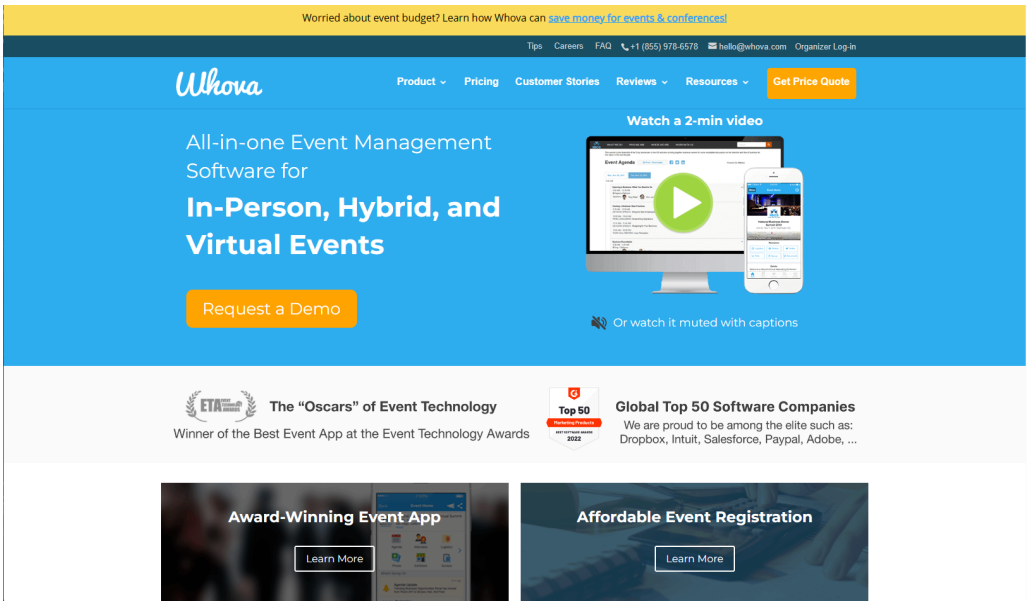

Figura 3. Plataforma Whova

4 Uma proposta de sistema para apoio à gestão de eventos em instituições públicas de ensino

4.1 Material e métodos

4.1.1 Gestão de demandas com Kanban

Como explicado na subseção 2.5, o Kanban foi utilizado para organização das tarefas de desenvolvimento em conjunto com o Discord descrito na subseção 2.6. Ele foi separado em colunas de controle para ter visão sobre como estavam o andamento das tarefas e o projeto como um todo, as colunas da tabela 1 e também ilustrado nas figuras 4 e 5.

Coluna no Kanban	Descrição
Backlog	Utilizado para requisitos do sistema que foram mapeados mas ainda não foram definidos se irão ser implementados ou passados para próximas versões do sistema com base no escopo e prioridade com o qual a funcionalidade vai impactar o projeto.
Secundário	Funcionalidades que foram mapeadas mas não irão entrar na primeira entrega do software, mas já estão mapeadas sua implementação como sugestão para futuras melhorias ou funcionalidades no projeto.
TODO	Atividades que estão refinadas e foram escolhidas para essa entrega por fazerem parte do escopo ou por serem importantes para a primeira versão do projeto

Done	Atividades que seu desenvolvimento foi finalizado, faltando somente integrar na versão final da primeira versão do projeto e revisar sua documentação.
Doing	Atividades no estágio de desenvolvimento onde são priorizadas e etiquetadas de acordo com sua categoria(funcionalidade,refatoração, documentação, testes) e seu grau de prioridade (baixa, média e alta).
Review	Após a atividade ser finalizada o outro participante do grupo olha as alterações para validar se estão conforme o refinado, podendo voltar para o estágio anterior em caso de algum problema encontrado.
Decision	Atividades sobre discussão arquitetural do sistema ou de prioridade relacionado a uma ou mais funcionalidades que foram alteradas após o refinamento ser concluído por causa de mudança de contexto.
Cancelled	Atividades que foram canceladas, podendo ser porque a funcionalidade não é mais necessária ou porque as circunstâncias em que o card foi escrito mudaram e ele não é necessário ou não faz mais sentido sua implementação no novo contexto, pois os objetivos podem ter mudado.

Tabela 1. Nome e descrição das colunas do kanban

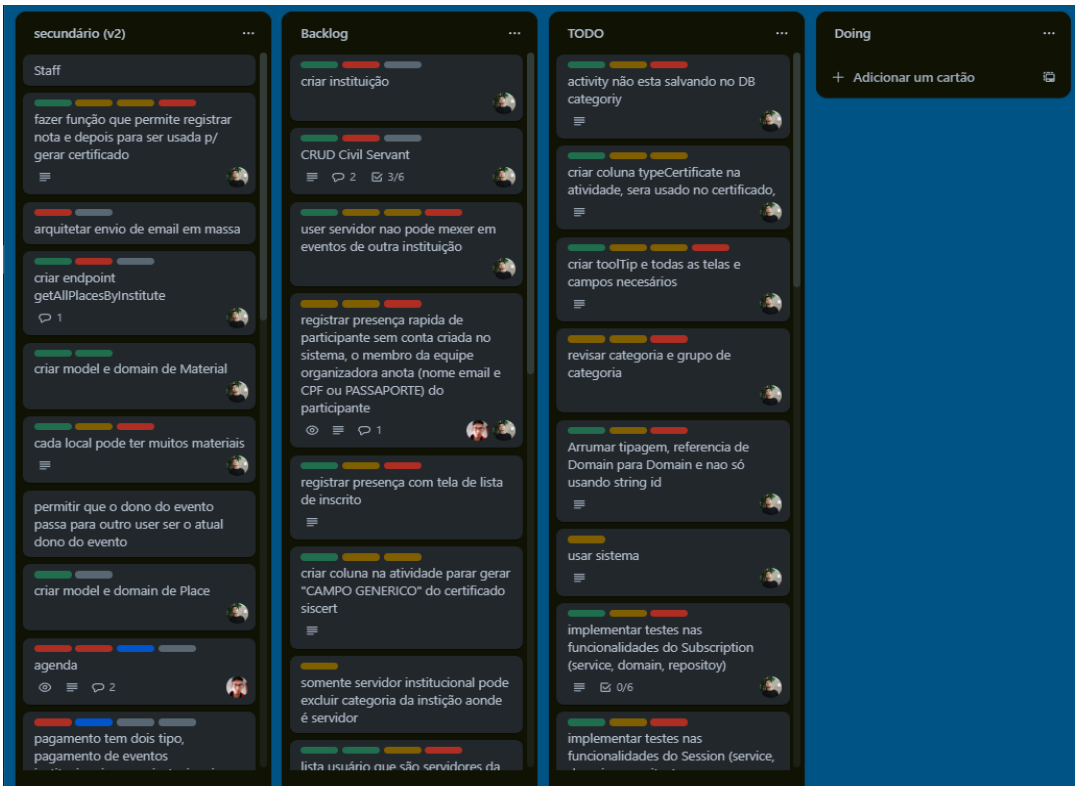

Figura 4. Kanban do projeto parte 1

Figura 5. Kanban do projeto parte 2

4.1.2 Discord

O Discord foi utilizado como plataforma de comunicação entre os membros da equipe e gerenciamento do projeto, onde maior parte das reuniões e decisões foram discutidas e documentadas, sendo utilizado em conjunto com o Kanban para controle de demandas dos pedidos de integração de código (*Pull Requests*).

4.2 Desenvolvimento incremental

4.2.1 Sistema descontinuado

O projeto foi originalmente concebido pelo PET (Programa de Educação Tutorial) Sistemas, de uma forma sucinta, assim, ocasionalmente durante seu uso, foram surgindo dificuldades tanto de atualizações quanto de usabilidade, para atender as necessidades que foram surgindo.

4.2.2 Entendendo o produto

Ao iniciar o projeto, o escopo final de entrega do produto ainda não havia sido definido totalmente, somente o objetivo de auxiliar na gestão de eventos. Apesar deste software criado pelo PET Sistemas se tornar difícil de manusear e complexo rapidamente, ele proporcionou o entendimento dos primeiros requisitos. As limitações relatadas por seus usuários foram coletadas e documentadas em textos simples, semelhante a descrições de caso de uso e as usamos para melhor compreensão do problema a ser abordado.

Os membros do projeto já tinham experiência prática na participação em organizações de eventos e tinham noção de algumas dificuldades de gestão de eventos. Que também contribuiu para o entendimento do produto. Mas além desta vivência, foi necessário um período de descoberta das premissas do problema e suas respectivas validações com o auxílio de possíveis clientes, e posteriormente foi elaborada uma proposta de solução com potencial de ser efetiva.

Para melhor definição dos problemas a serem abordados no produto final e refinar as suas prioridades, foram realizadas entrevistas com potenciais clientes para colher feedback dos requisitos levantados a fim de melhorar a compreensão das descobertas e suas validações e até as propostas de soluções.

E durante todo estes processos de entendimento, compreensão, validações e desenvolvimento do projeto, foram realizados diversos *brainstorming*, tanto para criar novas funcionalidades quanto para descoberta de melhores formas para mapear as funcionalidades já propostas. Com o intuito de gerar uma uma solução com potencial, que possa ser efetiva mas seja viável de ser criada e mantida.

4.2.3 Levantamento de requisitos

Conforme mencionado na subseção 4.2.1, a gênese deste projeto remonta à iniciativa do PET Sistemas. Nesse sentido, procedeu-se à análise do sistema preexistente, identificando suas funcionalidades para mapeá-las em forma de requisitos.

A segunda principal forma de levantamento de requisitos no projeto foi a partir da experiência prévia dos integrantes, tanto da perspectiva de participantes quanto de membros da equipe organizadora.

Inicialmente este projeto não tinha um cliente específico para acompanhar seu desenvolvimento e dar feedback, e para compensar, foi necessário entrevistas com possíveis potenciais usuários que teriam interesse em utilizar o sistema em diferentes perspectivas, por possuir diferentes perfis, como alunos, professores, servidores administrativos e professores ou servidores que já organizaram eventos.

A primeira perspectiva foi a partir de conversa com os professores que têm vivência organizando eventos em diferentes áreas do conhecimento. A segunda perspectiva, foi a partir dos servidores da PROECE (Pró-Reitoria de Extensão, Cultura e Esporte), onde houveram entrevistas para entender suas necessidades quanto ao gerenciamento de eventos dentro da UFMS.

Com base nessas duas primeiras formas de compreensão do problema e nas experiências dos entrevistados conforme pode ser visto na tabela 2 , foram encontrados os requisitos essenciais do projeto. Chegou-se ao resultado que atendia a intersecção entre as duas necessidades de cada perspectiva, sendo este o núcleo deste resultado, o registro de presenças, para emissão em massa de relatórios de certificados no SISCERT.

Perfil	Experiência	Descrição das atividades que participou
Professores da	Professores com experiência em	● Criação de eventos;

computação e química	criação e apoio na operação de eventos médios e grandes, como por exemplo Semana da Química e WCPD (Workshop de computação paralela e distribuída)	● Apoio como equipe; organizadora de eventos; ● Ministrantes de atividades em eventos.
Servidores da PROECE	Funcionários que trabalham na area de eventos da UFMS tendo projetado e apoiado a operação de eventos como o Integra e Peças de teatro no glauce rocha	● Criação de eventos; ● Apoio como equipe organizadora de eventos; ● Resolução de problemas burocráticos envolvendo documentação com professores.
Alunos da UFMS	Alunos com interesse em aprendizado e habilidades para compartilhar conhecimento, bem como em estabelecer conexões durante eventos.	● Apoio como equipe organizadora de eventos; ● Ministrantes de atividades em eventos; ● Participante em eventos.

Tabela 2. Perfis dos possíveis usuários dialogados

4.2.4 Análise de Requisitos

Ao decorrer do processo de descobertas das funcionalidades mapeadas, foi percebida a necessidade de ordenar as funcionalidades em prioridade. Para assim saber quais seriam priorizadas e quais poderiam ser implementadas posteriormente em uma segunda versão. Como abordado na subseção 2.1.7, foi escolhido o FDD, para organizar e priorizar os requisitos coletados na etapa de levantamento de requisitos.

As funcionalidades foram agrupadas por afinidade em módulos. Onde cada funcionalidade recebeu uma prioridade na escala de alto, médio e baixo. As principais funcionalidades que inicialmente contemplaria todos os módulos, foram seus respectivos *CRUD*, que é um acrônimo para *Create*, *Read*, *Update*, *Delete*. E os principais módulos criados foram de evento, atividade, inscrição, sessão, presença, relatório de inscritos e presentes, categoria, usuário, cargo e permissão.

Além dos principais módulos citados acima, existem funcionalidades que têm o potencial de agregar valor ao projeto, mas que foram catalogadas como não essenciais,

por exemplo, cadastro de locais com lotes e cadeiras numeradas, reserva de locais, cautela de materiais, *staff* (em portugues equipe de organizadora) e tarefas da *staff*.

4.3 Análise de Projeto

Após entender o produto e quais possíveis problemas dos usuários poderiam ser abordados neste projeto, foi iniciada a priorização das funcionalidades que deveriam compor a primeira versão entregável do produto e quais poderiam ficar para a segunda versão. As principais formas de definição das funcionalidades foram guiadas através de entrevistas com professores, servidores da PROECE e alunos da UFMS. Em paralelo, foi iniciada a documentação do sistema, sendo criado os documentos de escopo e funcionalidades em conjunto com as discussões sobre a arquitetura do sistema.

4.3.1 Funcionalidades mapeadas do projeto

Foram mapeadas diversas funcionalidades que podem agregar valor ao projeto, mas em virtude do tempo, foram selecionadas somente funcionalidades que compõem a proposta do núcleo, as quais giram em torno de registrar presença do participante.

Abaixo estão todas as funcionalidades que foram colhidas, e que tem o potencial de agregar valor ao projeto, estando organizadas por conjuntos de entregáveis, onde cada camada agrupa as funcionalidades em ordem de prioridade.

O Primeiro nível de entregável representa as funcionalidades que estão implementadas nesta versão (versão 1.0). O segundo nível de entregável representa as funcionalidades que recomendamos que sejam implementadas em seguida. E a terceira camada representa funcionalidades que agregam valor para públicos mais específicos.

4.3.1.1 Entregável um (Núcleo)

- Gerenciamento de usuário:
    - cadastrar, recuperar, atualizar, deletar, recuperar senha, autenticar;
- Gerenciamento de cargos e permissões:
    - cadastrar, recuperar, atualizar, deletar, autorizar, atribuir e revogar cargo;
- Gerenciamento de evento:
    - cadastrar, recuperar, atualizar, deletar;
- Gerenciamento de atividade:
    - cadastrar, recuperar, atualizar, deletar;

- Gerenciamento de sessão:
    - cadastrar, recuperar, atualizar, deletar;
- Gerenciamento de inscrição na atividade:
    - cadastrar, recuperar, deletar;
- Gerenciamento de presença:
    - registrar, recuperar, deletar;
- Gerador relatório de inscritos e de presentes:
    - cadastrar, recuperar, deletar;
- Gerenciamento de categoria:
    - cadastrar, recuperar, atualizar, deletar;

4.3.1.2 Entregável dois

- Gerenciamento de Instituições
    - cadastrar, recuperar, atualizar, deletar, atribuir responsável
- Gerenciamento de unidades da instituição
    - cadastrar, recuperar, atualizar, deletar, atribuir responsável
- Gerenciamento de servidores da instituição
    - cadastrar, recuperar, atualizar, deletar
- Gerenciamento de recursos da instituição
    - Gerenciamento de reserva de locais
        - cadastrar, recuperar, atualizar, deletar, solicitação de reserva, resposta de solicitação de reserva, termo de reserva
    - Gerenciamento de cautela de materiais
        - cadastrar, recuperar, atualizar, deletar, solicitação de cautela, resposta de solicitação de cautela, termo de cautela
- *Staff*
    - cadastrar, recuperar, atualizar, deletar, termo de compromisso e sigilo
- Tarefas para *staff*
    - cadastrar, recuperar, atualizar, deletar, atribuir tarefa para membro da *staff*
- Gerenciamento de envio de e-mail em massa
    - filtro para selecionar público alvo

4.3.1.3 Entregável três

- Venda de ingresso
- Página de divulgação do evento
    - notícias e avisos, fale conosco, mídia social e tira dúvidas/FAQ, cronograma e agenda, sugestões e enquete de votação
- Gestão de patrocinador e apoiadores e organizador do evento
- Gerador de arquivo com dados para impressão de crachás
- Geração de arquivo CSV para emissão de certificado baseado em notas por sessão para calcular média e gerar relatório de certificado da atividade
- Gerar relatório de análise
    - dashboard de relatórios e estatísticas do evento e suas atividades
- Gerenciamento de submissão de trabalho e avaliação
- Gamificação e insígnia
- Network de participantes
- Materiais cedidos para *staff* e evento
    - materiais de consumo, permanente e doação
- O sistema deve saber lidar com a falta de conexão à internet de forma manual ou automática de exportar e importar dados de registro de presença;

4.4 Divisão dos entregáveis do projeto

Os módulos foram implementados de acordo com o tempo disponível da equipe e com a prioridade das funcionalidades. Desta forma, nem todos os módulos estão completamente implementados. Das funcionalidades mapeadas na seção 4.3.1, somente as foram implementadas na versão 1.0, sendo elas as que constam na camada um.

A divisão de camadas dos entregáveis do projeto foi considerada significativamente abrangente, tendo diversas funcionalidades que não puderam ser contempladas em uma única entrega. E para esta, focamos nas funcionalidades básicas e nas que entendemos que serão cruciais. E foi construído um *roadmap* preparado com sugestões de implementações na ordem que é recomendada para o projeto.

5 Diferentes perspectivas de cada papel

Uma das maiores dificuldades encontradas no desenvolvimento do sistema foi nas diferentes perspectivas que cada perfil de usuário tem. A gestão de perspectiva do usuário logado com diferentes níveis de acesso, e assim exibir distintas funcionalidades, de acordo com as permissões de acesso.

Por exemplo, o usuário pode ser simultaneamente o dono de um evento, participante de uma atividade, ministrante de uma sessão e até executar tarefas da Staff. Outro exemplo, o usuário pode ser dono de eventos, sendo que este usuário pode criar um evento institucional por ser servidor como professor ou ser da PROECE, ou até mesmo criar um evento não institucional sendo somente estudante.

5.1 Responsável pela instituição

Nesse caso, pode ser o reitor, o diretor do departamento responsável por coordenar eventos da instituição, ou qualquer outra autoridade designada. Essa pessoa é responsável por supervisionar os servidores da instituição e até os eventos de sua instituição, para assim garantir que esteja alinhado com a visão e missão da instituição.

5.2 Servidor institucional:

São os funcionários ou servidores da instituição que podem estar envolvidos na organização e execução do eventos, ou até supervisão dos mesmos. Eles desempenham uma variedade de funções, desde a logística e a coordenação até o suporte administrativo.

Os servidores institucionais podem tanto criar eventos e utilizar espaços específicos da instituição, por exemplo o teatro Glauce Rocha da UFMS[7]. Como também poderiam acessar outros eventos, desta maneira o sistema proporciona que os servidores atuem como moderadores, para caso seja infringido as diretrizes da instituição seja possível censurar.

[7] Teatro usado para apresentações diversas dentro da UFMS, para mais informações acesse o link https://proece.ufms.br/teatroglaucerocha/

5.3 Dono do evento

O dono do evento é a pessoa responsável pela concepção e planejamento do evento. Esse papel pode ser desempenhado por professores, coordenadores de programas acadêmicos, comitês específicos, servidores e até por acadêmicos. O dono do evento define os objetivos, temas e conteúdos, além de estabelecer parcerias estratégicas, sendo o responsável por gerar os relatórios de certificados. Eles supervisionam todo o processo de organização para garantir a qualidade e o sucesso do evento.

5.4 Participantes

São os indivíduos que se inscrevem e participam do evento. Eles podem ser estudantes, pesquisadores, professores, profissionais de áreas relacionadas ou público em geral, dependendo da natureza do evento. Os participantes são o público-alvo do evento e têm a oportunidade de assistir às palestras, participar de workshops, compartilhar conhecimentos e interagir com outros participantes.

5.5 Ministrante

São os especialistas, profissionais ou acadêmicos convidados para realizar palestras ou apresentações durante o evento. Eles trazem conhecimentos, experiência e perspectivas relevantes para enriquecer o conteúdo do evento. Os palestrantes compartilham seus conhecimentos e insights, contribuindo para o aprendizado e o diálogo no evento.

5.6 Administrador do sistema (ROOT)

Esse papel é desempenhado por uma pessoa de confiança da instituição, normalmente algo da seção de Tecnologia da Informação. Esta pessoa deverá gerenciar as instituições que estão cadastradas no sistema, desde de adicionar uma nova instituição até atribuir o cargo de “responsável da instituição”. O administrador do sistema garante que as instituições cadastradas são permitidas.

5.7 *Staff*

A *staff* refere-se à equipe de apoio que trabalha no evento, selecionada pelo "dono do evento", executando tarefas como recepção, apoio logístico, organização de materiais, entre outros. Esses membros asseguram que todas as atividades e sessões aconteçam conforme o planejado, oferecendo suporte aos participantes e aos palestrantes.

O dono do evento, sendo um servidor da instituição, pode montar sua equipe organizadora *staff*, para executar tarefas, enviando convites para outros usuários do sistemas, para que eles componham a *staff*. O membro da equipe organizadora terá outra perspectiva quando estiver logado no sistema, tendo uma visão geral de quais eventos ele participa da *staff*, em cada evento poderá ver as tarefas criadas pelo dono do evento e se voluntariar para executar, assim se responsabilizando, representado na figura 6.

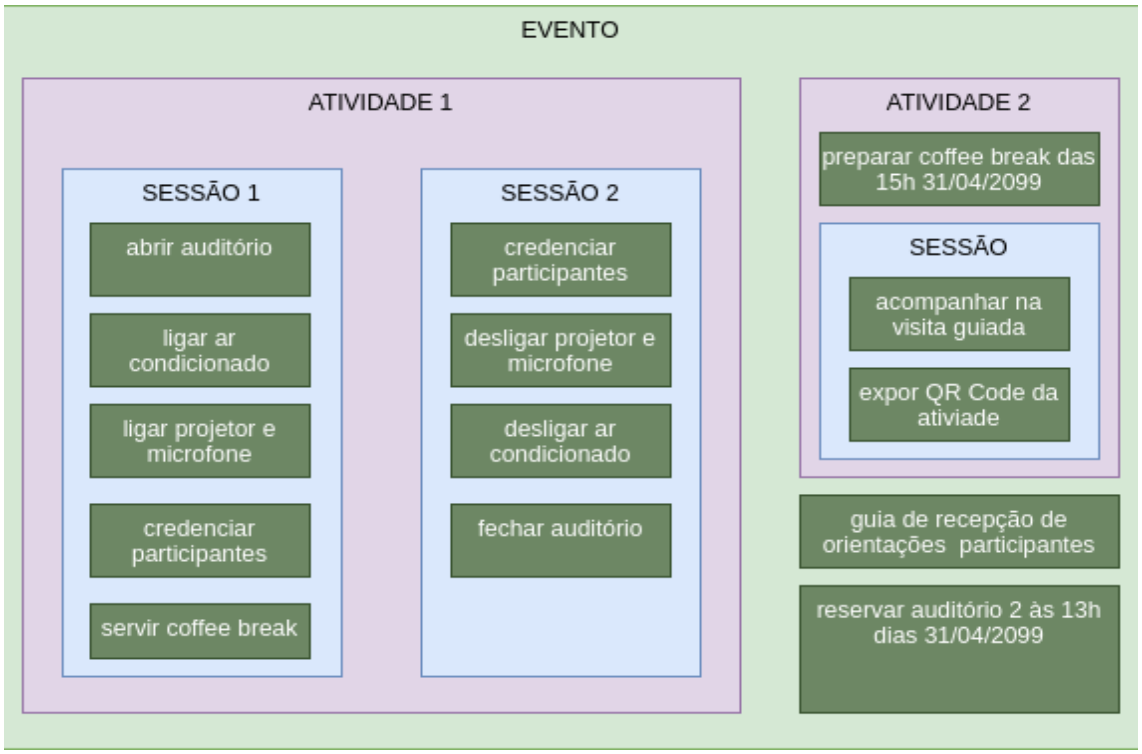

Figura 6. Representação de um evento no sistema com suas tarefas

5.8 Permissões

O permissionamento do sistema funciona com permissões que representam uma ação que um usuário pode fazer de forma global ou dentro de um evento específico que está representado na tabela 3, por exemplo criar atividade, e essas permissões são agrupadas em cargos. Na primeira entrega, o módulo de permissões trabalha somente de forma global, ou seja, não foi implementado a possibilidade de criar permissões para usar em um evento específico.

Perfil	Permissões	Observações	Está nesse entregável
Responsável Pela Instituição	• Pode cadastrar novos "servidores institucionais".	Pendente implementação do entregável dois	
Servidor Institucional	• Pode censurar evento [situação desativada].	Pendente implementação da entregável dois	
Dono do Evento	• Pode criar, editar e deletar seus próprios eventos, atividades, sessões; • Remover inscrições de atividades; • Registrar e deletar presenças; • Pode gerar arquivo de planilha de extensão .xlsx contendo a lista de inscritos de uma atividade do seu evento ou de		

	presenças de uma de suas atividades de seu evento.		
Administrador do Sistema	• Tem todas as permissões no sistema; • Pode cadastrar novas instituições.	Pendente implementação de instituição	
Staff	• Pode registrar presença de participantes em sessões em que seja membro da *staff* do evento.		
Participantes	• Pode ver a lista de eventos - pode ver a lista de atividades por evento; • Pode ver a lista de sessões por atividade - pode se inscrever em atividade - pode acessar seu próprio QR Code - pode registrar a própria presença (caso a sessão permita).		
Palestrantes	• Pode escrever na sessão texto livre, como link do conteúdo utilizado.	Pendente implementação da entregável dois	

Tabela 3. Perfis mapeados e suas permissões no sistema

6 Regras de Negócio

6.1 Regras de negócio da instituição

Dentro de cada instituição, o usuário terá somente um registro único por CPF e e-mail, para garantir a consistência. E no caso de pessoa estrangeira, será aceito o passaporte. Somente o responsável pela instituição pode cadastrar usuário como servidor da instituição, como servidor docente ou servidor administrativo. Os servidores devem ser vinculados a unidades da instituição, cada unidade terá um responsável, no qual fica sob sua responsabilidade gerenciar quais servidores institucionais tem vínculo com a unidade.

Somente servidor administrativo designado pelo responsável da instituição é encarregado por seus recursos, podendo cadastrar e alterar recursos como locais e materiais que podem ser reservados, aprovando solicitações e validando a devolução nas condições de conservação previstas no termo de responsabilidade. Qualquer usuário pode se autodeclarar como discente da instituição, mas necessitando de confirmação de sua legitimidade feita por um servidor da instituição. Desta forma é possível criar eventos e atividades exclusivos para os discentes.

6.2 Regras de negócio do evento

Quando um evento é criado, automaticamente o usuário logado se torna o dono, sendo que cada evento pode possuir somente um único dono. E em caso de necessidade poderá ser transferido para outro. Para garantir a segurança do controle de acesso, somente o dono do evento realizará algumas ações no sistema, como criar e atualizar as atividades e sessões, convidar outros usuários para compor a equipe organizadora da *staff*, criar as tarefas que a *staff* deve executar e solicitar reserva de locais e materiais.

Somente o dono do evento e sua equipe organizadora poderão visualizar evento, atividade e sessão que estiverem com status privado. Quando o evento estiver quando o status público, ficará visível até mesmo para usuário que não estiver logado. E sendo listado em buscas por filtros. A atividade pode estar com status que permita estar visível somente para seus inscritos e participantes que estavam presentes, para poderem acessar informações restritas, como links de formulários e conteúdos como PDF e slide, na

figura 7 é ilustrado como o sistema trabalha essa visibilidade e hierarquia entre evento, atividade e sessão, hierarquia demonstrado de melhor forma na figura 8.

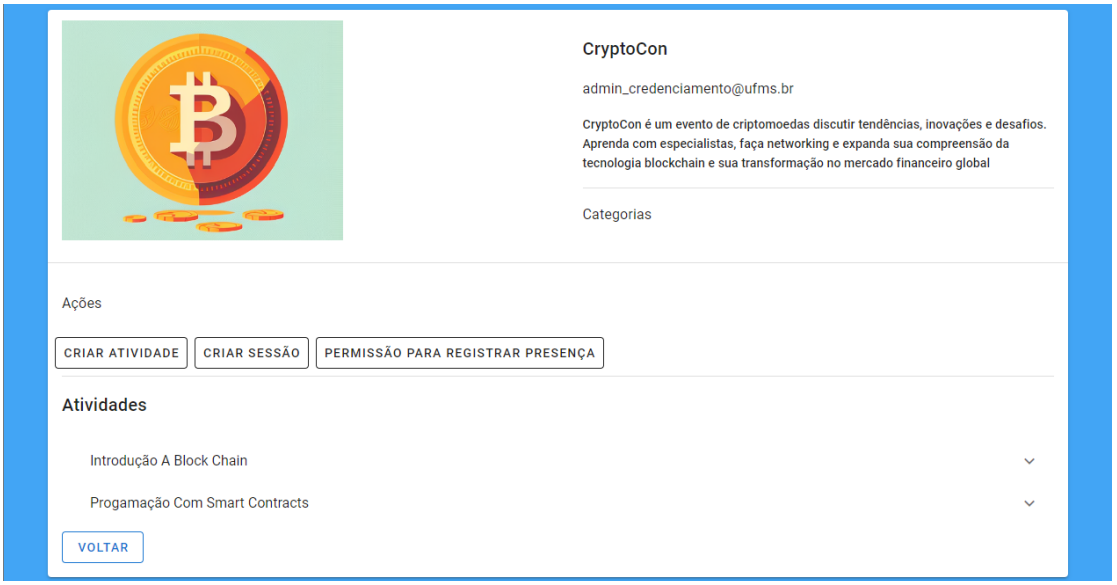

Figura 7. Composição do evento com atividade e sessão dentro do sistema

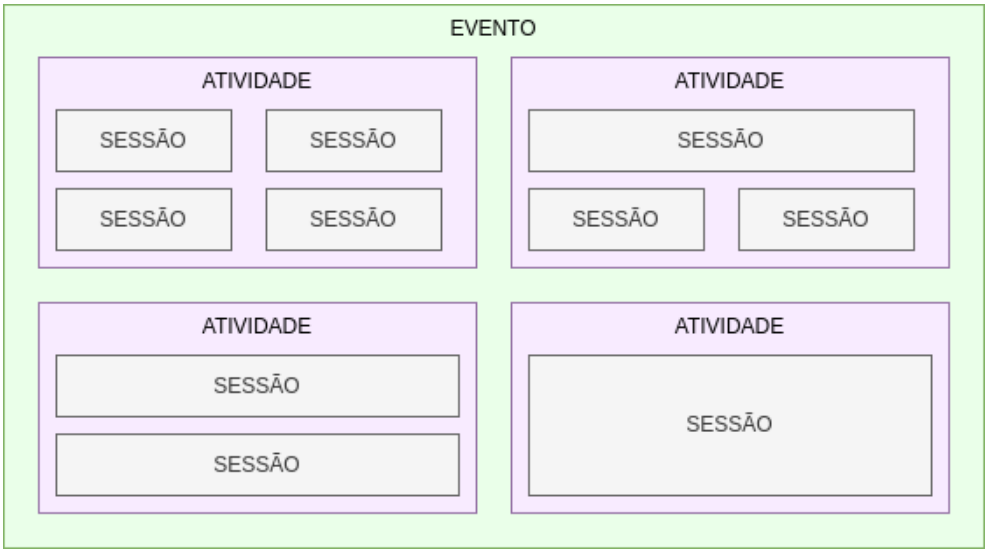

Figura 8. Hierarquia do evento com sua atividades e sessões

6.3 Registro de Inscrição

A inscrição na atividade, é uma forma de reservar vaga, para que o participante tenha segurança. Desta maneira é possível controlar a quantidade vagas seja respeitada. E também, que somente pessoas com os pré-requisitos participem da atividade.

6.4 Regras de negócio do credenciamento

Com base no entendimento de como é um evento para o sistema de acordo com a figura 10, a presença dentro do sistema está atrelada às sessões criadas dentro das atividades. A forma exigida para realizar o registro da presença está determinada pelo dono do em cada sessão da atividade, e que por padrão é “Ler QR Code do inscrito”.

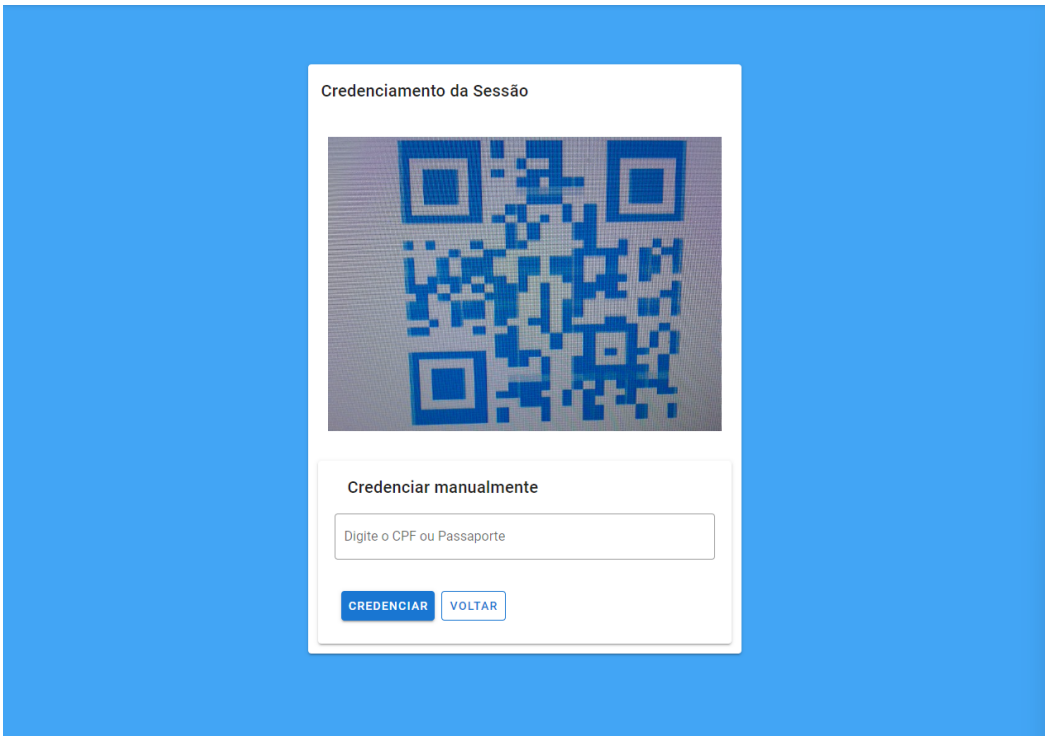

Figura 9. Tela de registro de presença do sistema

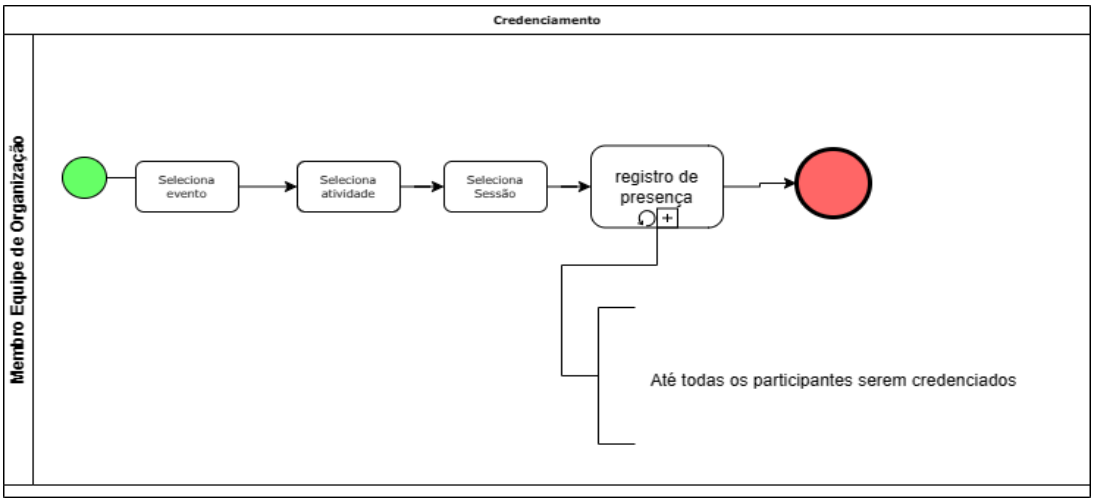

Figura 10. Diagrama BPMN registro de presença

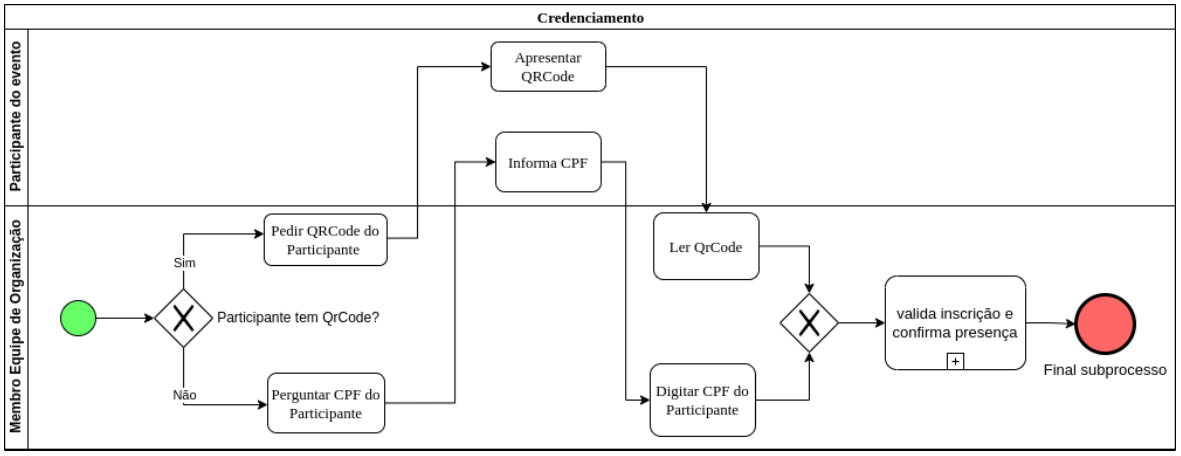

Figura 11. Diagrama subprocesso de registro de presença

Figura 12. Diagrama subprocesso de validação e confirmação da presença

Para poder realizar o registro de presença no momento do credenciamento, a sessão deve estar com status que permita a ação. Existem cinco cenários para realizar o registro de presença do credenciamento do participante em uma sessão. Dentre os quatro, existem duas principais formas e as outras duas são paliativas em cenários de exceções.

O dono do evento deve previamente determinar na atividade ou especificando na sessão, qual forma que deseja que o registro de presença seja feito no momento do credenciamento pelo membro da equipe da *staff*. O primeiro cenário é a forma padrão e tradicional de credenciamento, onde o participante já está inscrito na atividade, o'que lhe garante estar com vaga reservada, e se identifica apresentando seu QR Code para o membro da *staff* para ler, ou informar o CPF para que seja digitado manualmente.

O segundo cenário é com a forma de presença alternativa que é útil para eventos remotos ou em casos que existe uma grande quantidade de participantes para evitar gargalos no credenciamento. Se dá pelo membro da staff ou o ministrante expor o QR code da sessão ou o código de identificação da sessão, para que os participantes se auto registrem como presente.

O terceiro cenário é paliativo, e acontece quando o participante não possui inscrição na atividade desta sessão em que está tentando se credenciar. Neste cenário, o membro da *staff* irá receber um alerta, avisando que o participante não possui vaga reservada, e perguntando se deseja confirmar o registro de presença ou deseja cancelar.

O quarto cenário também é paliativo, acontece quando o participante não possui cadastro no sistema, mas deseja receber o certificado. Neste cenário, o credenciador pode solicitar para que o participante se cadastre no sistema antes de realizar o registro de presença. Ou registrar manualmente seu CPF e e-mail para posteriormente entrar em contato para colher as informações necessárias para gerar o certificado.

6.5 Regras de negócio da *staff*

Quando o dono do evento desejar enviar convite para outros usuários compor a *staff* para ser membro da equipe organizadora, deverá buscar cada usuário por CPF, para garantir que o sistema não vaze dados de usuários sem autorização, como orientado pela LGPD. Desta forma somente usuário que é servidor da instituição terá o privilégio de visualizar o nome de usuário a partir do CPF informado, assim ter a notificação de confirmação que o convite foi enviado com sucesso.

6.6 Regras de negócio da página principal

A página principal da perspectiva do participante por padrão inicia com a lista todos os eventos públicos que estão com as inscrições abertas, sendo possível aplicar diversos filtros para encontrar eventos. A página principal da perspectiva do dono do evento mostra suas opções de ações para gerenciar suas atividades e sessões, como a lista de inscritos, e a lista de presentes.

6.7 Relatório

O módulo de emitir relatório é uma forma estruturada para exportar dados do sistema, em extensão CSV. De acordo com a estratégia de processamento desejada e com as devidas condições necessárias para filtrar os registros, por exemplo considerando o status. Para atender as definições e critérios de condições desejado.

Sendo amplamente utilizada por lista de inscritos e lista de presentes para emissão de relatórios de certificados. No caso do relatório de inscritos, pode ser gerado pelo dono do evento ou membro da *staff*. É gerado a partir de uma atividade, listando todos os seus inscritos.

6.7.1 Regras de negócio para emissão de relatório de certificados

A geração de relatório de presença para emissão de certificados é um ponto importante para a longevidade do sistema, pois a partir dele pode-se exportar dados de forma estruturada de todos os tipos e extensão de arquivo e com regras variadas. Sendo que todas as configurações do evento devem ser configuradas pelo seu respectivo dono.

Com base na estratégia de emissão de relatório escolhida, as condições e critérios podem ser considerados para gerar o arquivo de extensão CSV contendo a lista de presentes que irão receber o certificado.

No modelo padrão de emissão de relatório de certificados em massa, a estratégia é focada no SISCERT, onde o critério é determinado pelo dono do evento, na criação da atividade, definindo a quantidade mínima de presença nas sessões para que o participante receba o certificado. E o status de emissão deve permitir a ação, como “Aprovado” ou “Emitido”. No qual gera um arquivo de extensão CSV e que por padrão, o sistema inicialmente irá ser contemplado com duas estratégias para gerar lista de presentes, uma que busca todos os presentes de um evento, e outra que busca todos os presentes por uma atividade.

Além dessas duas, é possível adicionar estratégias para emissão de relatório de presentes para poder atender às distintas necessidades que podem ser requisitadas. Desta forma o sistema é preparado para receber diferentes formas de emissão de relatórios de presenças, sem precisar alterar o código-fonte existente, apenas adicionando uma nova.

7 Fluxo de Desenvolvimento

7.1 Organização das demandas

As demandas foram organizadas levando em consideração suas prioridades e estimativa de custo de tempo para desenvolver. Foi promovido o acompanhamento das demandas foi feito regularmente com reuniões quase que diárias, garantindo que o progresso estivesse alinhado com as metas estabelecidas entre os membros do projeto.

Priorização das demandas foram criadas a partir das seguintes considerações. Ficando no começo do da coluna do Kanban com maior prioridade. Primeiro foi buscado priorizar demandas sem ou com menor dependências entre si. Segundo, as demandas mais críticas e estruturais receberam maior prioridade.

A distribuição de demandas entre os membros da equipe aconteceu de forma natural, onde cada um se responsabilizou pelas no qual tinha mais afinidade, por exemplo as de infraestrutura, devOps, configuração de ambiente, autenticação e autorização de usuários, implementar endpoints de CRUD e as regras de negócio.

7.2 Gerência de configuração de software

Como optamos por usar o Github como ferramenta de repositório remoto, usamos algumas de suas ferramentas em nosso processo de controle de versão, como *Pull Request*, e execução dos testes automatizados que são executados automaticamente após um *push* e processos de processo de CI/CD (Continuous Integration/Continuous Delivery) (integração e entrega contínuas).

O projeto foi dividido em três repositórios, sendo dois para o *frontend* e um para o *backend*. É esperado que desta forma será melhor para gerenciar a evolução do projeto e caso exista a necessidade de alteração e ou refatoração de algum componente, a separação em repositórios distintos facilitará a manutenção e a escalabilidade do sistema.

O fluxo de trabalho para as demandas começa com a criação da *branch* para o desenvolvimento da funcionalidade. Após o desenvolvimento, é aberta uma *Pull Request*, que é um pedido de integração do código-fonte de destino.

Em seguida, o código passa pelo processo de CI/CD, sendo revisado por outro membro do projeto e, se aprovado, é integrado à branch de desenvolvimento, onde é

mantido em sua versão de homologação. Após atualizar a branch de homologação, são realizados testes manuais pelos membros da equipe, e assim o código é incorporado à branch principal "*main*", e assim estando pronto para o ambiente de produção.

7.3 Processo de trabalho e comunicação da equipe

De maneira remota e assíncrona, a equipe se dividiu para executar as demandas estabelecidas no kanban. Durante o processo de desenvolvimento do dia a dia, a comunicação foi centralizada em um canal de texto no Discord.

Foram realizadas reuniões semanais por chamada de voz para entender o progresso, identificar dificuldades e replanejar as demandas, priorizando as que serão executadas nas próximas etapas. Simultaneamente, foram incorporadas reuniões adicionais ao nosso cronograma semanal para compor o alinhamento entre os membros.

7.4 Recursos auxiliares

Foi utilizado o Google Drive para armazenar a documentação e diagramas gerados durante o desenvolvimento deste projeto, a sua escolha foi feita de forma a facilitar o trabalho em equipe de acessar remotamente e atualizar os arquivos.

A gestão de arquivos como diagramas, prints, documentos, planilhas, slides e relatórios foi feita utilizando ferramentas do Google. E os diagramas foram gerados utilizando a plataforma Draw.io.

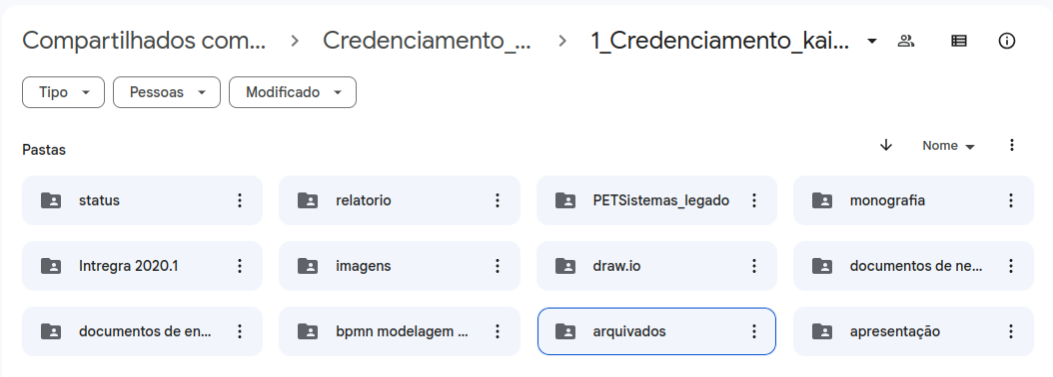

Figura 13. Print do Google Drive

8 Ambientes

8.1 Ambientes da aplicação

Foram preparados três ambientes para aplicação, utilizando os recursos de containers com Docker [Muddinagiri, 2019] e se utilizando de conceitos de Infraestrutura como Código. Para que assim facilite o processo de execução do projeto em cada ambiente.

O primeiro ambiente é o de desenvolvimento, e foi criado com o intuito de que a equipe de desenvolvimento possa trabalhar de forma harmoniosa na mesma aplicação mesmo em ambientes distintos. Foram criados scripts para automatização de processos, por exemplo de CI/CD [Singh, 2019], *Seed*, *Factory* e *script* bash.

O segundo ambiente é o de produção, e foi criado com o intuito de facilitar a implantação do sistema em um servidor para ficar utilizável pelos clientes. O processo de deploy está automatizado utilizando containers Docker com docker-compose.yml.

O terceiro ambiente é o de testes, configurado para executar os testes sem interferir em qualquer outro ambiente, um caso de teste não interfere em outro. Desta forma é possível executar funcionalidades automaticamente e verificar se o comportamento é esperado, capturando inconsistência mapeadas.

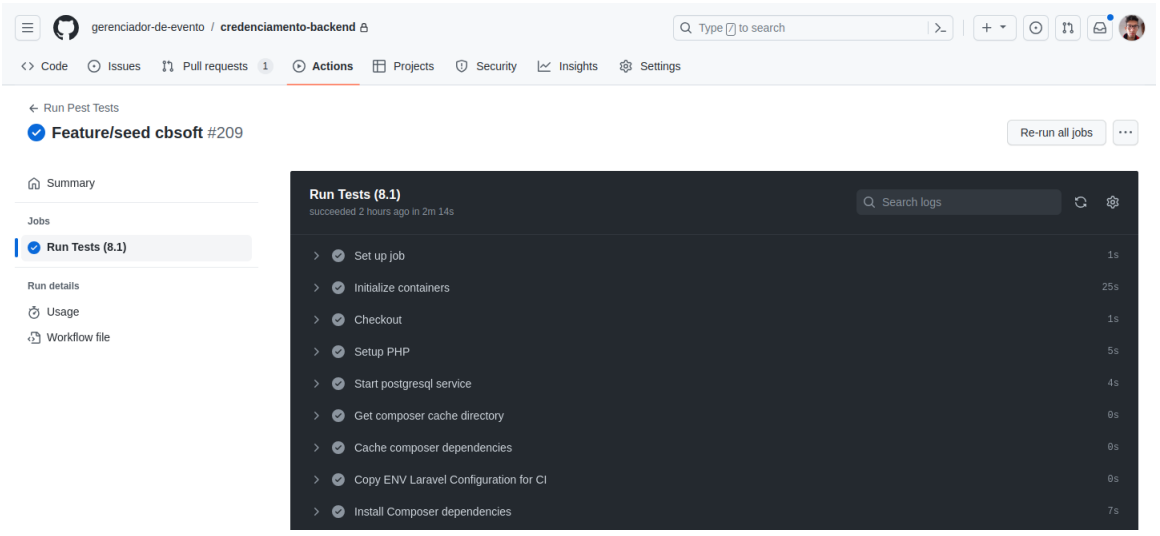

Figura 14. Print do Github *action* executando testes no CI

9 Banco de dados

9.1 Introdução

Existem dois principais modelos de banco de dados, o relacional (SQL, do inglês *Standard Query Language*) e não relacional (NoSQL, do inglês *Not Only SQL*). A principal diferença está na integridade de dados e flexibilidade. Banco de dados SQL são inflexíveis, com isso provendo maior integridade dos dados. Os bancos de dados NoSQL são flexíveis, deixando a integridade dos dados dependente do desenvolvedor, mas podem ajudar em outros requisitos como a clusterização. [Khan, 2019].

No desenvolvimento da aplicação foi utilizado o banco de dados relacional PostgreSQL, pelo seu funcionamento em conjunto com o *framework* utilizado e por todas os benefícios que um banco relacional traz consigo como ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

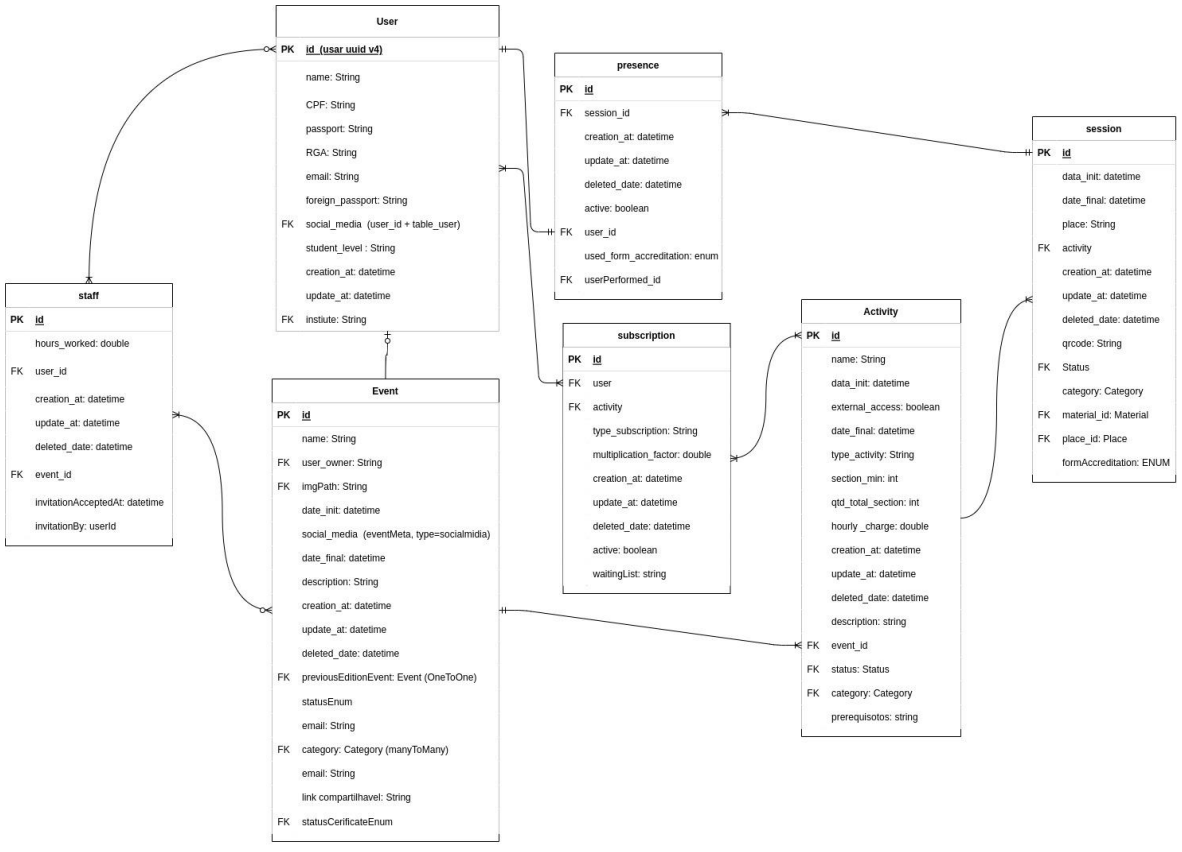

Figura 16. Modelo de entidade relacionamento reduzido para exemplificação

10 Arquitetura da aplicação

10.1 Arquitetura geral

As principais decisões arquitetônicas foram descritas ao decorrer do texto com suas respectivas considerações, desde os problemas e motivações, até seus benefícios, visando sua facilidade de manutenção para aumentar seu ciclo de vida do sistema.

A aplicação está em um arquitetura dividida em dois *frontend* e um *backend* com um único banco de dados. Sendo que a comunicação entre as aplicações *frontend* para com a aplicação *backend* é através de HTTP no padrão REST *(Representational State Transfer*, em português Transferência de Estado Representacional).

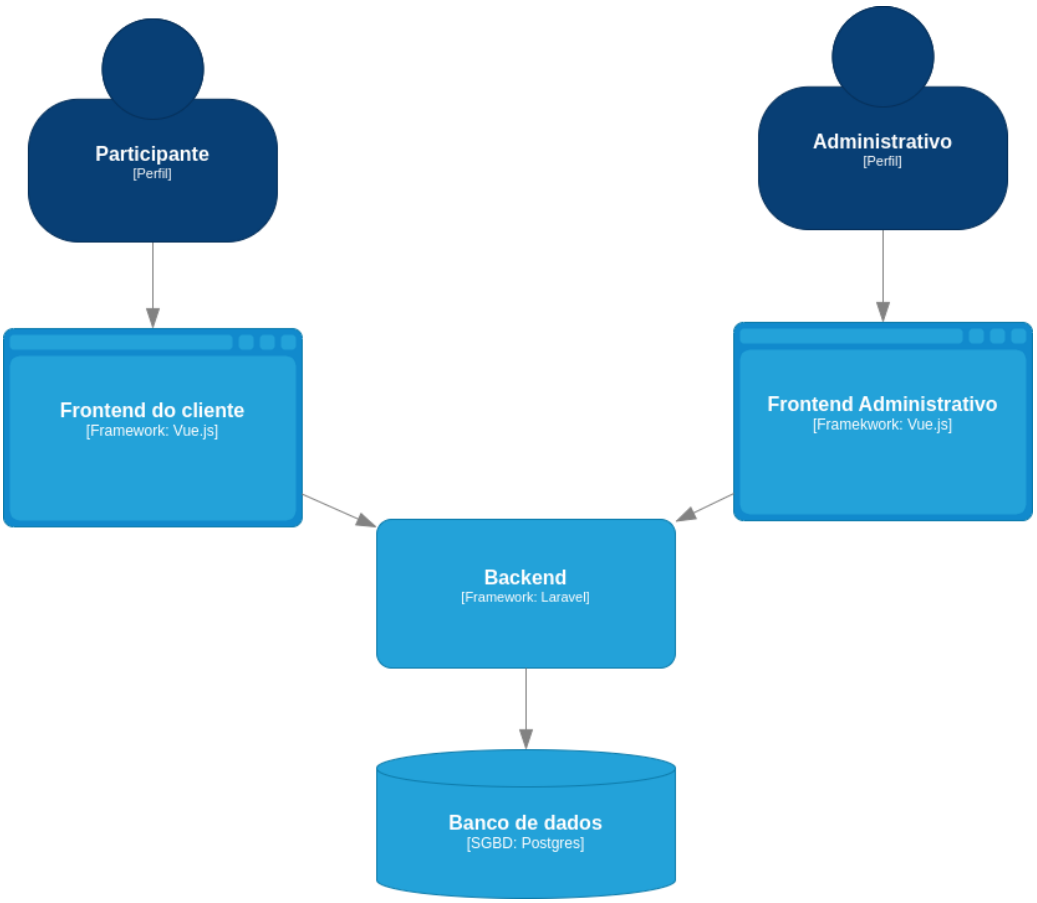

Figura 17. Diagrama C4 da arquitetura de solução do projeto

11 Backend

11.1 Arquitetura do backend

O *backend* está em uma arquitetura monolítica, inspirada principalmente na tradicional Arquitetura Hexagonal [Cockburn, 2023], também conhecida como "*Port and Adapter*". Que é um padrão arquitetural que visa criar sistemas de software altamente modulares, flexíveis e de fácil manutenção.

Essa abordagem arquitetural, foi concebida para separar claramente as preocupações do negócio central das implementações técnicas, promovendo assim uma melhor organização do código e uma maior adaptabilidade às mudanças.

Foi criada pelo Dr. Alistair Cockburn, em 2005, e abaixo temos uma figura criada em 2008 por Juan Manuel Garrido de Paz [Garrido, 2023] que, de forma simples, exibe a proposta desta arquitetura, com o seu principal intuito de isolar a camada de negócio de camadas externas [Cockburn, 2023].

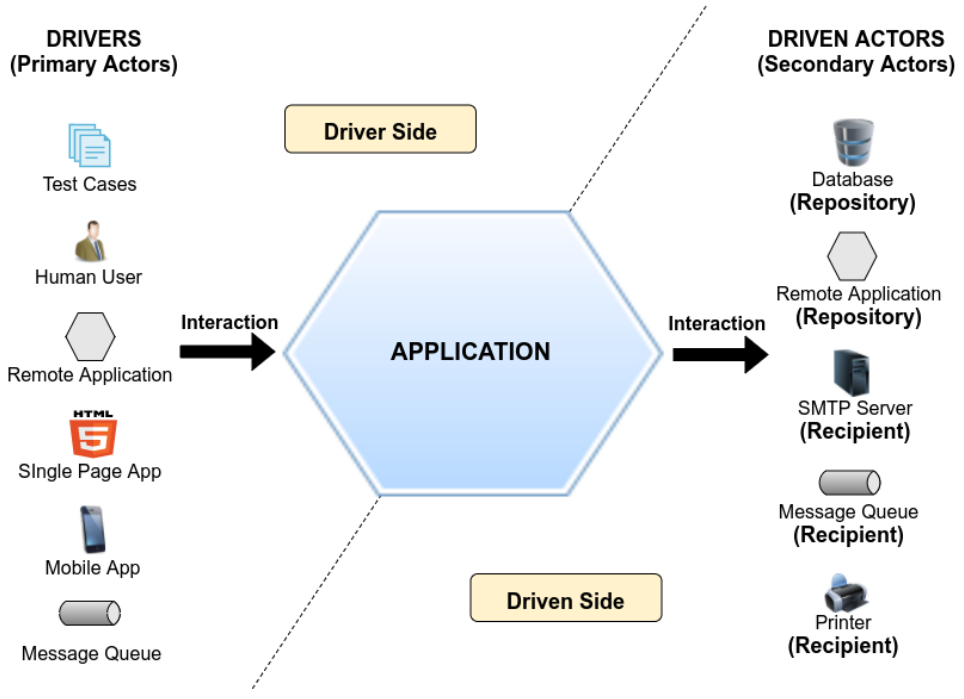

Figura 17. Diagrama Arquitetura Hexagonal [Garrido, 2023]

A aplicação foi baseada nessa arquitetura, trazendo de forma razoável o conceito de desacoplamento entre o framework utilizado e a separação da regra de

negócio implementada dentro de um única aplicação web monolítica usando API (*Application Programming Interface*, do portugues, Interface de Programação de Aplicação) REST.

E neste projeto, foi adotado limites arquitetônicos que atendem tanto o padrão do Laravel quanto permite isolar a regra de negócio, adicionando uma camada interna, separando casos de uso "*service*" de regra de negócio "*domain*".

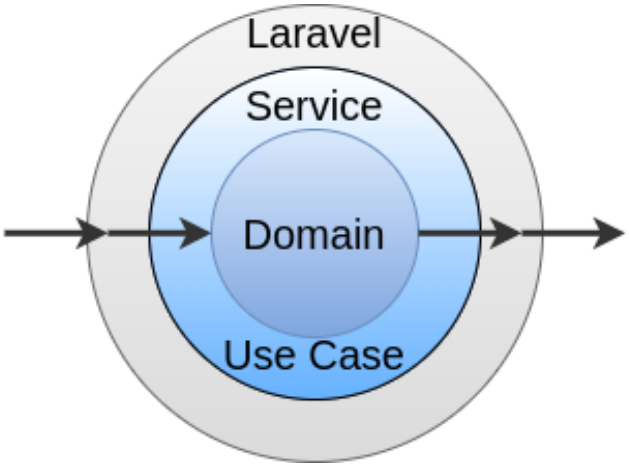

Figura 18. Diagrama dos limites arquitetônicos entre *framework*, negócio e serviço.

Utilizou-se uma arquitetura orientada a domínio dentro da camada de "*Application*". Assim, toda a camada de domínio é isolada, contendo somente regra de negócio. E a regra de negócio é executada pela camada de serviço "*service*", e não pode depender de implementações externas "*adapters*", somente de abstrações "*ports*".

Desta forma, mudanças externas não impactam diretamente nas regras de negócio e nem no fluxo que o negócio é executado. Tornando o sistema mais tolerante a mudanças. Usando o "Princípio da inversão de dependência" do SOLID [Yang, 2008] na camada *service*, os casos de uso não dependem diretamente de código externos, mas sim de abstrações "*port*". E o conceito de Injeção de Dependência é aplicado à camada de controle "*controller"* para que a implementação literal seja passada para o serviço.

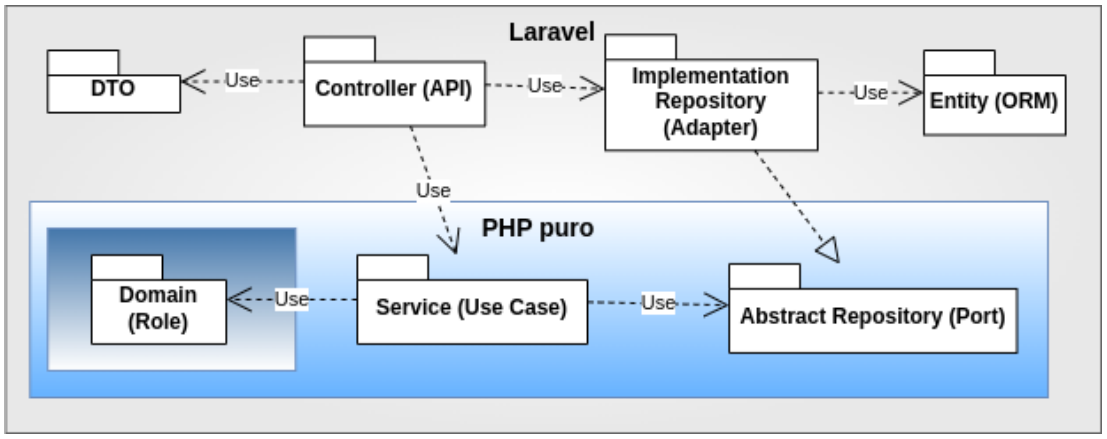

Figura 19. Diagrama de pacotes entre *framework*, serviço e regra de negócio

Alguns diretórios e arquivos foram criados respeitando o padrão recomendado pelo Laravel, como *route*, *controller*, *model*, *helper*, *migration*, *seed* e *tests*. Para chegar nesse objetivo foi proposta uma arquitetura separada da seguinte forma.

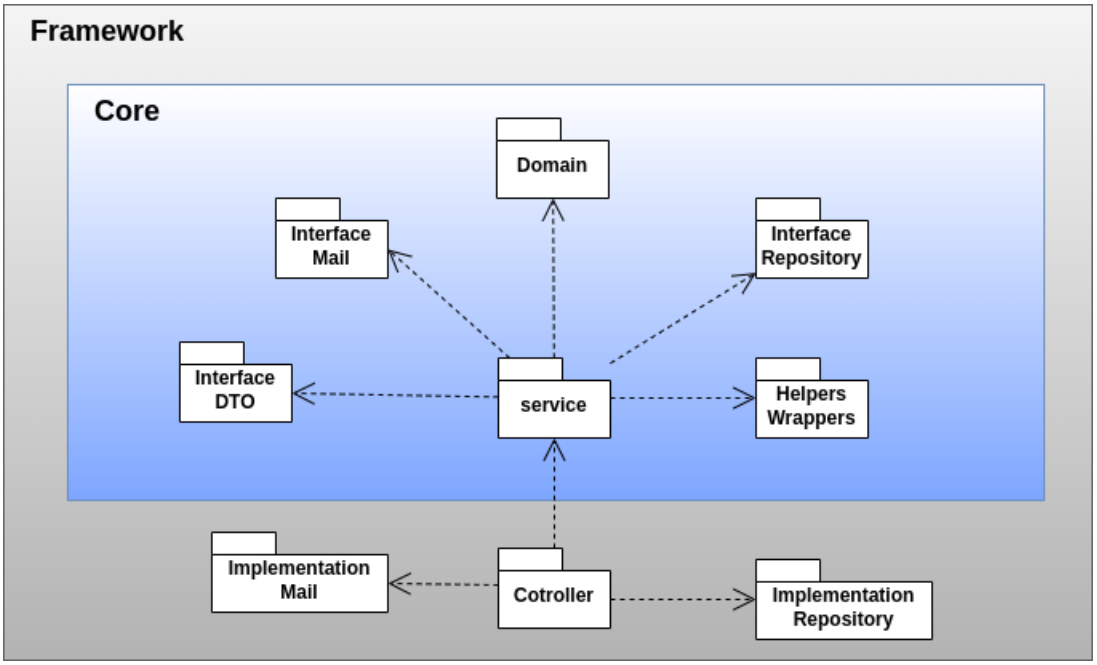

Figura 20. Diagrama de pacotes do *backend*

11.2 *Framework*

Essa camada é onde ficam as abstrações e implementações do *framework* Laravel, onde ele já traz consigo funcionalidades básicas de uma aplicações web, como

autenticação, serialização de dados, acesso a *query parameters*, *headers*, *upload* de arquivos e outros componentes básicos.

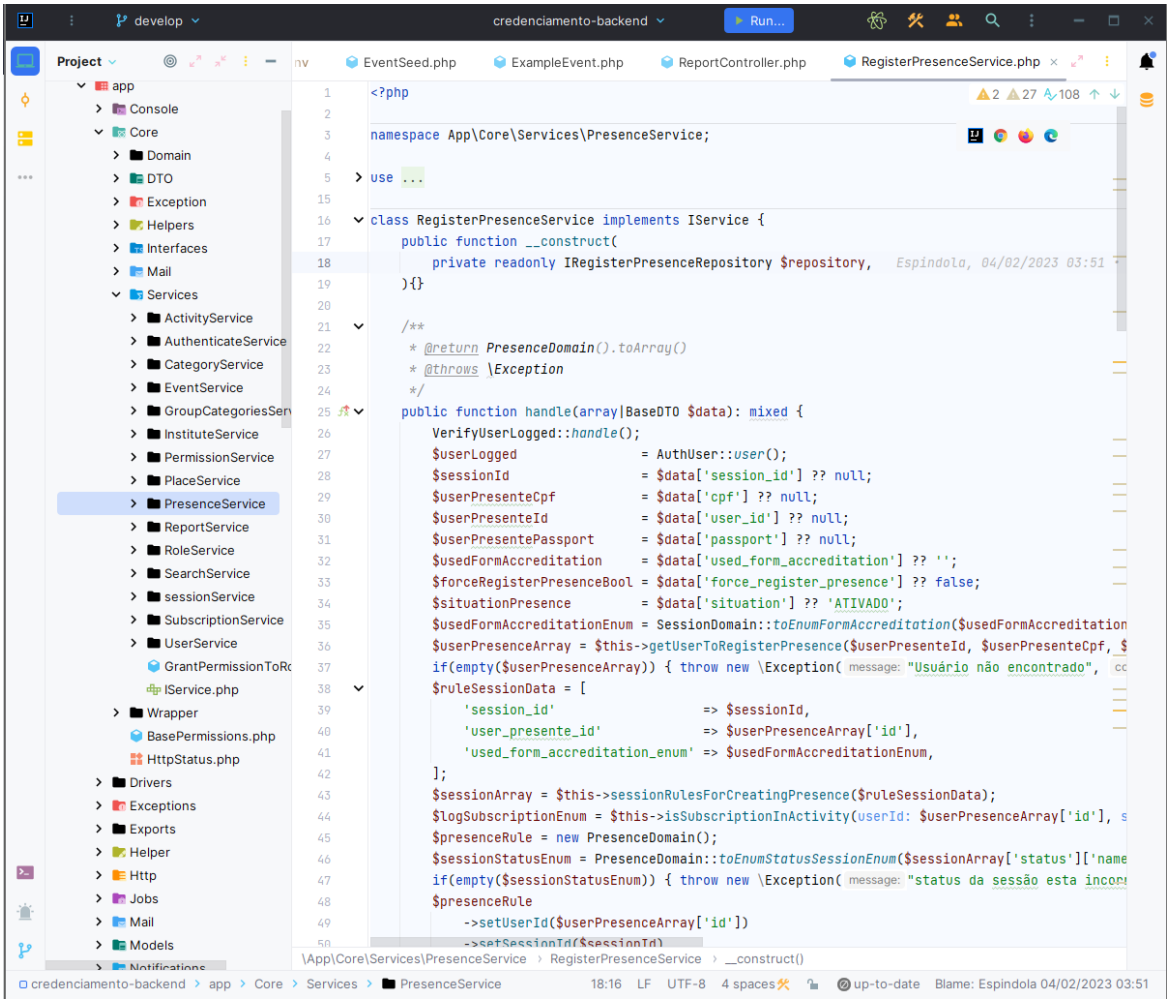

Figura 21. Imagem ilustrativa do código-fonte do *framework*

11.3 *Model*

O ORM (*Object Relational Mapping*) Eloquent proporciona uma abordagem simples e usual para a interação com bancos de dados relacionais. Ele permite a realização de operações de banco de dados por meio da linguagem de programação PHP, eliminando a necessidade de redigir consultas SQL manualmente.

Uma das características primordiais do Eloquent é sua sintaxe intuitiva e acessível. Esse ORM baseia-se no conceito de modelos "*models*" para representar as tabelas do banco de dados como classes em PHP.

11.4 *Interface repository*

Utilizando o conceito de inversão de dependência [Yang, 2008], essas interfaces são acessadas no serviço "*service*" e têm como principal propósito abstrair a interação com o banco de dados. Essa abstração visa a separação da camada de serviço, ela garante que as alterações nas regras de negócios não afetem o banco de dados, e vice-versa.

11.5 *Repository implementation*

Trata-se da implementação concreta das interfaces utilizadas no repositório. Essas implementações são injetadas no momento em que a instância do serviço "*service*" é criada no controlador "*controller*". Dentro desta implementação, está contida toda a lógica de acesso ao banco de dados por meio do ORM. Os dados assim obtidos são direcionados ao serviço, que executará as operações relacionadas às regras de negócio.

11.6 *Service*

Camada que vai executar as operações chamando a regra de negócio e de acesso ao banco de dados, seguindo uma ordem e fluxo estabelecidos, realizando as verificações necessárias e lidando com os possíveis erros que podem acontecer durante o processo.

11.7 *Domain*

Local onde as diretrizes fundamentais do sistema são aplicadas, ou seja, tem como responsabilidade de conter de forma simplista e de preferência indivisível cada uma das regras de negócio. Por exemplo, somente será permitido efetuar uma inscrição se ainda houver vagas disponíveis. Nessa camada, a preocupação não abrange interações com sistemas externos ou elementos relacionados à web somente com a regra de negócio usando o máximo de código puro "sem ferramentas externas".

12 *Frontend*

12.1 Abordagem arquitetural

Como mencionado na seção 10.1, o *frontend* foi separado em duas aplicações Vue.js. Cada uma dessas aplicações desempenha um papel específico: uma possui funcionalidades administrativas dependendo do perfil, como já foi detalhado na seção 5, enquanto a outra é voltada para o público em geral que utiliza a plataforma.

Essa abordagem foi escolhida com base em dois fundamentos: a segurança e a lógica de funcionamento do sistema. Primeiramente, a separação das aplicações foi adotada visando à segurança do sistema como um todo. Ao manter as funcionalidades administrativas e públicas em aplicações distintas, reduz-se a exposição de recursos sensíveis ao público externo, facilitando também o gerenciamento das informações.

12.2 *Views*

Juntamente com suas implementações e toda a parte de design. Foram realizados o controle de estado e dividimos a interface em componentes. É nesta camada que as chamadas aos serviços são feitas para gerenciar os dados que serão exibidos na tela, determinando também o formato em que esses dados serão apresentados.

12.3 *Services*

Esta camada tem a responsabilidade de conectar a interface com o repositório, realizando as conversões necessárias para transmitir apenas os dados essenciais no formato adequado para o *backend*. Além disso, assegura que a autenticação entre o *frontend* e o *backend* esteja estabelecida, garantindo assim a segurança do sistema.

12.4 *Repository*

Esta camada é responsável pela integração entre o *backend* e o *frontend*. Ela utiliza a biblioteca axios para realizar a comunicação com o *backend* utilizando o protocolo HTTP (*HyperText Transfer Protocol*), utilizando as informações necessárias para acessar o *backend*.

13 Implementação de testes na aplicação

13.1 Teste de *backend*

Foram feitos testes automatizados utilizando as ferramentas *PestPHP* e *PHPUnit* de integração em todos os endpoints críticos, passando pela maioria dos casos possíveis em um teste de integração, onde conseguimos 54,95% de cobertura de testes, onde esses testes cobriam desde a requisição backend validando a entrada e os dados, se as operações no banco de dados estavam ocorrendo de forma esperada.

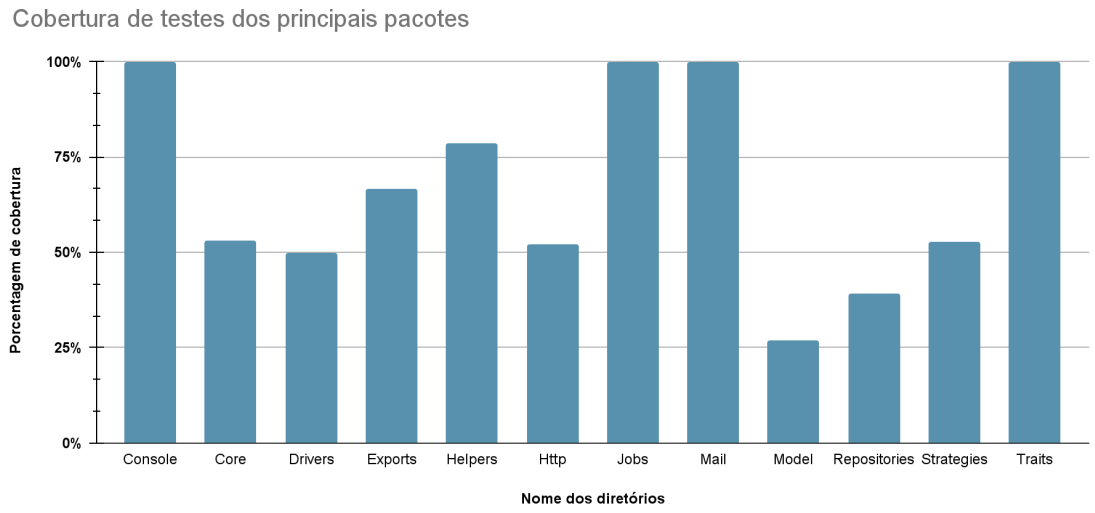

Figura 22. Porcentagem de cobertura de testes por diretório

13.2 Teste de *frontend*

Devido às restrições de tempo e priorização, em conjunto com a complexidade associada à automação de testes no *frontend*, optamos por realizar testes exploratórios. Estes testes se concentraram no comportamento do *frontend* em comunicação com o *backend*. Os testes exploratórios seguiram o roteiro de Abertura da tela/componente a ser testado, realização dos testes seguindo a documentação dos testes unitários do *backend*, em caso de falha, discussão se a falha seria consertada no *frontend* e/ou *backend e* abertura de *pull request* e merge para a branch *main*. Os esforços de teste foram direcionados principalmente para o *backend*, uma vez que é onde as regras estão definidas. O *frontend*, por sua vez, atua principalmente como consumidor de dados e segue as diretrizes estabelecidas pelo *backend.*

14 Limitações e dificuldades

14.1 Limitações e dificuldades encontradas

As principais limitações encontradas são voltadas para integrações com plataformas da UFMS como o SISCERT, SISCAD (Sistema Acadêmico de Graduação da UFMS) e SIGPROJ (Sistema de Informação e Gestão de Projetos). Desde a burocracia exigida para as integrações e possíveis riscos de alterações nas integrações.

Com o decorrer do projeto percebeu-se que a plataforma poderia ser independente da UFMS para ser utilizada em outras instituições públicas de ensino, pois cada instituição tem suas regras e cultura no quesito gestão de eventos mas o sistema foi pensado de forma onde a arquitetura proposta neste projeto suportasse genericamente os eventos.

Não ter um cliente fixo que pudesse acompanhar o processo de desenvolvimento, tornou-se uma grande limitação, pois o feedback durante o desenvolvimento seria útil para validar o avanço do projeto, trazendo segurança. Mas de forma consciente, foram utilizadas estratégias, por exemplo entrevistar possíveis interessados no sistema a fim compensar e mitigar riscos, para que o projeto atenda o problema proposto.

Como cada perfil de usuário exige diferentes perspectivas quando logado no sistema, tivemos dificuldades para gerenciá-las. Mas depois de realizarmos a análise, chegamos à conclusão que a criação de dois *frontends* seria uma estratégia eficiente para dividir esses perfis em dois grupos, um voltado para os participantes, e outro voltado para área administrativa e gestão.

Outra grande limitação encontrada é a adequação a leis que regem o funcionamento de softwares como a LGPD (Lei Geral de Proteção de Dados). Pois a LGPD impactou na análise de requisitos e nas decisões durante a codificação do projeto. Desde o modo como os dados são tratados dentro do sistema e as operações são feitas para remediar possíveis problemas de descumprimento desta lei.

15 Conclusões e perspectivas futuras

15.1 Considerações finais

Este trabalho apresentou a ferramenta web para gestão de eventos com ênfase nas necessidades de instituições públicas de ensino, em especial a UFMS. Desta forma acreditamos sanar as dificuldades relatadas por professores, e melhorar a divulgação dos eventos para assim aumentar o engajamento da comunidade acadêmica, e o registro de presença dos participantes auxiliando no desempenho na emissão de relatório de certificados.

15.2 Trabalhos futuros

A próxima etapa deste trabalho é dar continuidade à implementação dos requisitos restantes que foram coletados e planejados, porém não concluídos em tempo hábil. As funcionalidades e ações que estão nos planos de desenvolvimento:

1) Implementar segunda e terceira camada de funcionalidade;
2) Executar em eventos reais e colher feedback;
3) Fazer correções necessárias do feedback;
4) Integração com sistemas da UFMS como por exemplo o SISCERT.

Para que o sistema possa ser usado em eventos de pequenas e grandes proporções e de diferentes tipos, é necessário implementar demais funcionalidades, tanto para atender totalmente as diferentes demandas, quanto para melhorar sua usabilidade e acessibilidade. Desta forma, esta solução poderá vingar e ter uma boa longevidade, atendendo problemas reais e efetivamente melhorando a UFMS.

16 Referências

Matthew D. Beckman, Mine Çetinkaya-Rundel, Nicholas J. Horton, Colin W. Rundel, Adam J. Sullivan & Maria Tackett (2021) Implementing Version Control With Git and GitHub as a Learning Objective in Statistics and Data Science Courses, Journal of Statistics and Data Science Education, 29:sup1, S132-S144, DOI: 10.1080/10691898.2020.1848485

Mudholkar, Megha & Mudholkar, Pankaj. (2017). A Beginner's Guide to Git and GitHub. 10.13140/RG.2.2.20126.89927.

Coad, peter; Luca, de jeff; Lefebvre eric. Java Modeling in Color with UML: Enterprise Components and Process. Prentice Hall PTR, 1999.

M. Raglianti, C. Nagy, R. Minelli and M. Lanza, DiscOrDance: Visualizing Software Developers Communities on Discord, 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), Limassol, Cyprus, 2022, pp. 474-478, doi: 10.1109/ICSME55016.2022.00062.

S. Nakazawa and T. Tanaka, Development and Application of Kanban Tool Visualizing the Work in Progress, 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, Japan, 2016, pp. 908-913, doi: 10.1109/IIAI-AAI.2016.156.

IVO, A. A.; MARIN, E. C.; SOUZA, L. M. D. GESTÃO DE EVENTOS: ORIENTAÇÕES BÁSICAS PARA O CONTEXTO DAS UNIVERSIDADES. Kinesis, *[S. l.]*, v. 32, n. 2, 2014. DOI: 10.5902/2316546416508. Disponível em: https://periodicos.ufsm.br/kinesis/article/view/16508. Acesso em: 11 nov. 2023.

Laravel Development Team. Documentação Oficial do Laravel 9.x. Disponível em: https://laravel.com/docs/9.x. Acesso em: 11. nov. 2023.

Vue Development Team. Guia Oficial do Vue.js 2. Disponível em: https://v2.vuejs.org/v2/guide/. Acesso em: 11. nov. 2023.

Axios Development Team. Documentação Oficial do Axios. Disponível em: https://axios-http.com/docs/intro. Acesso em: 11. nov. 2023.

Vuetify Development Team. Vuetify Documentation. Disponível em: https://v2.vuetifyjs.com/en. Acesso em: 11. nov. 2023.

UFMS - Universidade Federal de Mato Grosso do Sul. Portal de Estatísticas. Disponível em: https://numeros.ufms.br/. Acesso em: 11.nov.2023.

MILI, A; MILI, B; MITTERMEIR, R. T. A survey of software reuse libraries. J. C. Baltzer AG**,** Science Publishers. v. 5, n. 1, p. 349-414, jan. 1998.

BATORY, D; SINGHAL, V; SIRKIN, M; THOMAS, J. Scalable Software Libraries. Association for Computing Machinery. v. 18, n.5, p. 191-199, dec. 1993.

M. O. G. Ramírez, M. De-la-Torre and C. Monsalve, Methodologies for the design of application frameworks: systematic review, *2019 8th International Conference On Software Process Improvement (CIMPS)*, Leon, Mexico, 2019, pp. 1-10, doi: 10.1109/CIMPS49236.2019.9082427

Garrido, J. M. Hexagonal Architecture. Disponível em: https://jmgarridopaz.github.io/content/hexagonalarchitecture.html. Acesso em: 11. nov. 2023.

Cockburn, A. Hexagonal Architecture. Disponível em: https://alistair.cockburn.us/hexagonal-architecture/. Acesso em: 11. nov. 2023

A. Contan, C. Dehelean and L. Miclea, "Test automation pyramid from theory to practice," 2018 IEEE International Conference on Automation, Quality and Testing,

Robotics (AQTR), Cluj-Napoca, Romania, 2018, pp. 1-5, doi: 10.1109/AQTR.2018.8402699.

R. Muddinagiri, S. Ambavane and S. Bayas, "Self-Hosted Kubernetes: Deploying Docker Containers Locally With Minikube," 2019 International Conference on Innovative Trends and Advances in Engineering and Technology (ICITAET), Shegoaon, India, 2019, pp. 239-243, doi: 10.1109/ICITAET47105.2019.9170208.

H. Y. Yang, E. Tempero and H. Melton, "An Empirical Study into Use of Dependency Injection in Java," 19th Australian Conference on Software Engineering (aswec 2008), Perth, WA, Australia, 2008, pp. 239-247, doi: 10.1109/ASWEC.2008.4483212.

T. Kinsman, M. Wessel, M. A. Gerosa and C. Treude, "How Do Software Developers Use GitHub Actions to Automate Their Workflows?," 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), Madrid, Spain, 2021, pp. 420-431, doi: 10.1109/MSR52588.2021.00054.

C. Singh, N. S. Gaba, M. Kaur and B. Kaur, "Comparison of Different CI/CD Tools Integrated with Cloud Platform," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2019, pp. 7-12, doi: 10.1109/CONFLUENCE.2019.8776985.

W. Khan, W. Ahmad, B. Luo and E. Ahmed, "SQL Database with physical database tuning technique and NoSQL graph database comparisons," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 110-116, doi: 10.1109/ITNEC.2019.8729264.

M. Hu and H. Li, "Application of Location-Controlled Mobile Attendance Recording System in College Classroom Teaching," *2020 International Symposium on Educational Technology (ISET)*, Bangkok, Thailand, 2020, pp. 18-22, doi: 10.1109/ISET49818.2020.00014.