

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Amanda de Souza Caetano

**Sistema Web Para Gerenciamento de Eventos e
Emissão de Documentos - SKED**

Uberlândia, Brasil

2019

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Amanda de Souza Caetano

**Sistema Web Para Gerenciamento de Eventos e Emissão
de Documentos - SKED**

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

Orientador: Prof. Dr. Rafael Dias Araújo

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2019

Amanda de Souza Caetano

Sistema Web Para Gerenciamento de Eventos e Emissão de Documentos - SKED

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

Trabalho aprovado. Uberlândia, Brasil, 29 de novembro de 2019:

Prof. Dr. Rafael Dias Araújo
Orientador

Prof. Dr. Fabiano Azevedo Dorça
FACOM/UFU

Prof. Dr. Renan Gonçalves Cattelan
FACOM/UFU

Uberlândia, Brasil
2019

Aos meus pais Geraldo e Sandra.

Agradecimentos

Agradeço primeiramente a Deus, por estar sempre caminhando ao meu lado.

Agradeço à minha família, minha maior dádiva. Aos meus pais, avós, irmã e sobrinha que sempre foram meu maior apoio e incentivo; à minha namorada Juliana e aos meus amigos Júlia, Juliano e Eduardo pelas forças que me dão para continuar a seguir meus sonhos, assim como Daniel, Nicolas e Rafael Matias que nunca mediram esforços para me ajudar, sanar minhas dúvidas e me mostrar novos caminhos.

Agradeço ao meu orientador, Prof^o. Rafael Dias Araújo, que com exímio me guiou por essa pesquisa e compartilhou dos seus conhecimentos comigo de forma paciente, compreensiva e carinhosa. Aos professores, coordenadores, secretário e demais funcionários da FACOM/UFU que fizeram possível toda minha jornada pela graduação.

Agradeço a todos que me apoiaram e compartilharam amor, conhecimento e carinho no trajeto para a consumação desse objetivo. Ofereço a vocês toda minha gratidão.

"That brain of mine is something more than merely mortal; as time will show."

(Ada Lovelace)

Resumo

Quando empresas e agências de eventos usam múltiplas plataformas para cadastrar eventos, gerar documentos e ter controle de agenda, uma série de complicações podem surgir, como atrasos, informações incompatíveis e dificuldade de manutenção dos dados. Dessa forma, propõe-se o sistema Web SKED, um sistema simples para auxiliar no gerenciamento de eventos e geração de documentos. Tal sistema proporcionará ao usuário cadastrar e manipular dados de eventos, visualizá-los em uma agenda preenchida automaticamente, confeccionar e gerar documentos usando os dados já cadastrados. Para entender o problema e definir os propósitos do sistema, um processo de levantamento de requisitos e de modelagem de dados foi executado. Além disso, com o objetivo de tornar o sistema fácil e confortável de usar, um protótipo foi criado e testado por usuários diversos, e os dados de tal teste foram usados para guiar a implementação do sistema, composto por aplicação Web, API REST e banco de dados.

Palavras-chave: Sistema Web, Gerenciador de Eventos, Avaliação de Usabilidade.

Lista de ilustrações

Figura 2.1.1 – Níveis de fidelidade de um protótipo.	20
Figura 2.1.2 – Resultado de renderização de código HTML e CSS.	22
Figura 2.1.3 – Ciclo de vida do evento.	23
Figura 3.1.1 – Diagrama de Casos de Uso	28
Figura 3.1.2 – Diagrama de Atividades do Caso de Uso 1: Cadastro de Usuários. . .	30
Figura 3.2.1 – Fragmento do DER do banco de dados.	31
Figura 3.3.1 – Página de listagem de propostas.	33
Figura 3.3.2 – Formulário de dados do cliente da nova proposta.	34
Figura 3.3.3 – Formulário de informações do serviço da nova proposta.	34
Figura 3.3.4 – Confirmação de dados preenchidos no formulário.	35
Figura 3.3.5 – Modal de confirmação de exclusão de dados.	36
Figura 3.3.6 – Tela de finalização do fluxo de cadastrar nova proposta.	36
Figura 3.4.1 – Fluxograma de etapas para execução do teste de usabilidade.	37
Figura 3.4.2 – Planta da sala de aplicação de testes.	37
Figura 3.4.3 – Representação gráfica sobre atuação profissional dos usuários que realizaram o teste.	38
Figura 3.4.4 – HTA das tarefas para teste do protótipo.	39
Figura 3.4.5 – Distribuição de nível de dificuldade que os usuários tiveram ao usar o sistema.	40
Figura 3.4.6 – Distribuição de opiniões dos usuários sobre ser possível usar o sistema sem treinamento prévio.	40
Figura 3.4.7 – Distribuição de opiniões dos usuários sobre se seria satisfatório usar o sistema frequentemente.	41
Figura 3.4.8 – Distribuição de classificação de dificuldade em executar tarefa. . . .	41
Figura 3.4.9 – Sugestões dadas pelos usuários sobre quais funcionalidades precisam ser melhoradas.	42
Figura 3.4.10 – Novo <i>layout</i> de tela de confirmação.	43
Figura 3.4.11 – Novo <i>layout</i> de modal para confirmar exclusão de proposta.	43
Figura 3.4.12 – Novo <i>layout</i> de modal para informar a finalização do processo de criação da proposta.	44
Figura 3.4.13 – Média de tempo para executar cada tarefa.	44
Figura 3.4.14 – Novo formulário para gerenciar datas da proposta.	45
Figura 3.4.15 – Tempo de duração para execução dos testes individualmente e média. .	46
Figura 3.5.1 – Fluxograma do sistema completo.	47
Figura 3.5.2 – Fluxograma de funcionamento da API.	48

Figura 3.5.3 – Mapa da aplicação Web.	52
Figura 3.5.4 – Fluxograma do funcionamento da aplicação web.	53
Figura 3.5.5 – Tabela com informações e menu da proposta.	53
Figura 3.5.6 – Exibição de detalhes da proposta.	55
Figura 3.6.1 – Tela de login.	56
Figura 3.6.2 – Áreas de cadastro e recuperação de senha.	57
Figura 3.6.3 – Página de <i>home</i>	57
Figura 3.6.4 – Área de propostas.	58
Figura 3.6.5 – Tela de gerenciamento e aprovação de usuários.	58
Figura 3.6.6 – Área de propostas.	59
Figura 3.6.7 – Gerenciamento de Usuário: usuários pendentes de aprovação.	60
Figura 3.6.8 – Agenda de eventos.	61
Figura I.0.1 – Formulário 1: Informações do Serviço.	83
Figura I.0.2 – Formulário 2: Dados do Cliente.	83
Figura I.0.3 – Formulário 3: Datas.	84
Figura I.0.4 – Formulário 4: Locação de Espaço.	84

Lista de tabelas

Tabela 1 – Descrição do Caso de Uso 1: Cadastrar Novo Funcionário.	29
Tabela 2 – Dados coletados sobre a execução da tarefa 6.	42
Tabela 3 – Relação entre implementação e requisitos.	60

Lista de abreviaturas e siglas

ABEOC	Associação Brasileira de Empresas de Eventos
API	<i>Application Programming Interface</i>
CSS	<i>Cascading Style Sheets</i>
DER	Diagrama Entidade-Relacionamento
HTA	<i>Hierarchical Task Analysis</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
JSON	<i>JavaScript Object Notation</i>
MVP	<i>Model-view-presenter</i>
ORM	<i>Object-relational Mapping</i>
PIB	Produto Interno Bruto
REST	<i>Representational State Transfer</i>
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SQL	<i>Structured Query Language</i>
URI	<i>Uniform Resource Identifier</i>
XP	<i>eXtreme Programming</i>

Sumário

1	INTRODUÇÃO	13
1.1	Objetivos	14
1.1.1	Objetivos Gerais	14
1.1.2	Objetivos Específicos	14
1.2	Método	14
1.3	Contribuições	15
1.4	Organização do Texto	15
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Conceitos Básicos	17
2.1.1	Engenharia de Software	17
2.1.2	Método Ágil	17
2.1.3	Engenharia de Requisitos	18
2.1.4	Banco de Dados Relacional	18
2.1.5	Usabilidade	19
2.1.6	Protótipos	19
2.1.7	API REST	20
2.1.8	Aplicação Web	21
2.1.9	Eventos	23
2.2	Trabalhos Relacionados	24
3	DESENVOLVIMENTO	25
3.1	Levantamento de Requisitos	25
3.2	Modelagem do Banco de Dados	27
3.3	Protótipo	32
3.4	Testes e Análises de Usabilidade	36
3.5	Implementação	46
3.5.1	Back-End	47
3.5.2	Front-End	51
3.6	Interface	54
4	CONCLUSÃO	62
4.1	Desafios Encontrados	62
4.2	Trabalhos Futuros	63
	REFERÊNCIAS	64

APÊNDICES	66
APÊNDICE A – TESTE DE USABILIDADE: TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO	67
APÊNDICE B – TESTE DE USABILIDADE: INSTRUÇÕES E INFORMAÇÕES IMPORTANTES	68
APÊNDICE C – TESTE DE USABILIDADE: CENÁRIO E TAREFAS	69
APÊNDICE D – TESTE DE USABILIDADE: QUESTIONÁRIO DEMOGRÁFICO	70
APÊNDICE E – TESTE DE USABILIDADE: TESTE DE USABILIDADE E AVALIAÇÃO DAS TAREFAS	72
APÊNDICE F – TESTE DE USABILIDADE: FICHA DE OBSERVAÇÕES	74
APÊNDICE G – DIAGRAMA ENTIDADE-RELACIONAMENTO .	76
APÊNDICE H – DOCUMENTAÇÃO DA API	77
APÊNDICE I – APRESENTAÇÃO DE FORMULÁRIOS IMPLEMENTADOS	83

1 Introdução

O mercado de eventos sociais é muito presente na movimentação da economia brasileira, representando mais de 4% do PIB nacional, de acordo com o II Dimensionamento Econômico da Indústria de Eventos do Brasil, realizado pela ABEOC¹ e Sebrae² em 2013 (ABEOC; SEBRAE, 2013). Além disso, foi apontado que o número de empresas organizadoras e agências de eventos teve um aumento de 596% entre 2001 e 2013. Recentemente, o Instituto Locomotiva apresentou uma pesquisa indicando que o faturamento do mercado de eventos foi de \$17 bilhões em 2016 e 2015 (Revista Eventos, 2017).

Para planejar e realizar um evento é necessário que as empresas e agências responsáveis por sua organização manipulem várias informações, como dados de clientes, produtos, serviços e espaços oferecidos, entre outras referências necessárias para o planejamento e execução do serviço contratado. Ademais, a preparação de um evento deve prever eventuais problemas como: conflito de eventos na mesma data e local; funcionários indisponíveis para a realização do evento; produtos insuficientes, dentre outros transtornos que possam gerar grande insatisfação para o cliente. Assim, a disposição e manutenção de todas essas informações se fazem extremamente necessárias para um funcionamento efetivo e bem-sucedido da empresa.

Sendo assim, é indispensável que haja um controle e organização sobre as informações citadas, por meio de funcionalidades para organizar e sincronizar agendas, espaços e manipulação de serviços e produtos oferecidos, bem como, simulação de orçamentos, para que seja possível gerar propostas, contratos e outros documentos para comunicação e formalização dos trâmites juntamente ao cliente. No entanto, para atender tais necessidades no exercício de organizar um evento social, é comum a utilização de múltiplas aplicações de software, sendo que cada aplicação efetua uma tarefa diferente, porém compartilham um conjunto de informações de forma não integrada, o que gera uma grande dificuldade de manutenção e garantia de veracidade/consistência dos dados.

Com o propósito de automatizar e unificar a administração de eventos, propõe-se neste trabalho o desenvolvimento de um sistema para gerenciamento de eventos e geração de documentos necessários para a realização do serviço e comunicação com o cliente, de forma a reduzir a sobrecarga do funcionário, a dificuldade de organização e a utilização de múltiplas aplicações. Assim, preza-se pela possibilidade de englobar todas as funcionalidades essenciais no fluxo de um evento, desde a simulação de orçamento até a construção e efetivação do contrato firmado entre as partes.

¹ Associação Brasileira de Empresas de Eventos

² Serviço Brasileiro de Apoio às Micro e Pequenas Empresas

Apresenta-se, então, o sistema Web SKED³ Gerenciamento de Eventos, cujo objetivo é auxiliar empresas e agências de eventos a otimizar suas atividades na organização e planejamento de um evento, unindo tarefas como cadastro de eventos, agenda e geração de documentos, de forma simples e mantendo aspectos mínimos da segurança da informação, por meio de login e permissionamento de usuários de acordo com cada perfil.

1.1 Objetivos

1.1.1 Objetivos Gerais

Este projeto de graduação tem como objetivo geral o desenvolvimento de um Sistema Web, que atenda às necessidades essenciais de empresas e agências de eventos para com a organização, administração e planejamento de seus eventos de forma centralizada em uma só ferramenta. Podendo ser utilizado por qualquer empresa da área, o sistema usará login e permissionamento de usuários para manter aspectos mínimos de segurança das informações, assim como as melhores práticas de usabilidade em sistemas Web para agregar facilidade ao se usar a ferramenta.

1.1.2 Objetivos Específicos

Como objetivos específicos deste projeto de graduação apresenta-se:

- Estudar as necessidades das empresas e agências de eventos, assim como suas atuais estratégias e métodos para suprir tais indispensabilidades;
- Estudo de tecnologias e *frameworks* que atendam ao objetivo geral proposto;
- Especificar e desenvolver *back-end*, *front-end* e banco de dados para um sistema Web para gerenciamento de eventos;
- Pesquisar e estudar boas práticas e princípios de usabilidade para sistemas Web;
- Comparar e analisar tempo de execução dos fluxos de eventos sem a ferramenta e com a ferramenta.

1.2 Método

Com a intenção de uma organização e planejamento do trabalho mais efetivas do trabalho, foram aplicadas técnicas do desenvolvimento ágil, especificadamente do *Kanban*⁴, que, segundo [Gomes \(2010\)](#) tem como objetivo apresentar a evolução em forma

³ *sked*: fonética que lembra o substantivo *schedule*, do inglês, que significa “cronograma; atividades planejadas ou coisas a serem feitas”.

⁴ Palavra de origem japonesa que significa “cartão” ou “sinalização”.

visual, evidenciar os problemas e manter uma cultura de melhoria contínua. Para aplicar esse método, foi utilizado o aplicativo de gerenciamento Trello⁵.

Para melhor entendimento do contexto, foram feitas reuniões e visitas à empresas e agências de eventos sociais, utilizando métodos de conversação com os funcionários e observação do fluxo existente de agendamento de eventos. Dessa forma, observou-se as necessidades e foram decididos e documentados os principais requisitos para o sistema.

Para mensurar a usabilidade do sistema foi desenvolvido e analisado um protótipo de alta fidelidade, ilustrando um dos seus fluxos principais. Com o objetivo de aprimorar a interface e trazer melhor experiência para o usuário, o protótipo exemplifica uma interface que considera boas práticas de interação entre humanos e computadores, particularmente, as heurísticas de usabilidade definidas por Nielsen (1993).

O protótipo foi, então, usado na aplicação de testes de usabilidade, com o objetivo de coletar e analisar dados sobre a usabilidade do sistema, usando as análises feitas para tornar o sistema mais simples e melhorar a experiência do usuário. Para a aplicação dos testes, usuários voluntários seguiram instruções para realizar algumas atividades no protótipo enquanto eram observados por instrutores. Posteriormente, os participantes responderam questionários sobre o teste.

A próxima etapa foi pesquisar, comparar e escolher as ferramentas e linguagens de programação usadas para o desenvolvimento do projeto. Foram escolhidas as linguagens de programação HTML 5, CSS 3, Typescript e Javascript. No gerenciamento do banco de dados foi usado MySQL. Também houve a utilização de *frameworks*, como Angular 8 e NodeJs. Para a arquitetura de software foram usados os modelos REST (*Representational State Transfer*) e MVP (*Model-View-Presenter*).

1.3 Contribuições

Espera-se como contribuições do projeto, um sistema Web para auxiliar no agendamento de eventos e emissão de documentos, seguindo os princípios usabilidade de Nielsen (1999), resultando em uma aplicação de fácil uso.

1.4 Organização do Texto

Este documento está organizado da seguinte forma: o Capítulo 2 apresenta fundamentações teóricas sobre assuntos que constroem uma base para o desenvolvimento e entendimento do trabalho, sendo eles, Engenharia de Software, Metodologia Ágil, Engenharia de Requisitos, Usabilidade, Prototipação, API (*Application Programming Inter-*

⁵ <https://trello.com>

face) REST, Aplicações Web e Eventos, bem como a apresentação de trabalhos similares; no Capítulo 3, as etapas do desenvolvimento do projeto são descritas, sendo elas: entendimento do tema, modelagem dos dados, prototipação, testes e análise de usabilidade, desenvolvimento da aplicação e apresentação da interface implementada; por fim, no Capítulo 4, são apresentadas as considerações finais acerca do trabalho, que inclui as dificuldades encontradas e possibilidades de trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos básicos necessários para compreensão do trabalho, bem como os trabalhos relacionados.

2.1 Conceitos Básicos

2.1.1 Engenharia de Software

A Engenharia de Software é definida por [IEEE \(1990\)](#) como “aplicação de uma abordagem sistemática, disciplinada e quantificável, para desenvolvimento, operação e manutenção do software”. Durante o desenvolvimento deste projeto, prezou-se pela utilização de princípios da Engenharia de Software para manter uma melhor organização e planejamento.

Segundo [Pressman \(2011\)](#), a aplicação da Engenharia de Software envolve cinco atividades genéricas: comunicação, planejamento, modelagem, construção e emprego. Há diferentes métodos que definem a ordem de execução dessas tarefas e suas especificidades. Nesse projeto, optou-se pelo Método Ágil para auxiliar na aplicação da Engenharia de Software.

2.1.2 Método Ágil

O Método Ágil surgiu com o Manifesto Ágil em 2001 ([BECK et al., 2001](#)), publicado por 17 autores, no intuito de padronizar uma metodologia ágil comum para o desenvolvimento de software. Suas principais diretrizes envolvem priorizar indivíduos, desenvolvimento de software, colaboração do cliente e boas respostas a mudanças. Os métodos ágeis trazem diversos benefícios, como melhor colaboração e comunicação entre os participantes do projeto e melhoria na qualidade do produto, sendo esses os principais motivadores para o uso do método nesse projeto.

Existem várias metodologias ágeis de desenvolvimento de software ([HAZZAN; DUBINSKY, 2009](#)), como *Scrum* ¹, *Kanban* ([HAMMARBERG; SUNDEN, 2014](#)), *eXtreme Programming (XP)* ². Neste projeto, foi utilizado o *Kanban* ³, que, segundo GOMES ([GOMES, 2010](#)) tem como objetivo apresentar a evolução em forma visual, evidenciar os problemas e manter uma cultura de melhoria contínua. Além disso, existem várias ferramentas que podem ser utilizadas para auxiliar na aplicação dos métodos ágeis, como Trello

¹ <https://www.scrum.org/>

² <http://www.extremeprogramming.org/>

³ Palavra de origem japonesa que significa “cartão” ou “sinalização”.

⁴, Asana ⁵ e Jira Software ⁶. Neste projeto, foi utilizada o aplicativo de gerenciamento Trello para dar suporte ao gerenciamento das tarefas.

2.1.3 Engenharia de Requisitos

A Engenharia de Requisitos é um dos processos da Engenharia de Software, que consiste em estudar e entender o problema antes de iniciar as atividades técnicas do desenvolvimento. É importante pois, através dela, é possível alcançar um melhor entendimento do problema, visar com clareza quais as vontades e objetivos do cliente e, assim, ter mais assertividade em definir como o software em desenvolvimento poderá auxiliar o cliente na solução dos problemas e no alcance de suas metas para o produto final.

Dividida em concepção, levantamento, elaboração, negociação, especificação, validação e gestão, a Engenharia de Requisitos é fundamental para o entendimento do que o cliente almeja com a construção do software em questão. A atividade de Levantamento de Requisitos, que combina elaboração, negociação e especificação, foi executada junto ao potencial cliente da solução. Essa atividade inclui o cliente, desenvolvedores, engenheiros de software e outros interessados em trabalho em equipe para detectar problemas, propor soluções e negociar possíveis abordagens. Além disso, é definido requisitos preliminares do software em questão (PRESSMAN, 2011).

2.1.4 Banco de Dados Relacional

Um banco de dados relacional usa um modelo simples para armazenar e manipular dados, que consiste em tabelas de duas dimensões, denominadas de “relação”. Tais relações representam os dados usando três princípios: estrutura do dado, suas operações e suas restrições (GARCIA-MOLINA; ULLMAN; WIDOM, 2011).

Uma forma de descrever dados para representá-los em um modelo relacional é usar linguagens de alto nível dedicadas à isso, como por exemplo a SQL (*Structured Query Language*). A SQL permite representar os dados e manipulá-los por meio de *queries*, um comando para manipular os dados, que pode ditar ações como ler, criar, atualizar, deletar dados, entre outros.

Há também *softwares* que viabilizam o uso do SQL para implementar bancos de dados relacionais, como o MySQL⁷: um Sistema Gerenciador de Banco de Dados relacionais que, dentre uma vasta lista de aplicações e usabilidades, permite criar e manipular bancos de dados relacionais de forma simples.

⁴ <https://trello.com/>

⁵ <https://asana.com/>

⁶ <https://www.atlassian.com/software/jira>

⁷ <https://www.mysql.com>

2.1.5 Usabilidade

A usabilidade é, segundo Nielsen (2012), um atributo qualitativo que avalia a facilidade de uso de uma interface, como também é uma forma de aprimorar tal facilidade durante o processo de criação de uma aplicação.

Nielsen cita, ainda, que a boa usabilidade é uma necessidade essencial em aplicações Web e quando um usuário tem dificuldade em usar tais aplicações ele tende a abandoná-la. Dessa forma, testar, avaliar e classificar a usabilidade de uma aplicação pode ser essencial para torná-la melhor e garantir maior satisfação do usuário.

Existem várias técnicas para avaliar a usabilidade de um software. Uma delas é por meio de testes qualitativos aplicados no sistema ou em um protótipo. Uma das formas de execução desses testes é por meio da execução de tarefas pré-definidas, realizadas por um potencial usuário final do sistema, que avalia sua experiência com o sistema.

Para extrair os dados dos testes a fim de analisá-los, é necessário utilizar princípios da Interação Homem-Máquina, como técnicas de observação ou de perguntas à usuários (DIX et al., 2003). Os métodos observacionais são as estratégias onde o instrutor, que está aplicando o teste, extrai informações observando como o usuário realiza o teste. Dentre esses métodos, temos o *think aloud*, onde o usuário é instruído a verbalizar seus pensamentos enquanto executa o teste. Já nos métodos de perguntas, o instrutor pode conduzir entrevistas ou questionários, sendo uma forma mais direta de extração de dados à partir do usuário.

2.1.6 Protótipos

Um protótipo é uma aplicação experimental para planejar, simular, testar, avaliar e adaptar características da aplicação pretendida antes ou durante o processo de desenvolvimento da aplicação, de modo rápido e mais barato. Ele permite exemplificar as características da aplicação, como o *design* da interface e sequências de operações necessárias para concluir uma ação.

Os protótipos podem ter diferentes níveis de fidelidade à aplicação que está representando (Figura 2.1.1). Um protótipo de baixa fidelidade é feito no papel, podendo ser confeccionado rapidamente e ser uma forma de reunir ideias e sugestões durante uma reunião com clientes, profissionais ou outros interessados no produto. Já o protótipo de média fidelidade tem como foco o fluxo do usuário entre as telas, e é feito em forma de *wireframes*⁸. Por sua vez, os protótipos de alta fidelidade são os que mais se aproximam do sistema real, tanto em layout como comportamento. São uma representação que podem exemplificar fluxo, design e comportamentos do sistema real, de forma limitada.

⁸ <https://www.usability.gov/how-to-and-tools/methods/wireframing.html>

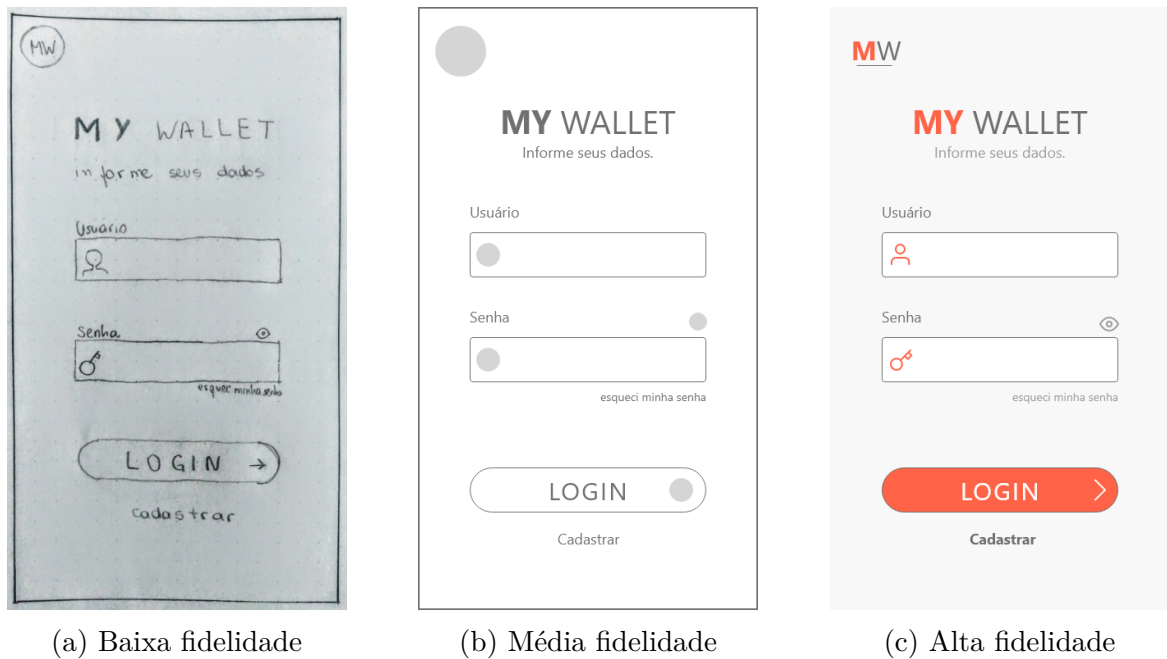


Figura 2.1.1 – Níveis de fidelidade de um protótipo.

2.1.7 API REST

Uma API, do inglês *Application Programming Interface*, é uma aplicação que provê serviços para que outras aplicações possam usa-los. Segundo [Jacobson, Brail e Woods \(2011\)](#), API's são uma forma de comunicação entre duas aplicações, onde a API fornece um elo pra outras aplicações acessar serviços e dados de forma rápida. Tal comunicação ocorre por meio de uma linguagem de comum entendimento entre as partes, onde uma aplicação, como, por exemplo, um *Website*, pode requisitar os recursos de uma API para efetuar um serviço sem a necessidade de entender como funciona o processo de execução.

Uma forma de implementar API's de modo a representar um servidor Web é com o uso da arquitetura REST (*Representational State Transfer*). Tal arquitetura consiste em princípios, regras e limitações para aplicações Web. Quando uma API é implementada sobre a arquitetura REST, é dito que tal API é RESTful ([MASSE, 2011](#)).

Uma das regras impostas pela arquitetura REST é o uso do URI (*Uniform Resource Identifiers*), que é uma cadeia de caracteres que identifica um recurso que faz parte da API. O URI permite que a comunicação entre uma aplicação externa e a API seja possível, pois tem um formato padronizado que executa o papel de linguagem em comum entre as aplicações. Tal formato, definido pela documentação RFC 3986 ([BERNERS-LEE; FIELDING; MASINTER, 2005](#)), tem a seguinte estrutura:

$$\text{URI} = \text{scheme} \text{ "://" } \text{authority} \text{ "/" } \text{path} \text{ ["?" } \text{query} \text{] ["#" } \text{fragment} \text{]}$$

Por exemplo, a URI a seguir pode ser um identificador para um serviço que,

manipulando um banco de dados sobre animais, busca as informações dos cachorros da raça *Poodle* presentes no banco de dados. O *path* “cachorro/” indica sobre qual modelo de dados a ação precisa ser executada, e a *query* “raca=poodle” serve para informar qual raça é objeto de interesse para a requisição citada.

`http://tudosobreanimais/cachorro/?raca=poodle`

Para implementar uma API REST há várias ferramentas, linguagens e tecnologias. Nesse projeto foi usado o `node.js`, um ambiente de execução montado sobre o *Google V8 Javascript Engine*⁹, motor JavaScript¹⁰ do Google Chrome. É um ambiente assíncrono e orientado a eventos que permite o desenvolvimento de aplicações escaláveis implementando fluxos não bloqueantes. Assim sendo, se torna ideal para sistemas distribuídos que trocam dados em tempo real, como uma API e uma aplicação Web.

2.1.8 Aplicação Web

Uma aplicação Web é um *software*, executado em um servidor Web por meio de requisições advindas de um *browser*, que utiliza de tecnologias Web para atuar no papel de cliente de um sistema cliente-servidor. É encarregado da parte visual e lógica do cliente, sendo também quem conduz a comunicação entre o cliente e o servidor.

Tais aplicações são desenvolvidas através do uso de algumas tecnologias comuns, como o protocolo de comunicação *Hypertext Transfer Protocol* (HTTP), a *Hypertext Markup Language* (HTML), a *Cascading Style Sheets* (CSS) e *script languages*.

O HTTP é um protocolo cliente-servidor que define um padrão para que o cliente efetue requisições de recursos ao servidor. Para Casteleyn et al. (2009), o HTTP é componente básico no qual a Web é fundada. Formada por método, identificador e versão, uma requisição HTTP contém as informações necessárias para que o servidor identifique, manipule e disponibilize ao cliente dados e recursos.

Já o HTML, é uma linguagem que descreve a estrutura de uma página Web, usando elementos representados por *tags*, que são usadas pelos *browsers* para renderizar o conteúdo da página, como apontado na plataforma W3C (W3C,). Sua versão atual é a 5, que apresenta um conjunto maior de tecnologias, elementos e recursos.

Para dar estilo às informações apresentadas pelo HTML, faz-se o uso da folha de estilos CSS, uma forma de adicionar características como cor, tamanho e posicionamento aos elementos que compõem uma página web.

O HTML, junto ao CSS, permite exibir ao cliente informações textuais, imagens, tabelas, gráficos e várias outras formas de informação multimídia. Além da exibição,

⁹ <https://v8.dev/>

¹⁰ <https://www.javascript.com>

também é possível manipular dados, como acontece no uso de formulários, por exemplo. No trecho de código 2.1 é mostrado um exemplo de código HTML e CSS e na Figura 2.1.2 tem-se uma exemplificação do resultado final da renderização do mesmo.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Fundamentação Teórica</title>

    <style>
      h1 {
        color: purple
      }
    </style>
  </head>
  <body>
    <h1>HTML e CSS</h1>
    <p>Esta página web foi construída usando HTML e CSS.</p>
  </body>
</html>
```

Código Fonte 2.1 – Exemplo de código HTML e CSS.

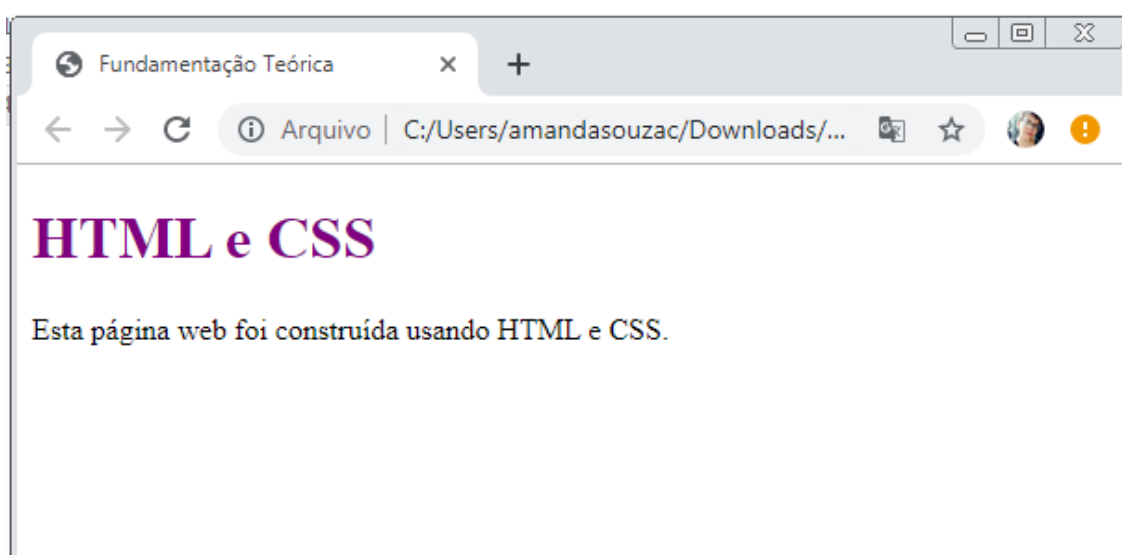


Figura 2.1.2 – Resultado de renderização de código HTML e CSS.

Para efetuar as requisições e descrever a lógica do cliente, são usadas *script languages*, ou linguagens de *script*. São linguagens de alto nível que são interpretadas em tempo de execução. No caso de aplicações Web, essas linguagens, como Javascript, são

responsáveis por efetuar as requisições HTTP e, dessa forma, conduzem a comunicação com o servidor.

Atualmente há vários *frameworks* e tecnologias que facilitam o desenvolvimento e execução de aplicações Web complexas e de grande escala, como, por exemplo, o Angular, um framework *open-source* baseado em Typescript focado na criação de aplicações Web multiplataformas.

2.1.9 Eventos

Neste trabalho, usamos o termo “eventos” para caracterizar eventos sociais, corporativos, religiosos, acadêmicos e quaisquer tipos de eventos que possam ser realizados por uma empresa organizadora de eventos. Vale ressaltar que se optou por modelar um evento em forma de proposta (não necessariamente evento confirmado). Isso se deu devido ao ciclo de vida de um evento, na perspectiva da empresa. Esse ciclo, representado na Figura 2.1.3, usa do nome “proposta” para se referir ao planejamento de um evento.

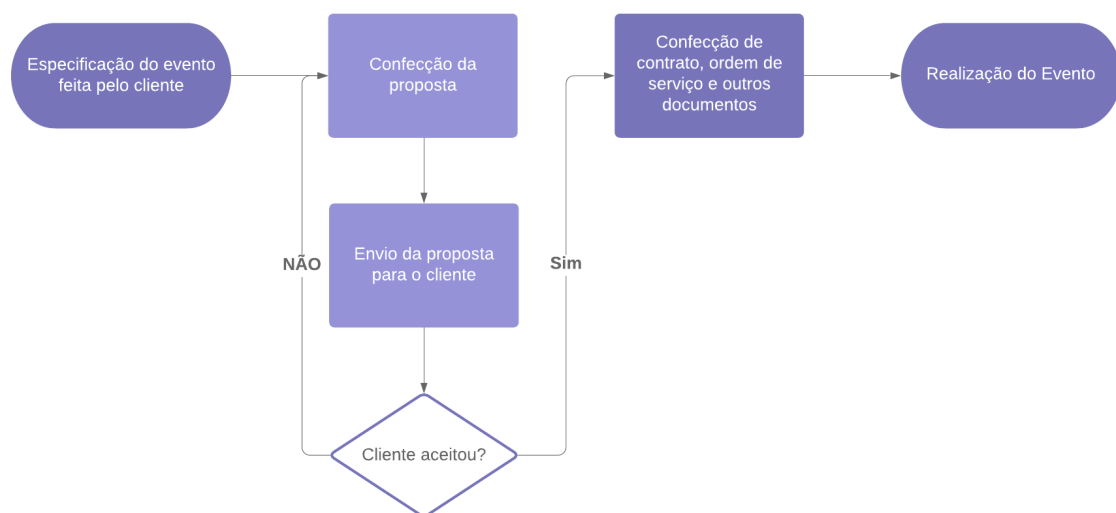


Figura 2.1.3 – Ciclo de vida do evento.

Isto é, quando um cliente entra em contato com a empresa e idealiza o evento que pretende realizar, a empresa trata isso como uma proposta de evento. A empresa, então, analisa os recursos disponíveis e propõe ao cliente um conjunto de alguns desses recursos, planejando como executar o evento. Desse modo, a proposta é como uma modelagem do evento, o que define suas características, registrando dados sobre como, quando e por quem ele será executado.

A proposta permite que o cliente explore as opções possíveis para realizar o seu evento. É por meio da proposta também que o cliente recebe o orçamento do evento, verificando o preço final, as condições de pagamento e condições gerais para a sua realização. O cliente pode aceitar uma proposta ou exigir mudanças no planejamento.

2.2 Trabalhos Relacionados

Atualmente, é possível encontrar alguns *softwares* similares ao SKED no mercado, como o MeEventos¹¹, desenvolvido pela TD Consultoria BI, o Buffetmax¹², desenvolvido pelo Crio Digital e o Buffet Mais¹³. Todos esses *softwares* tem 3 funções similares ao SKED, sendo elas gerenciamento de eventos ou propostas, controle de agenda e confecção de propostas, orçamentos ou contratos.

O MeEventos é uma plataforma Web que atende mais de 200 empresas atualmente, além de ter enviado mais de 200.000 orçamentos pela plataforma. Apesar de ser um sistema completo e similar à aplicação desenvolvida nesse trabalho, o sistema MeEventos foca a funcionalidade de geração de documentos somente nos orçamentos, que são enviados por e-mail, telefone ou como um *link* para o cliente. Informações sobre outros documentos, como contratos, não foram citadas nas informações principais da plataforma.

Já os sistemas Buffetmax e Buffet Mais são mais focados na parte de serviços alimentícios para eventos. Porém, são sistemas com muitas funcionalidades similares ao SKED. Ambas aplicações dispõem de funcionalidades acerca da confecção de contratos, propostas, orçamentos e reservas, algo que se aproxima mais ao sistema proposto. O Buffetmax tem um *layout* bem diferente do sistema proposto,

Além desses *softwares*, Leite Pereira (2012) propõe um sistema muito semelhante em sua monografia, o Sistema Gerenciador de Eventos Corporativo: SIGEC. Apesar de não citar assuntos como usabilidade e prototipação, o que sugere uma metodologia diferente, sua dissertação também é sobre a implementação de um sistema para gerenciamento de eventos e se assemelha a esta pesquisa em vários pontos.

¹¹ <https://www.meeventos.com.br/>

¹² <https://www.buffetmax.com.br/>

¹³ <https://site.buffetmais.com/>

3 Desenvolvimento

Este capítulo apresenta as etapas executadas para o desenvolvimento do projeto, explicando o processo de desenvolvimento do trabalho, envolvendo entendimento do problema, modelagem dos dados, prototipação, testes e análise de usabilidade do sistema, implementação e apresentação da aplicação.

3.1 Levantamento de Requisitos

O levantamento de requisitos foi a primeira etapa do desenvolvimento do projeto. Deu-se por meio de reuniões com profissionais da área administrativa e comercial em empresas locais de eventos, e aconteceram dentro e fora das empresas em questão.

Em um primeiro momento, foi observado como tais profissionais realizavam suas tarefas, focando no fluxo de cadastro de novo evento, geração de documentos e gerenciamento de agendas. Essa etapa foi de suma importância para notar as dificuldades do profissional em realizar tais tarefas, e assim, entender melhor o problema.

Após a observação foram conduzidas conversas com os profissionais, onde focamos em definir o problema. Foi mapeado que, no mínimo 4 atividades eram essenciais para a realização do fluxo completo, sendo elas:

1. Cadastro do evento
2. Inserção dos dados do evento na agenda
3. Visualização da agenda
4. Geração de documentos

Para cada tarefa citada, os dados do evento tinham que ser reinseridos e outros dados tinham que ser consultados em arquivos e aplicações diferentes. Por exemplo, para um dado evento, era necessário consultar tabelas de preço e produtos disponíveis para aluguel em uma Aplicação A, inserir dados do evento em uma Aplicação B responsável pela agenda, e reinserir tais dados na confecção de documentos, feita por uma terceira aplicação C.

A inserção repetitiva de dados e a utilização de múltiplas ferramentas e aplicações gera maior possibilidade de dados divergentes acerca do mesmo evento, sobrecarga de funcionários, e, principalmente, atraso na realização das atividades.

Para solucionar o problema, então, foi decidido que o principal objetivo para o sistema seria unificar geração de documentos, cadastro de eventos e visualização de agenda em uma só aplicação, de forma a diminuir o tempo e quantidade de aplicações necessárias para executar as atividades. Também foi discutido a plataforma em que o sistema funcionaria, considerando uma aplicação *desktop* ou Web.

Após decidir seu principal objetivo, deu-se início a elaboração preliminar dos requisitos. Foram decididas as funcionalidades prioritárias, sendo elas: cadastro de usuário, aprovação ou reprovação do usuário, login, cadastro e edição de proposta de evento, cadastro e edição de modelos de documento, geração de documentos e visualização de agenda. Tais funcionalidades foram descritas da seguinte forma:

R1. Cadastro de Usuário: o usuário deve informar e-mail, nome e senha para se cadastrar no site. No momento de cadastro um acrônimo é atribuído àquele usuário, como forma de identificação. O usuário deverá ser aprovado por um usuário administrador para ter sua conta ativada.

R2. Aprovação ou Reprovação de Usuário: o sistema deverá notificar que houve novo cadastro de usuário e permitir que o administrador aprove ou reprove o mesmo. No caso da reprovação, ele não terá acesso ao sistema, e no caso de aprovação poderá ter acesso quando realizar o login.

R3. Login: O usuário deve informar e-mail e senha para acessar o sistema. É necessário que o usuário esteja cadastrado e aprovado. O sistema deverá retornar mensagens de erro usando linguagem natural, como por exemplo “senha incorreta”.

R4. Cadastro e Edição de Propostas: o sistema contempla uma sequência de formulários para o cadastro de uma proposta de evento. Após cadastrada a proposta terá um código identificador e poderá ser editada somente pelo usuário que a cadastrou.

R5. Cadastro e Edição de Modelos de Documentos: o sistema disponibiliza um editor de texto para que os modelos de documento sejam criados e editados. Também existirá uma lista de possíveis dados que o usuário poderá colocar no modelo.

R6. Geração de Documentos: a partir de uma proposta e um modelo de documento, o sistema gera um documento PDF, preenchendo o modelo com os dados da proposta em questão.

R7. Visualização de Agenda: uma agenda de eventos fará parte do sistema, informando data, horário e usuário responsável pelo evento. Tem um modelo de cores para indicar o status do evento e sua exibição pode ser organizada por dia, semana ou mês, e a disposição dos eventos dá-se em uma matriz que relaciona data com espaços da casa de evento.

Além disso, também foram definidos os *stakeholders*, ou seja, os usuários que utili-

zarão o sistema. Dentro do contexto de uma empresa com diversas áreas, foi definido que 4 funcionários poderiam se beneficiar do sistema, sendo eles das áreas comercial, administrativa, recepção e a equipe responsável por montagem, manutenção e desmontagem do evento. Dessa forma, também foi definido que diferentes perfis de usuários seriam necessários, com diferentes níveis de permissionamento, uma vez que funcionários de diferentes áreas podem interagir de forma diferente com o sistema. Dessa forma, 3 perfis foram definidos, sendo eles:

1. **admin**: perfil com permissão para acessar todas as áreas, assim como autorizar acesso de outros usuários e definir seus perfis;
2. **writer**: perfil com permissão para acessar todas as áreas, com exceção da área de gerenciamento;
3. **reader**: perfil de usuários que apenas visualizam a agenda e dados de propostas já cadastradas.

Sendo assim, somente uma pessoa pode ter perfil de *admin*, sendo um funcionário de alto cargo, como um gerente ou gestor. Já o segundo perfil é destinado aos funcionários da área comercial, que alimentam o sistema cadastrando novos eventos e fazendo a manutenção dos que já existem. E o perfil de *reader* seria para os funcionários de recepção e manutenção dos eventos, que não podem inserir novos dados, mas se beneficiam ao visualizar a agenda e dados de um evento já cadastrado. A Figura 3.1.1 apresenta o Diagrama de Casos de Uso que permite ter uma visão geral das funcionalidades do sistema e os atores envolvidos.

Como descrito nos requisitos R1 e R2 acima, para um usuário de perfil *writer* ou *reader* se cadastrar e ter acesso ao sistema, é necessário que um funcionário com perfil *admin* aprove seu acesso. Tal fluxo está descrito no “Caso de Uso 1: Cadastrar Novo Funcionário”, mostrado na Tabela 1, como forma de descrever a sequência de acontecimentos necessários para o cadastro de um novo usuário *reader* no sistema. Além disso, a Figura 3.1.2 apresenta um diagrama de atividades para melhor ilustrar esse caso de uso, onde é possível perceber os diferentes papéis que os usuários têm, de acordo com seus respectivos perfis e níveis de permissionamento e interação com o sistema.

3.2 Modelagem do Banco de Dados

Depois de definir os requisitos do sistema sucedeu-se o início à modelagem do banco de dados, que mantém as informações do sistema e o alimenta. O banco de dados foi definido e manipulado através do *software* MySQL e da ferramenta MySQL Workbench,

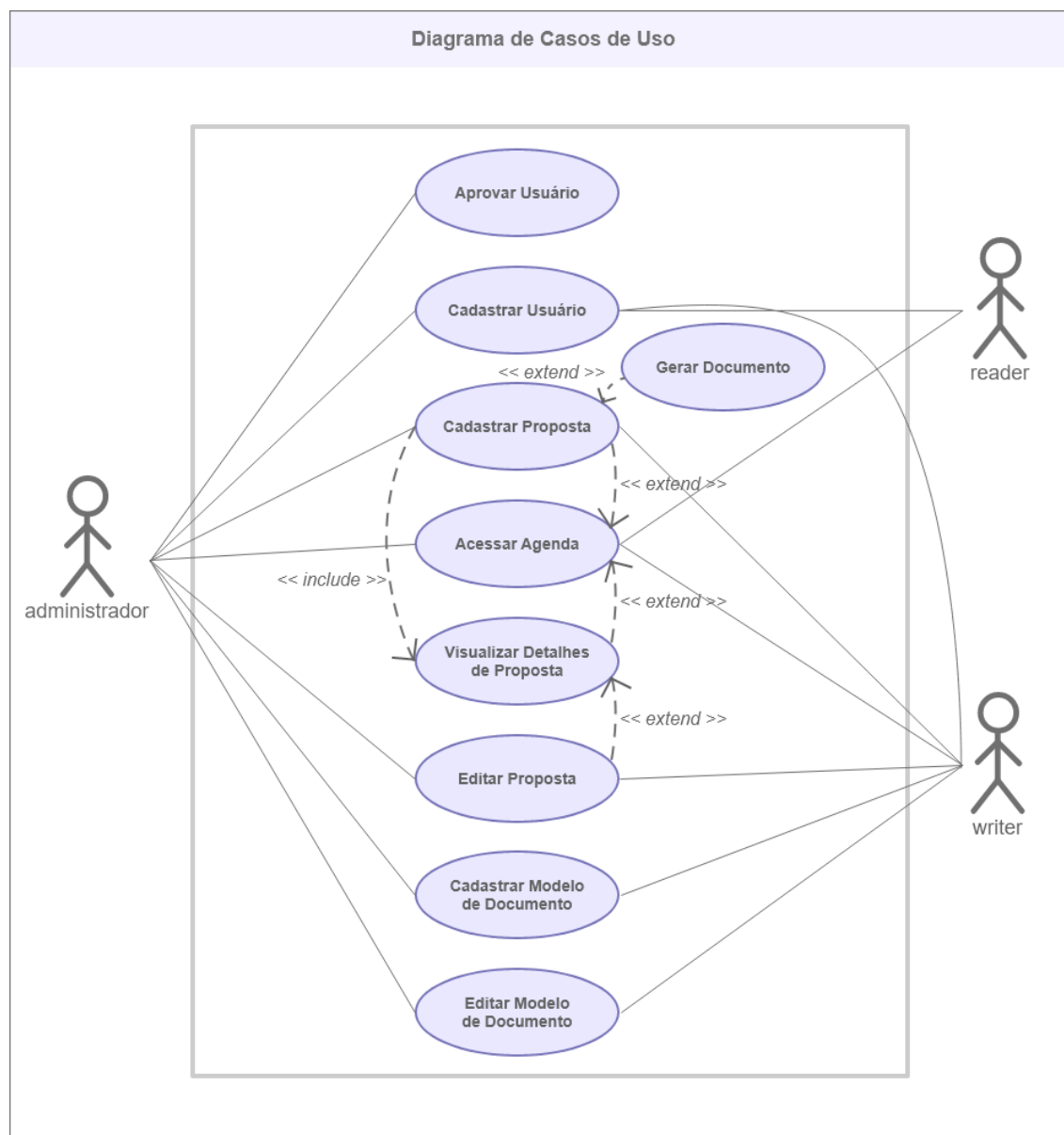


Figura 3.1.1 – Diagrama de Casos de Uso

e tem como principais entidades as tabelas referentes aos dados de cliente, usuário, documento, proposta, espaço e itens. Cada uma dessas tem outras tabelas para armazenar informações secundárias, como tipos e categorias.

Abaixo, na Figura 3.2.1 vemos um fragmento do Diagrama Entidade-Relacionamento (DER). Tal fragmento exhibe a tabela de propostas, sua estrutura e as relações que ela tem com outras tabelas que mantêm informações secundárias sobre as propostas geradas no sistema, sendo essas informações o status da proposta, seu produto e sua categoria. O DER completo pode ser observado no Apêndice G.

Caso de Uso 1: Cadastrar Novo Funcionário	
Escopo	O sistema deverá cadastrar os dados do funcionário que deseja acesso e emitir uma notificação para que o usuário administrador ative a conta recém criada e permita o acesso da mesma ao sistema.
Pré-condição	Existência de um usuário de perfil admin já cadastrado no sistema.
Garantia de Sucesso	Para formulário de cadastro dos dados do novo funcionário tem validação em todos os campos e a conta só é criada se todos os campos forem preenchidos de acordo com as regras do sistema; sempre que uma conta é criada ela tem seu estado como inativo e uma notificação para ativação é enviada para o admin; o usuário recebe notificações quando sua conta é ativada; o login só é possível se a conta estiver ativa;
Cenário de sucesso principal	
Fluxo principal: Cadastro de Dados	1. O usuário clica no link "cadastre-se" e é redirecionado para página de cadastro; 2. O usuário preenche os dados necessários para o cadastro; 3. O usuário recebe confirmação que a conta foi criada e que é necessário aguardar aprovação.
Fluxo secundário: Ativação da conta e login	1. O usuário administrador recebe notificação de nova conta cadastrada no sistema; 2. O usuário administrador acessa lista de usuários pendentes de aprovação e encontra o usuário recém criado; 3. O usuário administrador aprova o usuário, ativando a conta; 4. O usuário administrador atribui um perfil ao usuário recém criado; 5. O sistema notifica o usuário que sua conta foi ativada e o login está autorizado; 6. O usuário acessa o sistema ao efetuar o login preenchendo os dados corretamente.

Tabela 1 – Descrição do Caso de Uso 1: Cadastrar Novo Funcionário.

O modelo de dados da proposta, quando relacionado com outros modelos, armazena os seguintes dados:

- Título do evento;
- Quantidade de pessoas participantes;

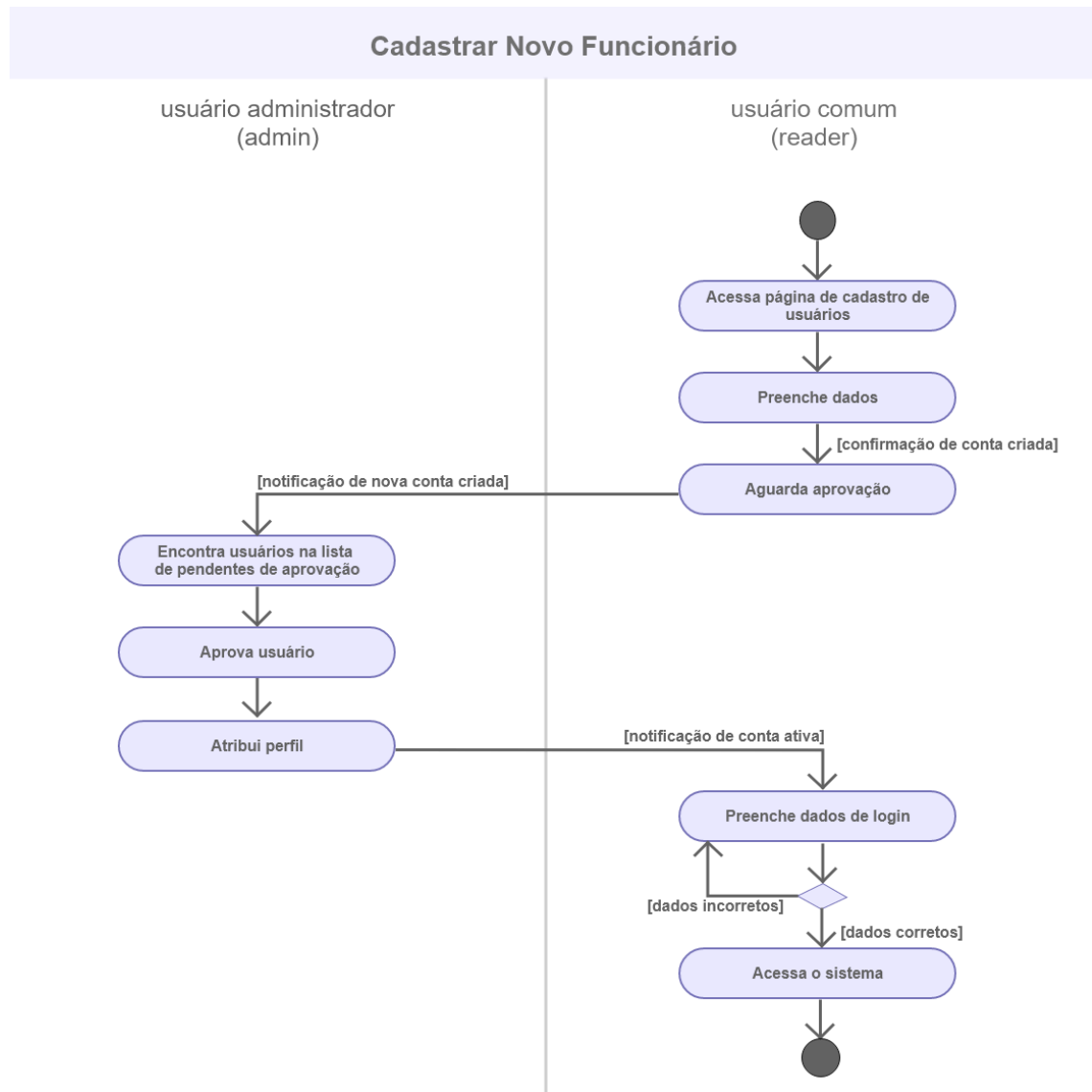


Figura 3.1.2 – Diagrama de Atividades do Caso de Uso 1: Cadastro de Usuários.

- Responsável: um funcionário responsável por aquela proposta;
- Dados do cliente;
- Um produto e sua categoria: definem o tipo do evento, podendo ser:
 - Produto Corporativo
 - * Categoria Simples
 - * Categoria Completa
 - * Categoria "Eventos Próprios"
 - Produto Social
 - * Categoria "Eventos Próprios"
 - * Categoria "Eventos Clientes"

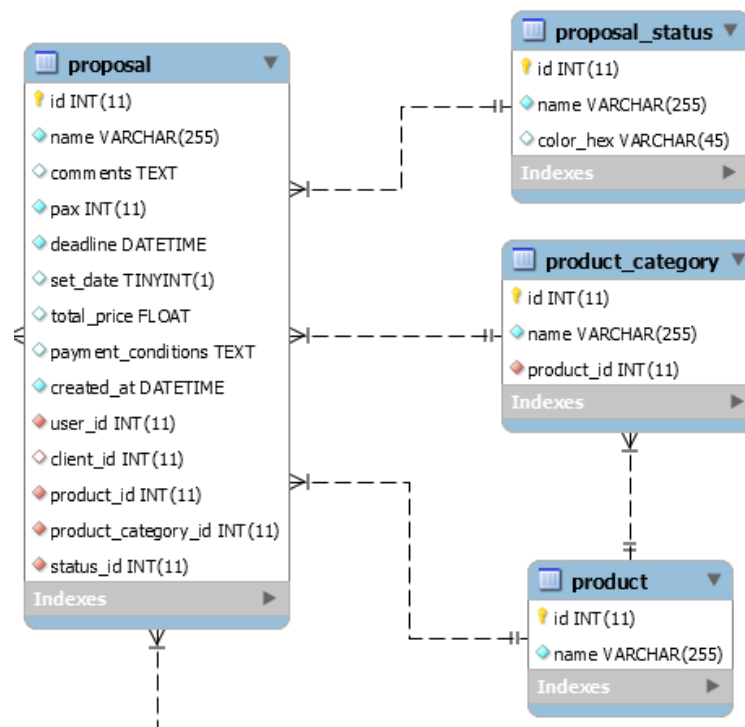


Figura 3.2.1 – Fragmento do DER do banco de dados.

- Status: caracteriza o status da proposta, podendo ser:
 - A Confirmar: esperando por confirmação do cliente.
 - Confirmado: confirmado e aguardando datas de realização.
 - *Wait List*: aguardando confirmação ou desistência de alguma data já reservada à outra proposta.
 - Realizado
- Data limite para definição da proposta: intitulada *deadline*, essa é uma data limite para que o cliente analise a proposta, critique-a ou a aceite;
- Datas de realização do evento: são as datas que serão reservadas àquele evento, tanto para montagem, desmontagem ou realização.
- Data de criação da proposta;
- Espaços: salas, combinações de salas e espaços externos que são alugados para a realização do evento.
- Serviços prestados: são os serviços que a empresa disponibiliza para a realização do evento, podendo ser das seguintes categorias:
 - Alimentos e Bebidas
 - Projeção, luz e som

- Decoração
 - Música
 - *Staff*
 - Estacionamento
 - Diversos
- Preço total: soma dos preços dos serviços prestados e salas alugadas;
 - Condições de pagamento;
 - Comentários adicionais.

3.3 Protótipo

Para implementar as funcionalidades propostas e disponibilizá-las de forma a diminuir as possíveis dificuldades que o usuário teria ao usar o sistema, efetuou-se um estudo de usabilidade com o objetivo de testar e melhorar a sua interface. No estudo previamente citado foi usado um protótipo de interface de alta fidelidade que ilustra parte do sistema proposto.

No desenvolvimento do protótipo foram seguidas boas práticas de interação entre humanos e computadores, particularmente os fundamentos adotados por [Nielsen \(1993\)](#), apresentados como 10 heurísticas de usabilidade. No processo de criação da interface foram usadas algumas dessas heurísticas como guias de *design* e desenvolvimento do protótipo, sendo elas parte do produto final e podendo ser destacadas e analisadas. Cinco das heurísticas previamente citadas foram amplamente usadas na interface do sistema, sendo elas:

- Visibilidade do status do sistema;
- Compatibilidade do sistema com o mundo real;
- Controle e liberdade para o usuário;
- Consistência e padronização;
- Estética e design minimalista;

O protótipo trata do fluxo da funcionalidade “Cadastrar nova proposta”, iniciado na sessão principal de propostas, e composto por diferentes formulários. Cada formulário é apresentado em uma nova página, e ao final há uma página de confirmação dos dados. Também há a tela de agenda dos eventos.

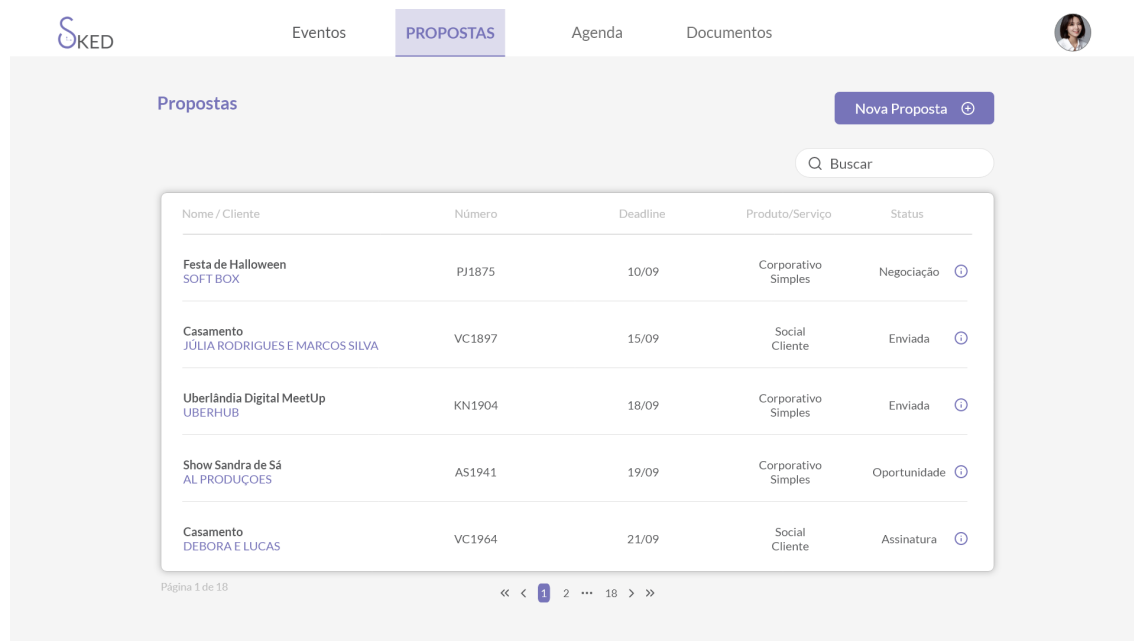


Figura 3.3.1 – Página de listagem de propostas.

A Figura 3.3.1 exemplifica a sessão de propostas do sistema, iniciando o fluxo referido, onde há a opção de criar nova proposta ou buscar por uma existente, além da visualização, em forma de tabela, das propostas já cadastradas, ordenadas de modo decrescente pelas suas respectivas *deadlines*. A heurística “visibilidade do *status* do sistema”, que preza por indicar ao usuário o *status* do sistema onde ele se encontra, pode ser notada na paginação e legenda da tabela, que destacam qual página da tabela está sendo exibida. Tal heurística também é percebida no menu superior do sistema, comum em todas as telas, que informa qual seção do sistema o usuário está acessando.

Uma vez que o usuário escolhe criar uma nova proposta, ele é redirecionado para uma sequência de formulários referentes aos dados necessários para a proposta em questão. Tais formulários são separados por: dados do cliente, informações do serviço, locação de espaço e serviços e produtos. Os dois primeiros formulários, referente aos dados do cliente e informações do serviço, estão representados pelas Figuras 3.3.2 e 3.3.3, respectivamente.

Em todos os formulários há a opção de preencher os dados exigidos, cancelar a operação, seguir para a próxima etapa ou voltar à etapa anterior. É notável que em todos os formulários tais opções são dispostas nos mesmos lugares e seguindo os mesmos padrões. Os dados são exigidos em *inputs* dispostos em duas colunas, os botões de cancelar ou prosseguir estão na parte inferior e a barra de progresso que permite a navegação pelos formulários é exibida na parte superior. Essas características que fazem parte de todas as telas de formulário são evidências da heurística “consistência e padronização”, que sugere manter consistência entre as telas para que o usuário possa reaproveitar ao máximo o entendimento que ele já adquiriu do sistema.

The screenshot shows the 'Nova Proposta' (New Proposal) form in the SKED system. The 'PROPOSTAS' tab is selected in the top navigation bar. The form is titled 'Nova Proposta' and has a progress bar with four steps: 'Dados do Cliente', 'Informações do Serviço', 'Locação de Espaço', and 'Serviços e Produtos'. The 'Dados do Cliente' tab is active, showing a form with the following fields:

Dados do Cliente	
Tipo	PESSOA JURÍDICA
CEP	69049-450
CHPJ	18.101.544/0001-05
Inscrição Estadual	388.108.598.269
Rua	RUA BISPO DE HEBRON
Nome	PRODUTOS ALIMENTÍCIOS LTDA
Número	115
Bairro	REDENÇÃO
Email	PRODUTOS@ALIMENTICIOS.COM
Cidade	UBERLÂNDIA
Celular	(34) 99874-9874
Telefone	(34) 3225-8696
Estado	MINAS GERAIS

At the bottom of the form, there are two buttons: 'Cancelar' (Cancel) and 'Próximo >' (Next).

Figura 3.3.2 – Formulário de dados do cliente da nova proposta.

The screenshot shows the 'Nova Proposta' (New Proposal) form in the SKED system, with the 'Informações do Serviço' (Service Information) tab selected. The form is titled 'Nova Proposta' and has a progress bar with four steps: 'Dados do Cliente', 'Informações do Serviço', 'Locação de Espaço', and 'Serviços e Produtos'. The 'Informações do Serviço' tab is active, showing a form with the following fields:

Informações do Serviço	
Produto	CORPORATIVO
Nome	FESTA DE FIM DE ANO
Categoria do Produto	SIMPLES
Tipo	CONFRATERNIZAÇÃO
Quantidade de Convidados	300
Deadline	22/10/2019
DEFINIR DATAS?	☐

At the bottom of the form, there are two buttons: 'Cancelar' (Cancel) and 'Próximo >' (Next).

Figura 3.3.3 – Formulário de informações do serviço da nova proposta.

Nova Proposta
Confirmação de Dados

Dados do Cliente	Informações do Serviço
Nome: Produtos Alimentícios LTDA.	Nome do Serviço: Festa de Fim de Ano
Tipo: Pessoa Jurídica	Produto/Serviço: Corporativo Simples
CNPJ: 18.101.544/0001-05	Convidados: 300
Insc. Estadual: 388.108.598.269	Tipo: Confraternização
Email: produtos@alimenticios.com	Cronograma:
Telefone: (34) 99874-9874	22/12: 15:00 Montagem - 19:00 Início - 23:59
CEP: 69049-450	23/12: 00:00 - 07:00 Fim - 13:00 Desmontagem
Endereço: Rua Bispo de Hebron, 115	
Bairro: Redenção	
Cidade/UF: Uberlândia/MG	

Locação de Espaço	Serviços e Produtos
Espaços: Sala Évora 1 + Sala Évora 2 + Varanda Douro	A&B: ----- R\$ 11.700,00
Évora 1: ----- R\$ 3.000,00	02 Mesa de Frios: ----- R\$ 5.100,00
Évora 2: ----- R\$ 3.000,00	01 Buffet de Jantar - Opção 01 ----- R\$ 6.600,00
Varanda Douro: ----- R\$ 3.000,00	Estacionamento: ----- R\$ 0,00
Valor Total: ----- R\$ 9.000,00	80 Vagas ----- R\$ 0,00
	Valor Total: ----- R\$ 11.700,00

Valor Total da Proposta: R\$ 20.700,00

Valor por Pessoa: R\$ 69,00

Figura 3.3.4 – Confirmação de dados preenchidos no formulário.

Após o preenchimento dos dados, o usuário pode visualizá-los em uma página de confirmação, representada na Figura 3.3.4. Dessa forma, é possível analisar todos os dados e, se necessário, editar os mesmos. Também há as opções de excluir e salvar a proposta. No caso da exclusão, é exibido um modal de confirmação, como consta na Figura 3.3.5. Tal modal dá ao usuário a opção de confirmar a exclusão dos dados ou manter os mesmos, dessa forma diminuindo o risco da exclusão ser feita por engano. Tais princípios são referentes a heurística “Controle e liberdade para o usuário”.

Uma vez que a proposta foi salva, uma tela de finalização é mostrada, como na Figura 3.3.6. Nessa tela há a confirmação de que a proposta foi salva com sucesso e também há as opções de voltar à tela de propostas ou gerar um documento de texto para a proposta recém cadastrada. Nessa página é notável duas das heurísticas previamente citadas, sendo elas “Compatibilidade do sistema com o mundo real” e “Estética e design minimalista”. A primeira delas faz-se presente na escolha de adicionar aos botões ícones que remetem ao mundo real, facilitando que o usuário reconheça mais facilmente qual ação tal botão pode executar. Já a segunda heurística consiste em manter somente informações

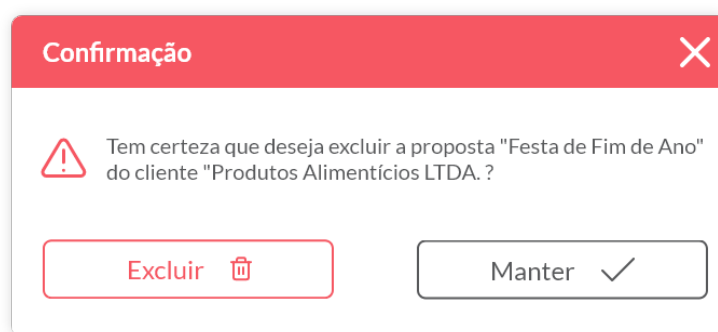


Figura 3.3.5 – Modal de confirmação de exclusão de dados.

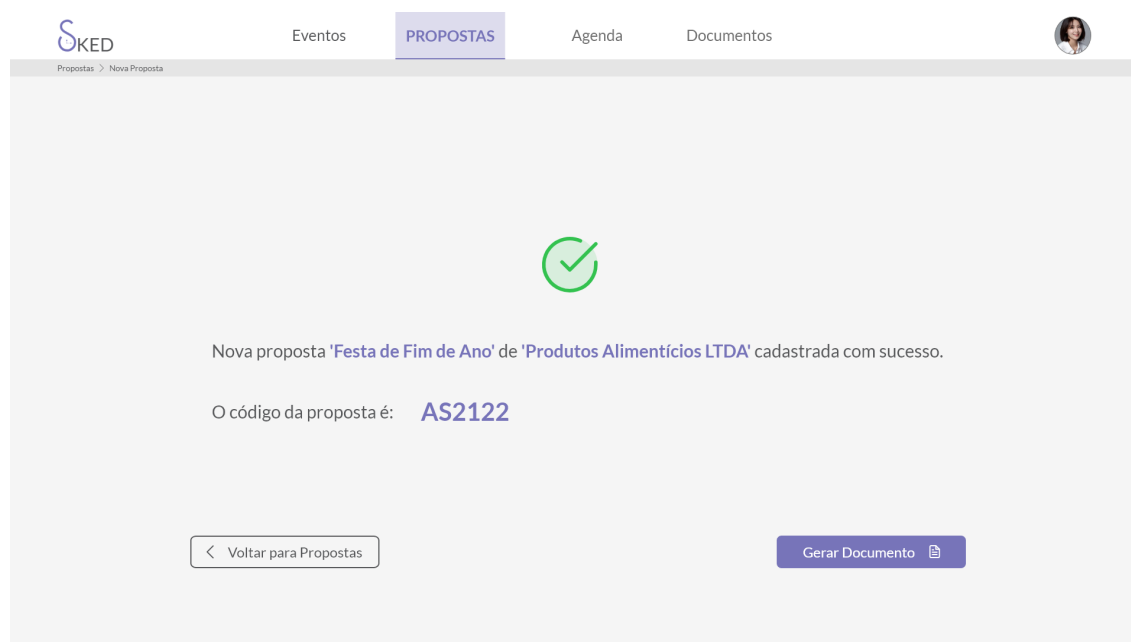


Figura 3.3.6 – Tela de finalização do fluxo de cadastrar nova proposta.

relevantes na tela, como o nome do cliente e o serviço proposto, de forma que o usuário pode facilmente encontrar o que deseja.

3.4 Testes e Análises de Usabilidade

Com o objetivo de testar a facilidade de uso da interface proposta, foram aplicados testes de usabilidade com usuários voluntários, utilizando o protótipo de alta fidelidade citado anteriormente. O principal objetivo de tais testes era identificar e analisar possíveis dificuldades que o usuário pudesse encontrar ao utilizar o sistema, considerando seu *design*.

O teste consistiu em um usuário usar o protótipo para efetuar tarefas pré-estabelecidas (ver Apêndice C), com o objetivo de cadastrar uma nova proposta. Além disso, também faziam parte do teste a assinatura de um Termo de Consentimento Livre e Estabelecido, Instruções e Informações Importantes, Questionário Demográfico, Questionário de Análise

de Usabilidade e uma ficha para o instrutor fazer anotações e perguntas (ver apêndices A, B, D, E e F).

A execução do teste completo continha uma ordem específica, representada pela Figura 3.4.1, e aconteceu em um ambiente controlado, com a participação de dois instrutores e um usuário por vez, conforme *layout* mostrado na Figura 3.4.2.

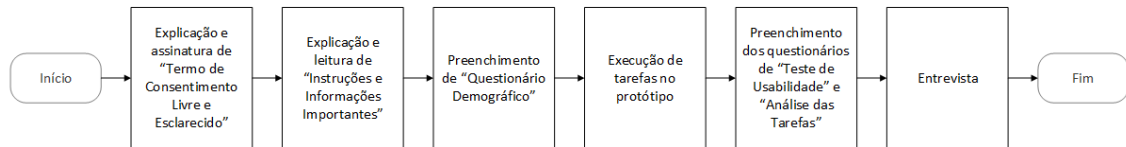


Figura 3.4.1 – Fluxograma de etapas para execução do teste de usabilidade.

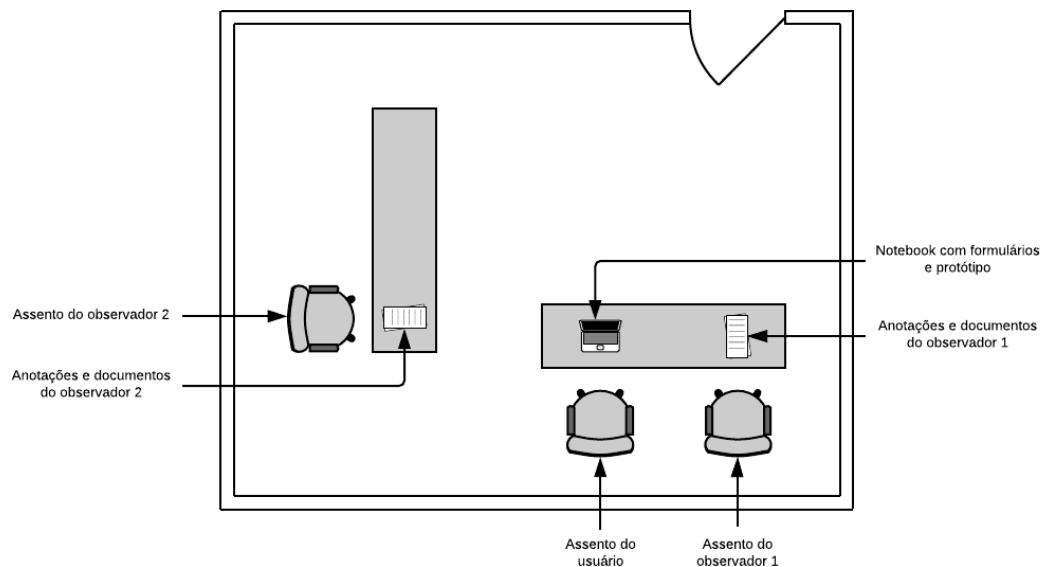


Figura 3.4.2 – Planta da sala de aplicação de testes.

No total, cinco usuários participaram do teste, sendo três profissionais da área de gestão de eventos e dois outros usuários (Figura 3.4.3). Seus objetivos eram executar tarefas para visualizar a agenda e cadastrar uma nova proposta no sistema. Para isso, eles deviam realizar as tarefas em ordem decrescente, visto que há uma hierarquia entre elas, como mostrado na Figura 3.4.4, representando o HTA (*Hierarchical Task Analysis*) do teste.

Durante o teste foram utilizados métodos observacionais e de perguntas aos usuários para a extração de dados (DIX et al., 2003). O método observacional que mais se assemelha à estratégia abordada foi o *think aloud*, de forma que o instrutor, por meio de anotações, capturava dados acerca de erros na execução das tarefas, indagações do usuário e suas reações afetivas. Após a execução das tarefas, o usuário respondia um questionário



Figura 3.4.3 – Representação gráfica sobre atuação profissional dos usuários que realizaram o teste.

misto, com escalas e perguntas abertas ou fechadas. E para finalizar a coleta de dados, foi feita uma entrevista não-estruturada, ou seja, o instrutor fazia perguntas abertas, onde o usuário decidia o nível de detalhamento na resposta. Os tópicos da entrevista eram relacionados com pontos importantes que o instrutor observou enquanto o usuário realizava as tarefas.

Os questionários geraram dados importantes para a análise da usabilidade do sistema e a determinação do que precisava ser alterado ou mantido. O questionário feito após o teste foi dividido em duas sessões: a primeira sobre a experiência geral do usuário com o sistema e a segunda era focada nas tarefas realizadas.

Essa primeira parte trouxe dados como o quanto o usuário achou o sistema simples, foco principal do teste. Feita usando escalas, o usuário tinha que analisar uma afirmação e concordar ou discordar com a mesma. Na Figura 3.4.5 nota-se que 4 dos participantes (80%) concordam que houve certa facilidade ao usar o sistema, e um deles concorda totalmente com essa afirmação.

Além disso, resultados similares acontecem em outras perguntas, onde a maioria dos usuários concordam que conseguiriam usar o sistema sem treinamento prévio (Figura 3.4.6) e que seria confortável usar o sistema frequentemente (Figura 3.4.7).

Tais dados mostram que o sistema cumpriu o propósito de ser simples e de fácil utilização, e assim não seria necessário efetuar grandes alterações. Apesar disso, na segunda parte do questionário temos maiores informações sobre as tarefas individualmente, sendo possível explorar melhor os pontos fracos e fortes do sistema.

Primeiramente os usuários classificaram o nível de dificuldade para efetuar cada tarefa, como “fácil”, “médio” e “difícil”. A Figura 3.4.8 mostra que as tarefas consideradas menos fáceis foram as duas últimas no teste, tarefas 6 e 7. A tarefa 6 simulava a exclusão e o ato de salvar a proposta sendo cadastrada, enquanto a tarefa 7 consistia em verificar o

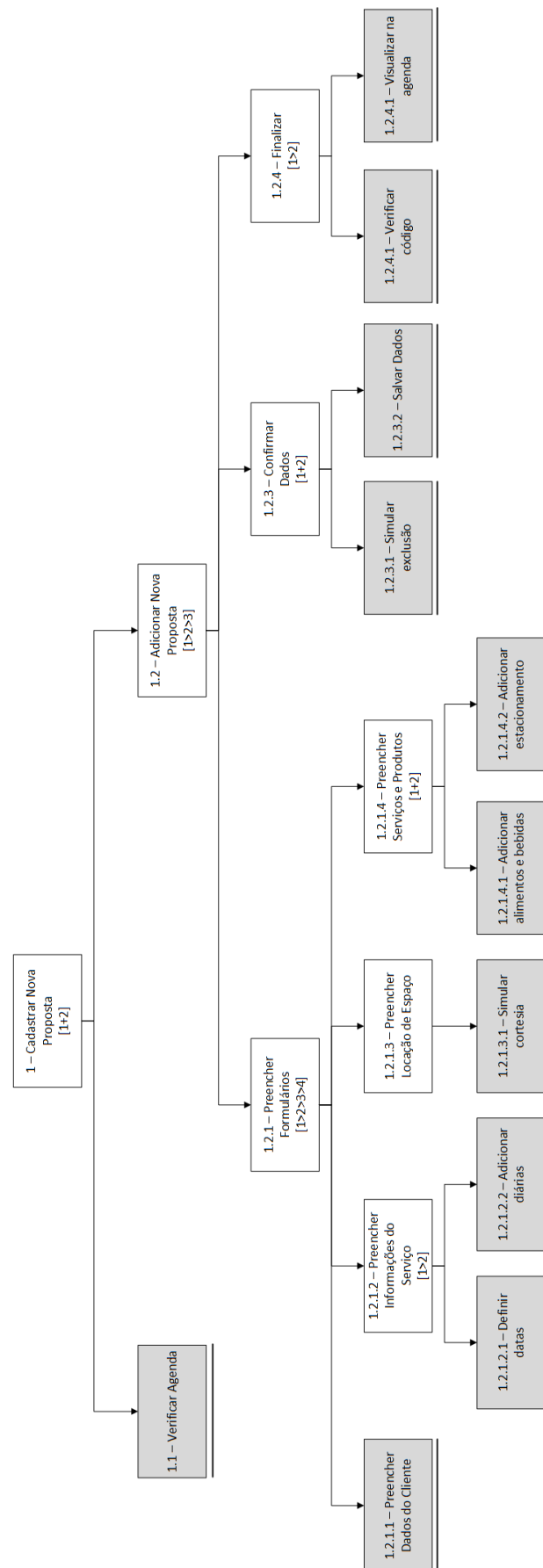


Figura 3.4.4 – HTA das tarefas para teste do protótipo.

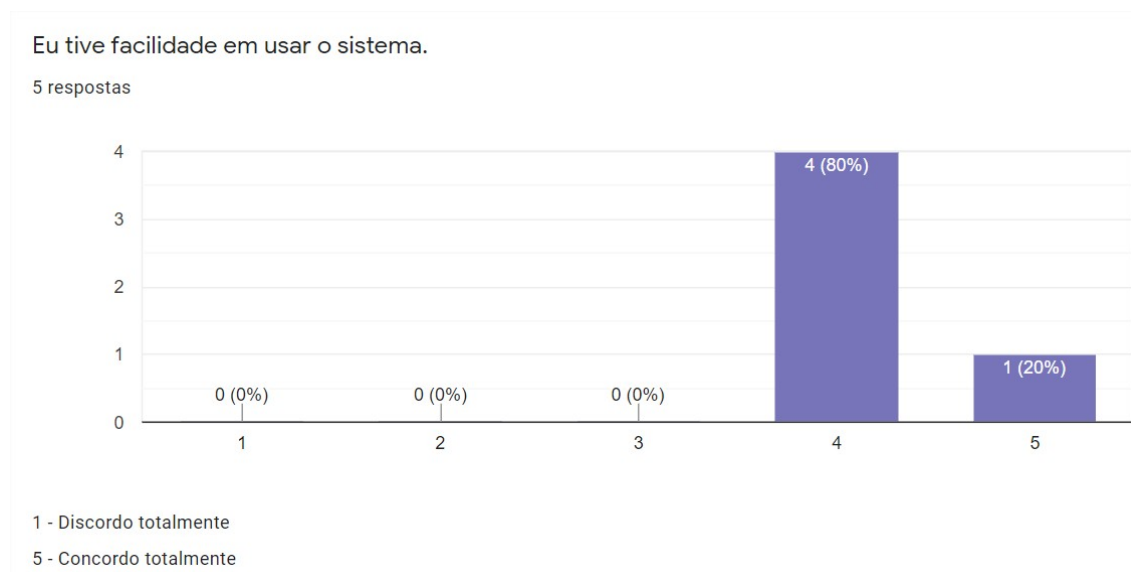


Figura 3.4.5 – Distribuição de nível de dificuldade que os usuários tiveram ao usar o sistema.

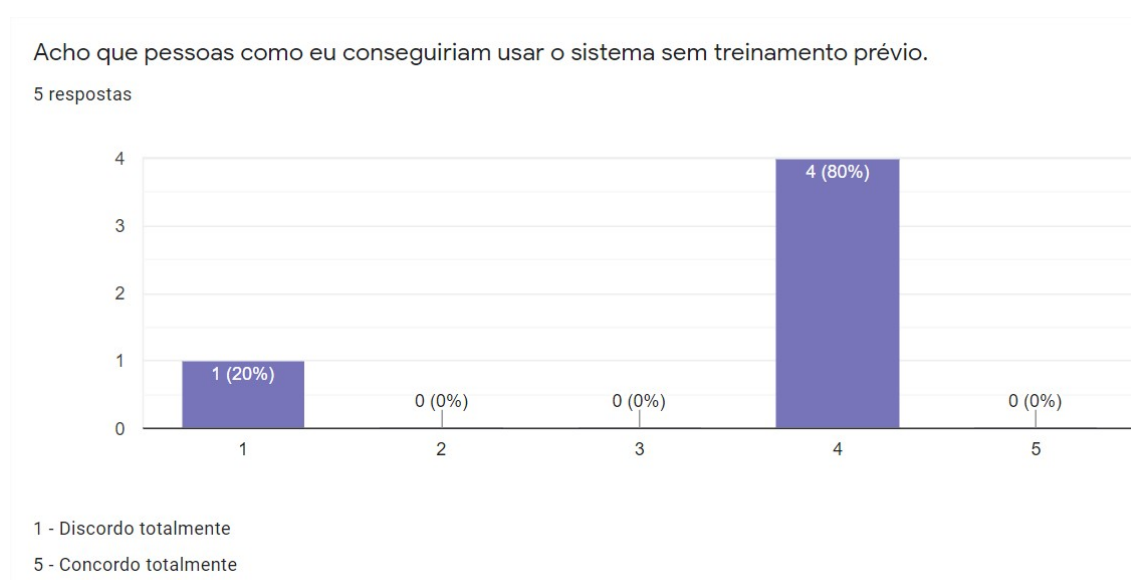


Figura 3.4.6 – Distribuição de opiniões dos usuários sobre ser possível usar o sistema sem treinamento prévio.

código da proposta recém criada e usar tal código para encontrar a proposta em questão na agenda.

Os dados extraídos da tarefa 6 (Tabela 2) mostram que 4 dos 5 usuários apresentaram ações fora do esperado, caracterizando erro. Além disso, tais participantes tiveram sucesso parcial na execução, e apenas um teve sucesso total. Tais resultados interferiram na tarefa posterior diretamente, uma vez que para realizar a tarefa 7 era necessário sucesso na execução da tarefa 6. Tal fato pode ser notado nas observações feitas pelos instrutores, como por exemplo, a observação a seguir, feita durante um dos testes:

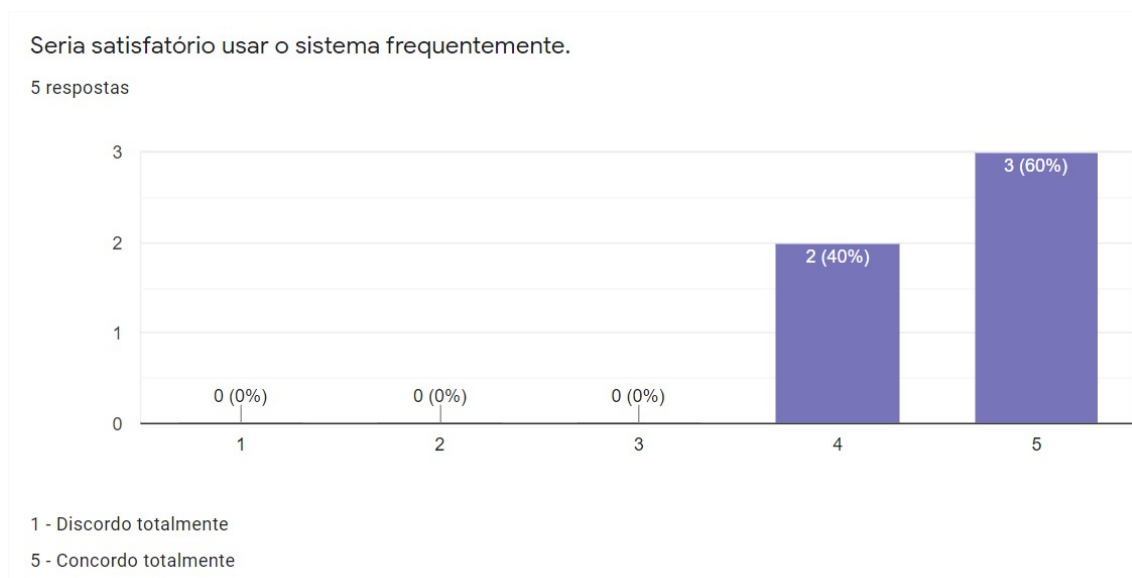


Figura 3.4.7 – Distribuição de opiniões dos usuários sobre se seria satisfatório usar o sistema frequentemente.

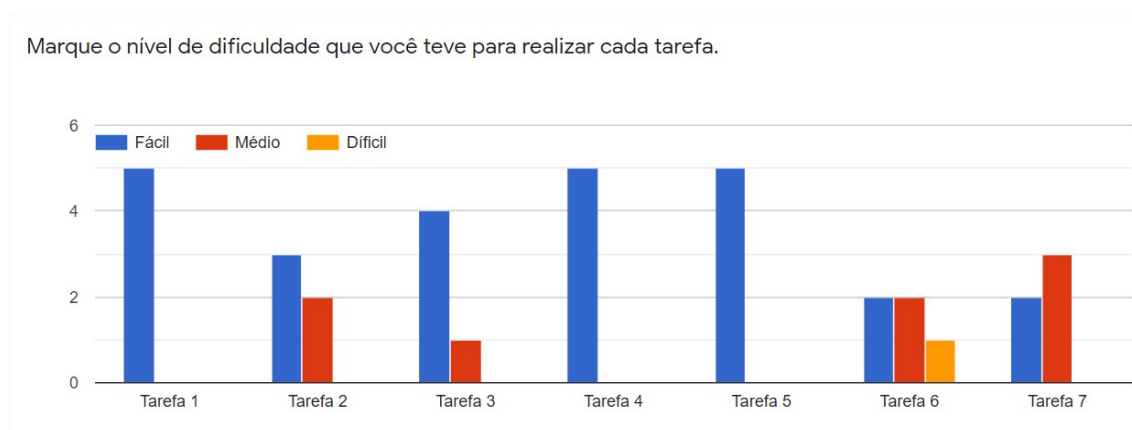


Figura 3.4.8 – Distribuição de classificação de dificuldade em executar tarefa.

“Como o usuário não terminou a tarefa 6, ele não atingiu a tela de confirmação e não conseguiu achar o código da proposta. Então, tentou editar os dados, cancelou o formulário e voltou para a tela inicial de Propostas. Partindo desta tela, ele verificou a agenda, que não continha a proposta recentemente adicionada.”

Todos esses dados mostravam fortes indícios de que as duas últimas tarefas precisavam ser alteradas na implementação, e além disso, os usuários também sugeriram claramente que as funcionalidades acerca dessas tarefas precisavam ser melhoradas, como mostrado na imagem [3.4.9](#).

Dessa forma, duas telas foram alteradas, sendo elas a de confirmação de dados da proposta e a de finalização, que mostrava o código da proposta salva. Além disso também foi alterado o modal de confirmação para a exclusão da proposta.

	#1	#2	#3	#4	#5
Tempo (s)	30	82	25	53	206
Número de Erros	0	1	1	1	2
Ajuda	0	0	0	0	0
Sucesso	Sim	Não	Não	Não	Não
Sucesso Parcial	Não	Sim	Sim	Sim	Sim
Fracasso	Não	Não	Não	Não	Não
Confusão/Frustração	Não	Não	Não	Não	Sim
Desistência	Não	Não	Não	Não	Não

Tabela 2 – Dados coletados sobre a execução da tarefa 6.

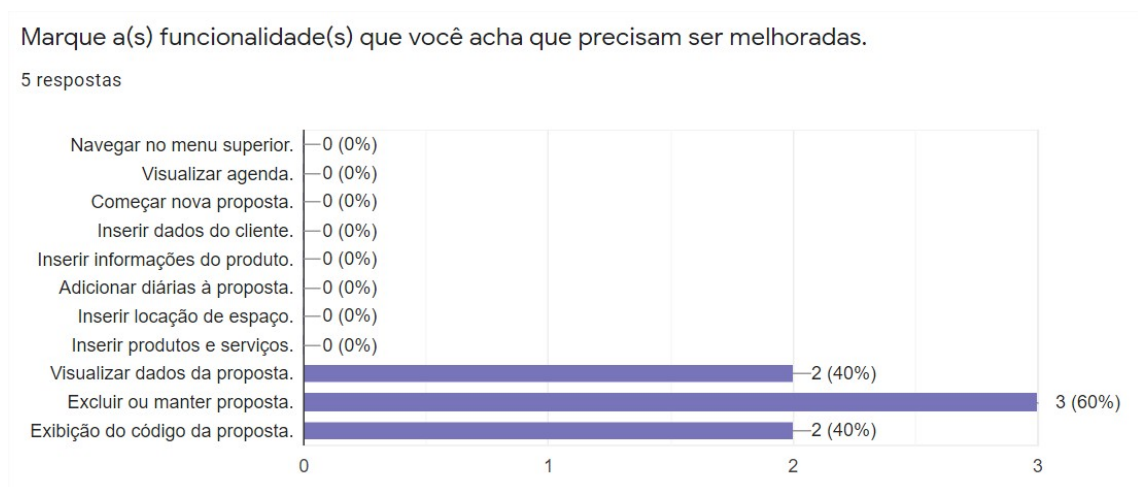


Figura 3.4.9 – Sugestões dadas pelos usuários sobre quais funcionalidades precisam ser melhoradas.

Para a tela de confirmação manteve-se as opções de deletar e salvar apresentadas no protótipo (Figura 3.3.4). Porém, para manter um *layout* mais simples e minimalista, somente as informações do serviço são mostradas nessa tela (Figura 3.4.10). Para visualizar os outros dados da proposta, o usuário deve clicar no link "mais detalhes", que abrirá em nova aba a página de detalhes da proposta, já apresentada na figura 3.5.6.

A alteração do modal de confirmação de exclusão foi guiada pela experiência de um dos usuários do teste. No protótipo, o modal de confirmação tinha duas opções, representadas por dois botões, sendo elas: “excluir”, para deletar a proposta, e “manter” para não deletar (Figura 3.3.5). Logo, na execução de um dos testes, o instrutor pode observar que o usuário não completou a tarefa 6, pois deduziu que ao apertar no botão “manter” a proposta seria automaticamente salva. Dessa forma, optou-se por um novo *layout* (Figura 3.4.11), que usou o termo “cancelar” para a opção referente ao não cancelamento da proposta, deixando mais claro que tal ação não era relacionada com salvar a proposta.

Já a finalização da proposta, que no protótipo era uma tela com nome da proposta e do cliente, código da proposta e opção de gerar documento, foi alterada para um modal (Figura 3.3.6). Em um dos testes, um dos usuários verbalizou que não tinha certeza se

Informações do Serviço	
Nome	Apresentação De Tcc
Funcionário Responsável	Usuário Admin
Status	● Confirmado
Produto/Categoria	Social - Eventos Clientes
PAX	150
Deadline	20/12/19
Cronograma	Diária → 29/11/19 - 15:00 - 16:30
Preço	R\$ 1,920.00
Data de Entrada	17/10/19

[Ver proposta completa](#)

[Deletar](#) [Salvar](#)

Figura 3.4.10 – Novo *layout* de tela de confirmação.

Confirmação

⚠ Tem certeza que deseja excluir proposta "Apresentação de TCC" do cliente "Amanda de Souza Caetano" ?

[Excluir Proposta](#) [Cancelar](#)

Figura 3.4.11 – Novo *layout* de modal para confirmar exclusão de proposta.

aquela era realmente a tela de finalização e se o processo tinha de fato sido finalizado. Além disso, somente um dos usuários conseguiu acessar a agenda diretamente da tela de finalização e, assim, completar a tarefa totalmente.

Dessa forma, um novo modal foi feito (3.4.12), mantendo as informações da proposta e adicionando um título para indicar mais claramente que o fluxo de criação da proposta tinha sido concluído. Também foi adicionado um botão para acesso direto à

agenda, facilitando a funcionalidade de ir até a agenda e verificar a nova proposta.



Figura 3.4.12 – Novo *layout* de modal para informar a finalização do processo de criação da proposta.

Além disso, algumas alterações também foram feitas acerca da tarefa 3, que tinha como objetivo adicionar datas à proposta. Apesar de não citada pelos usuários nas sugestões de melhorias ou na pergunta sobre funcionalidade a serem melhoradas, foi a tarefa que teve maior média de tempo para ser executada (Figura 3.4.13).

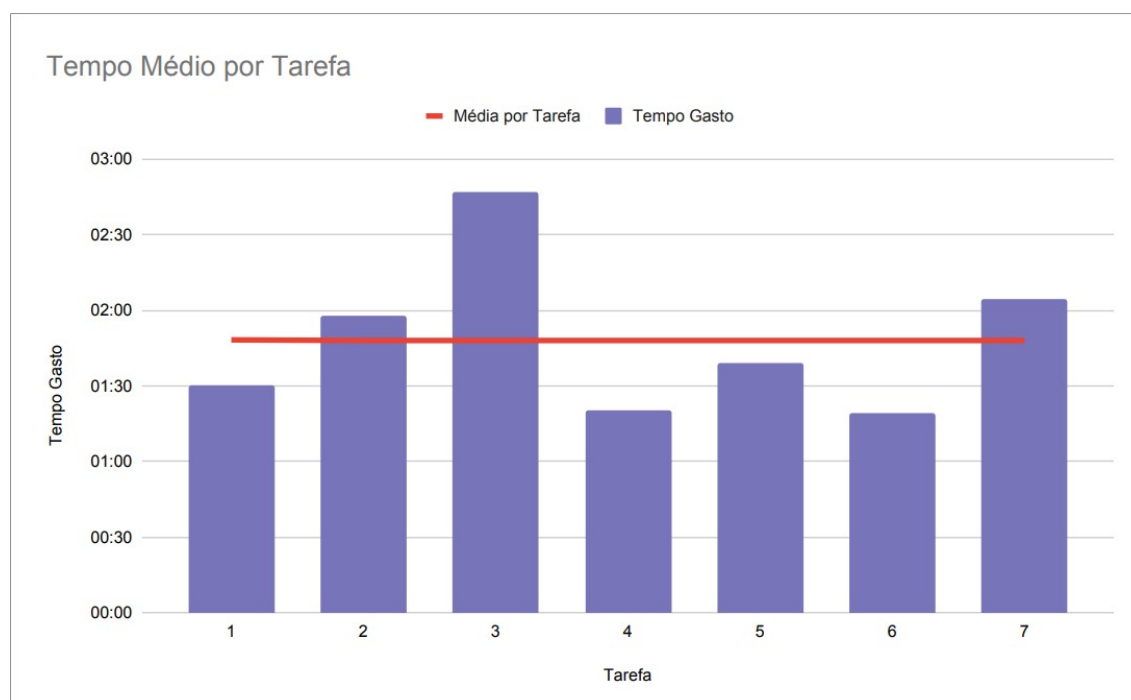


Figura 3.4.13 – Média de tempo para executar cada tarefa.

Também foi observado pelos instrutores, que o campo *deadline*, responsável pela data de expiração da proposta, causava confusão com as datas de acontecimento do evento,

Figura 3.4.14 – Novo formulário para gerenciar datas da proposta.

e 2 dos 5 usuários tentaram editar a *deadline*, ação que não fazia parte da tarefa. Um trecho de entrevista com um usuário não profissional da área concretiza tal fato, a saber:

Pergunta: Vi que você insistiu em alterar a deadline da proposta, como se fizesse parte da tarefa 3. Por que?

“Não vi que se tratava da deadline. Como o nome estava muito pequeno, achei que pudesse ser algo relacionado com as datas do evento.”

Dessa forma, o fluxo de formulários para cadastrar uma proposta foi acrescentado de um novo formulário, exclusivo para o cadastro das datas de acontecimento do evento (I.0.3). Além disso, aumentou-se o tamanho da fonte para o *label* dos campos do formulário. Uma das principais diferenças entre o formulário implementado e o formulário apresentado no protótipo é que, na implementação, as datas são divididas em abas, de acordo com seu tipo (montagem, diária e desmontagem).

Quanto ao tempo que os usuários levaram para completar o teste, a média foi de 12 minutos e 37 segundos (Figura 3.4.15). Uma das queixas que funcionários relatavam na etapa de entendimento do problema, era exatamente a do consumo excessivo de tempo para registrar os dados da proposta, visto que algumas informações, como datas disponíveis e dados acerca de serviços, produtos e espaços, encontravam-se em outras aplicações e arquivos. Além disso, era necessário que os dados da proposta fossem inseridos em múltiplas aplicações, prolongando ainda mais o tempo da tarefa.

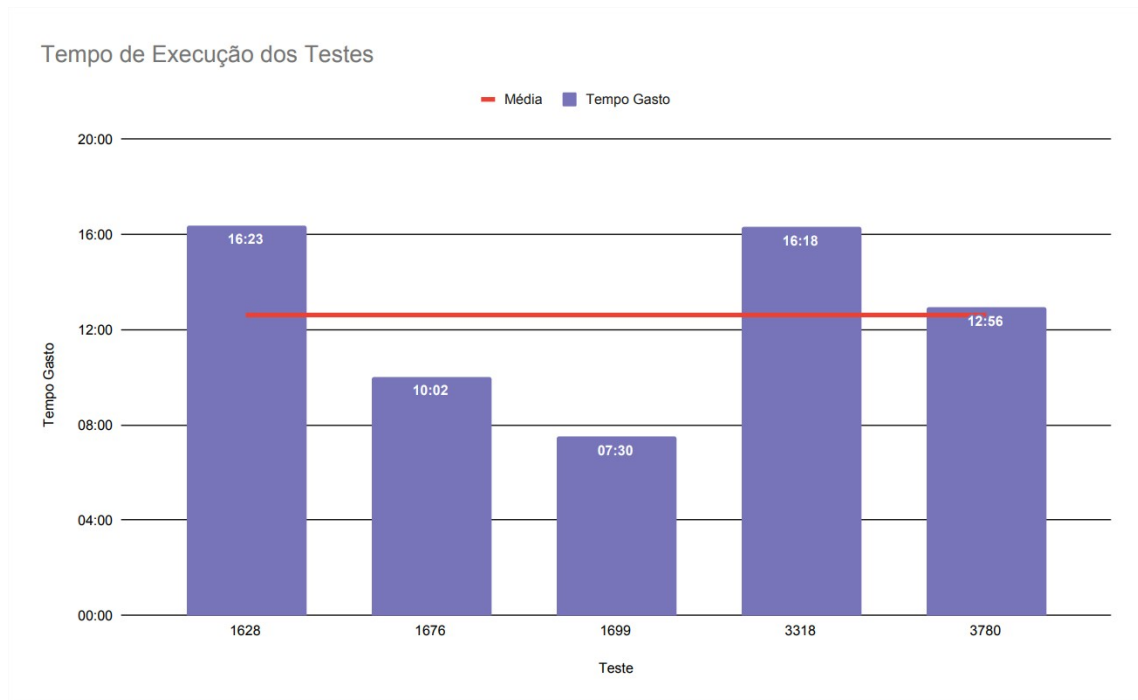


Figura 3.4.15 – Tempo de duração para execução dos testes individualmente e média.

Essa média mostra, então, o potencial da aplicação de otimizar o uso do tempo de tais funcionários. Tal fato pôde ser notado em uma das entrevistas, feita com um profissional da área de eventos, exposta no trecho a seguir:

Pergunta: O que achou de ter tudo(agenda, documentos e proposta) em um sistema só?

“Mais simples, mais interessante, pois todas as informações estão em um lugar só. Nosso sistema não aparece a agenda junto, é tudo separado: agenda, documentos, propostas, contratos.”

Pergunta: Como você acha que essa unificação impactaria no seu trabalho?

Resposta: “Com tudo num sistema só é mais ágil e sobra tempo para outras atividades, como vender mais.”

Os testes foram capazes de contribuir, através de entrevistas, observações e questionários, com a qualidade do sistema, otimizando sua facilidade e provendo de um *feedback* essencial para que, na fase do desenvolvimento, melhores escolhas fossem feitas.

3.5 Implementação

Após os testes e a análise de seus resultados, começou-se a etapa de implementação do sistema, que consiste em três módulos: a aplicação Web, a API e o banco de dados.

A aplicação Web é a parte de *front-end* do sistema, responsável pela interface visual e da interação do usuário. É através dela que o usuário faz requisições à API, parte do *back-end*. Esta, por sua vez, é o ponto de comunicação entre a aplicação Web e o banco de dados, onde ocorre o armazenamento e manipulação dos dados do sistema.

A comunicação entre tais módulos é feita através do protocolo HTTP e *queries* SQL. Na Figura 3.5.1 temos um fluxograma que representa tal comunicação, que permite que o usuário use o sistema e manipule os dados necessários para executar suas tarefas.

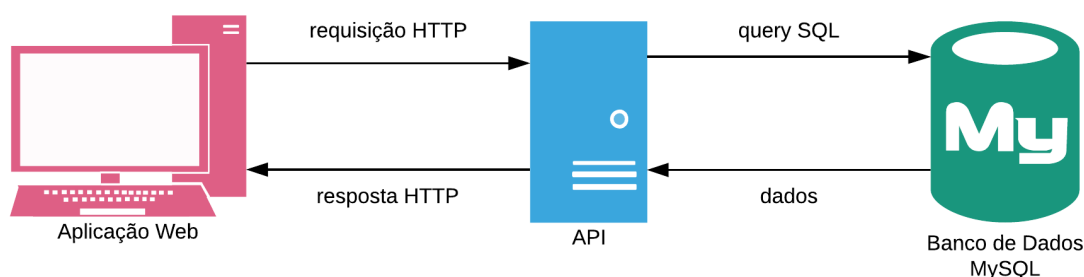


Figura 3.5.1 – Fluxograma do sistema completo.

As três partes foram implementadas em paralelo, onde uma funcionalidade era implementada por vez. A divisão das funcionalidades a serem implementadas foi feita de acordo com o grau de hierarquia entre elas. Por exemplo, para cadastrar uma proposta era necessário ter um usuário logado, logo as funcionalidades acerca do usuário foram implementadas primeiro.

3.5.1 Back-End

Para construir o *back-end* do sistema foi implementada uma API RESTful usando Node.js. O objetivo principal desta aplicação é fornecer e manipular os dados do sistema, que ficam armazenados no banco de dados. Para efetuar tal tarefa, o sistema fez uso do ORM (*Object-relational mapping*) Sequelize¹, que executa consultas ao banco de dados usando linguagem de alto nível e de melhor entendimento.

Além disso, a aplicação contém dois principais grupos de arquivos: Modelos e Controladores. Os modelos definem a estrutura dos dados, sendo tais estruturas similares às tabelas do banco de dados, e os controladores definem e executam os métodos que manipulam tais dados, realizando ações como ler, criar, atualizar e excluir.

Para acessar tais métodos, o Node.js, em conjunto com o *framework* Express², mapeia os serviços disponíveis pela aplicação usando rotas no formato URI e métodos do padrão HTTP, como GET, POST, PUT e DELETE. Assim, quando uma requisição

¹ <https://sequelize.org/>

² <https://expressjs.com/>

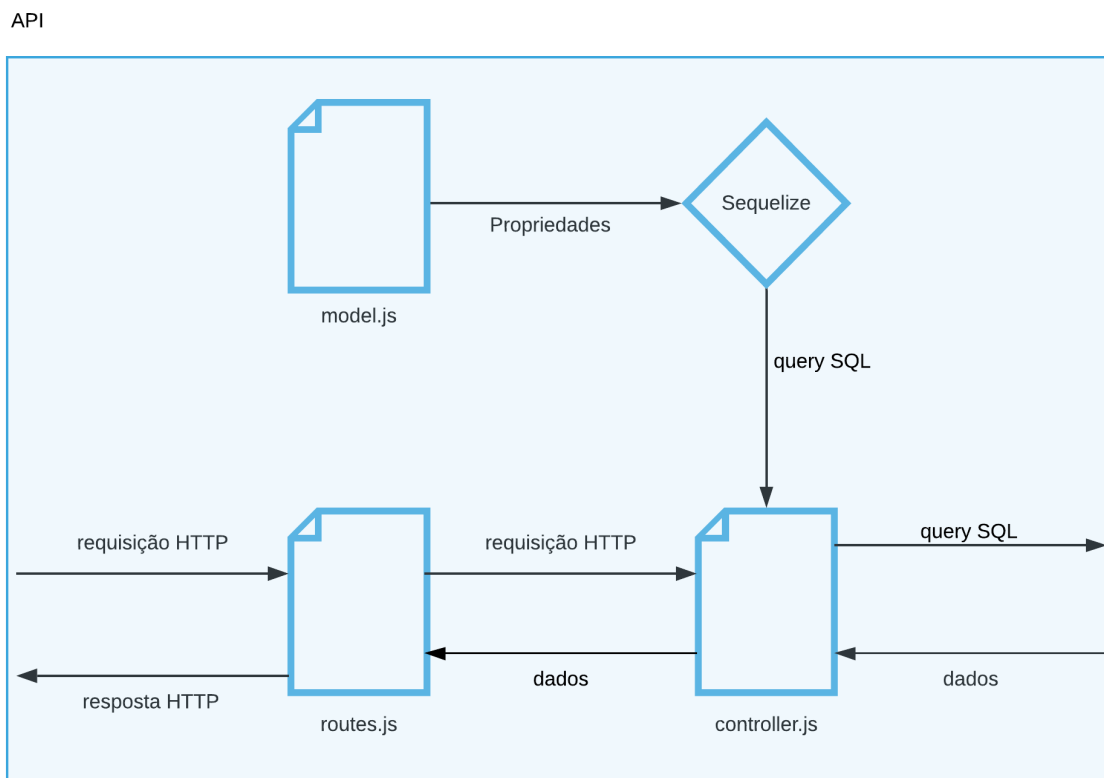


Figura 3.5.2 – Fluxograma de funcionamento da API.

HTTP é feita para a API, o mapeamento de rotas é usado para chamar o controlador responsável pelo serviço e o método que irá executá-lo.

Consequentemente, quando a API recebe uma requisição HTTP para realizar um serviço, ela efetua o passo-a-passo abaixo, também representado pela Figura 3.5.2:

1. Reconhecer o método HTTP da requisição;
2. Usar o *path* no URI para identificar o tipo de serviço que será executado e chamar o método responsável por ele;
3. Localizado no controlador, o método usa o Sequelize e os modelos já definidos para gerar uma *query* e executá-la no banco de dados;
4. Retornar o resultado da requisição.

Para demonstrar tal fluxo vamos exemplificar o processo para efetuar duas requisições: criar e ler uma proposta. Para criar uma proposta é usado o método HTTP POST, o *path* “/proposal” e as informações da proposta a ser criada, que são mandadas no *body*. Já no caso de ler uma proposta é usado o método GET e o *path* é composto pela *string* “/proposal/” concatenada ao ID da proposta em questão. As duas requisições estão exemplificadas abaixo, no fragmento de código 3.1. Ambas requisições, em caso de

sucesso na execução, retornam os dados da proposta em formato JSON (*JavaScript Object Notation*).

```
//criar proposta
POST /proposal HTTP/1.1

{
  "name": "Apresentação de TCC",
  "pax": "150",
  "deadline": "2019-12-31",
  "user_id": 4,
  "product_id": 1,
  "product_category_id": 1
}

//ler proposta
GET /proposal/9 HTTP/1.1
```

Código Fonte 3.1 – Requisição HTTP para criar e ler proposta.

Uma vez que o servidor recebe tais requisições, é feito um mapeamento de rotas, usando o método HTTP e o *path* para identificar qual serviço precisará ser feito e designá-lo ao controlador e método responsáveis por ele. A seguir, o trecho de código 3.2 mostra como foram mapeadas as rotas usadas nesse exemplo.

```
import { Router } from 'express';

import ProposalController from './app/controllers/ProposalController';

// classe que permite mapear as requisições e os métodos
const routes = new Router();

routes.post('/proposal', ProposalController.store);
routes.get('/proposal/:id', ProposalController.getProposal);
```

Código Fonte 3.2 – Trecho do arquivo routes.js.

Dessa forma, quando uma requisição chega, ela é repassada para o controlador responsável por aquele serviço. Nesse exemplo, os métodos executados pelo controlador de dados de propostas são *store* e *getProposal*, encarregados de criar uma nova proposta e ler os dados de uma já existente, respectivamente. Esses dois métodos estão apresentados no fragmento de código 3.3 abaixo:

```

async store(req, res) {
  const proposal = await Proposal.create({ ...req.body });

  if (!proposal) {
    return res
      .status(status.errors.bad_request)
      .json({ message: msg.errors.generic_error });
  }

  return res.json({ proposal });
}

async getProposal(req, res) {
  const { id } = req.params;
  const proposal = await Proposal.findOne({
    where: { id },
    [...] //outros parâmetros podem ser adicionados
  });

  return res.json({ proposal });
}

```

Código Fonte 3.3 – Trecho do controlador responsável por serviços de propostas.

No segmento de código 3.3 do controlador, é possível ver dois métodos do modelo Proposta sendo chamados, sendo eles: *create* e *findOne*. Tais métodos geram as duas *queries*, representadas abaixo (Código Fonte 3.4), respectivamente:

```

-- criar proposta
INSERT INTO `Proposal`
  (`id`,`name`,`pax`,`deadline`,`user_id`,`product_id`,`product_category_id`)
VALUES (DEFAULT,?,?,?,?,?,?);

-- ler proposta
SELECT
  `Proposal`.`id`, `Proposal`.`name`, `Proposal`.`comments`,
  `Proposal`.`pax`, `Proposal`.`deadline`, `Proposal`.`set_date`,
  `Proposal`.`total_price`, `Proposal`.`payment_conditions`,
  `Proposal`.`created_at`, `Proposal`.`user_id`, `Proposal`.`client_id`,
  `Proposal`.`product_id`, `Proposal`.`product_category_id`,
  `Proposal`.`status_id`
FROM `Proposal` AS `Proposal`
WHERE `Proposal`.`id` = '9';

```

Código Fonte 3.4 – *Queries* geradas pelo sequelize.

O Sequelize só consegue gerar essas *queries* pois existe a definição do modelo dos dados da entidade Proposta na aplicação. Tal modelo segue exatamente o mesmo padrão de dados do que o definido no banco de dados, e assim é possível que o Sequelize consiga mapear os dados do banco em objetos manipuláveis e gerar tais *queries*. Abaixo, no fragmento de código 3.5, temos a definição do modelo Proposta.

```
export default (sequelize, DataTypes) => {
  const Proposal = sequelize.define('Proposal', {
    name: DataTypes.STRING,
    comments: DataTypes.STRING,
    pax: DataTypes.INTEGER,
    deadline: DataTypes.DATE,
    set_date: DataTypes.BOOLEAN,
    total_price: DataTypes.FLOAT,
    payment_conditions: DataTypes.TEXT,
    created_at: DataTypes.DATE,
  });

  // relaciona outros modelos à proposta
  Proposal.associate = models => {
    // define uma chave estrangeira 'user_id' referente
    // ao modelo 'user', campo onde estará armazenado
    // o usuário que criou a proposta
    Proposal.belongsTo(models.User, {
      as: 'user',
      foreignKey: 'user_id',
    });

    [...]
  };

  return Proposal;
};
```

Código Fonte 3.5 – Trecho do modelo definido para a entidade proposta.

As requisições que atuam sobre dados da proposata estão listadas no Apêndice H, como fragmento da documentação da API³, a qual detalha cada requisição, informando o URI, os parâmetros e, quando pertinente, um exemplo do *body*.

3.5.2 Front-End

O módulo responsável pelo lado do cliente foi implementado usando o *framework* Angular, na versão mais recente até o momento, Angular 8. Foi construída para permitir

³ <https://documenter.getpostman.com/view/3846650/SW7aZUHz>

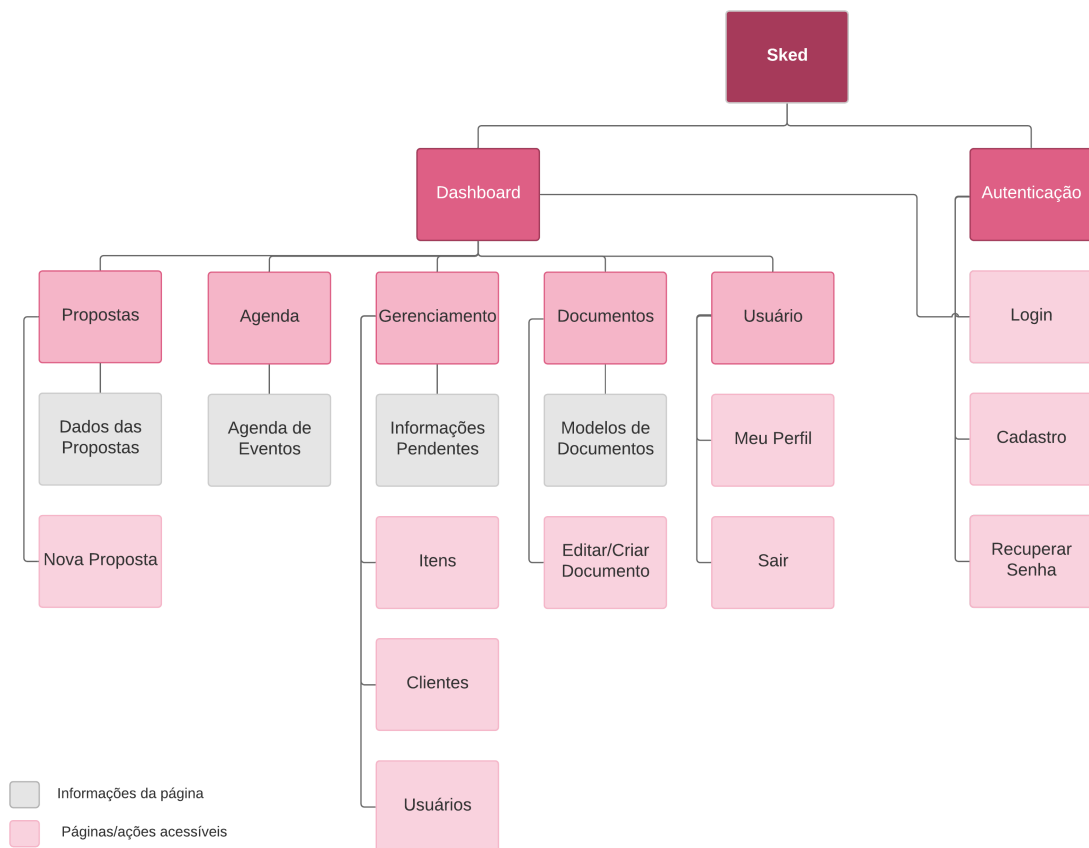


Figura 3.5.3 – Mapa da aplicação Web.

que, através da comunicação com a API, o usuário conseguisse manipular os dados do sistema de forma simples e rápida.

O resultado da implementação dessa aplicação é, então, um site dividido em áreas (módulos), onde cada área é responsável por um assunto. Uma área pode conter outras áreas e existe uma hierarquia entre elas. Abaixo podemos ver, na Figura 3.5.3, um mapa do site que mostra tais áreas, suas subáreas e a hierarquia dessa aplicação.

Para implementar essa estrutura foi usado um dos principais recursos do Angular, que é a utilização de componentes. Um componente é formado por um arquivo TS (Typescript), um HTML e um CSS. O importante é que o uso desses componentes pode ser hierarquizado, processo que acontece quando um componente se encontra dentro de outro.

Dessa forma, para efetuar a comunicação entre o cliente e o servidor, o arquivo HTML de um componente registra uma ação do usuário, que dispara um gatilho para o arquivo TS do componente. Os arquivos TS usam da classe `HttpClient`, parte integrada do Angular, para montar a requisição HTTP e enviá-la ao servidor. Também é comum a criação de arquivos nomeados *services* para efetuar as requisições HTTP. Esse funcionamento pode ser visto no fluxograma representado na Figura 3.5.4.

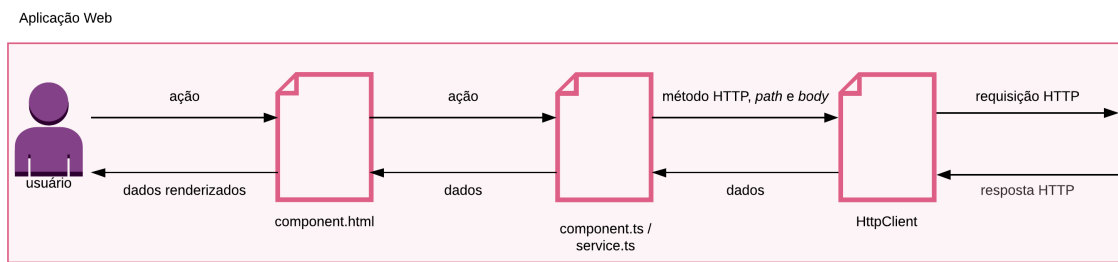


Figura 3.5.4 – Fluxograma do funcionamento da aplicação web.

Nome/Cliente	Número	Deadline	Produto/Serviço	Status
Apresentação de TCC Amanda de Souza Caetano	ADM9	30/12/19	Corporativo Soluções Simples	A Confirmar

Figura 3.5.5 – Tabela com informações e menu da proposta.

Seguindo com o exemplo da seção “back-end”, vamos usar a requisição de ler uma proposta para exemplificar tal fluxo. Na aplicação há várias funcionalidades que exigem tal requisição, mas vamos usar como exemplo a ação onde o usuário deseja ver os detalhes de uma proposta.

Tal funcionalidade encontra-se na página de propostas, que contém uma tabela que mostra o resumo das propostas já cadastradas no sistema, usando um layout de tabela. Para cada proposta há um menu de opções, e entre essas opções está a de ver detalhes da proposta. A ação do usuário é capturada quando ele clica em tal opção, como ilustrado na Figura 3.5.5.

Nesse menu, como mostrado no fragmento de código 3.6, há a definição do botão responsável por disparar o gatilho para a função “*getProposal*”, que recebe o id da proposta em questão.

```
<button (click)="getProposal(row.id)">Ver detalhes</button>
```

Código Fonte 3.6 – HTML que define o botão para a opção "Ver Detalhes" da proposta.

Tal método é definido no arquivo TS do componente (trecho 3.7). Nele, o serviço responsável pelas propostas é chamado, juntamente com o método responsável por montar

a requisição HTTP, usando a classe `HttpClient`, como mostrado no fragmento 3.8.

```
getProposal(id) {
  this.proposalService.getProposal(id).subscribe((res: any) => {
    if (res && res.proposal) {
      this.proposal = new Proposal(res.proposal);
    }
    //exibe dados da proposta
  })
}
```

Código Fonte 3.7 – Função definida no arquivo TS do componente para chamar serviço de propostas e método para requisitar dados da proposta.

```
import { environment } from 'src/environments/environment';
import { HttpClient } from '@angular/common/http';

@Injectable()
export class ProposalService {
  URL = `${environment.API}/proposal`; //formação da string no padrão URI

  constructor(
    private http: HttpClient
  ) { }

  getProposal(id) {
    const url = `${this.URL}/${id}`
    return this.http.get(url)
  }

  [...]
}
```

Código Fonte 3.8 – Serviço responsável por montar e executar a requisição HTTP para acessar detalhes da proposta.

Dessa forma, quando o serviço executa o método *getProposal* uma requisição é feita para a API, que será responsável por acessar o banco de dados e recuperar os dados daquela proposta. Quando tais dados são retornados cabe à aplicação exibi-los ao cliente, como ocorre, por exemplo, na tela "Detalhes da Proposta" (Figura 3.5.6).

3.6 Interface

Como resultado das etapas anteriores temos o sistema já implementado, cujo a interface da aplicação foi dividida em áreas, onde cada área era responsável por um de-



Propostas

Documentos

Agenda

Gerenciamento ▾

ADM ▾

Dashboard > Propostas > Informacao Da Proposta

Detalhes da Proposta ADM9

Verifique os detalhes da proposta.

Informações do Serviço

Editar

Nome	Apresentação De Tcc	
Funcionário Responsável	Usuário Admin	
Status	● Confirmado	
Produto/Categoria	Corporativo - Soluções Simples	
PAX	150	
Deadline	20/12/19	
Cronograma	Diária → 29/11/19 - 15:00 - 16:30	
Preço	R\$ 1,500.00	
Data de Entrada	17/10/19	

Dados do Cliente

Editar

Nome	Amanda De Souza Caetano	
Tipo	Pessoa Física	
CPF/CNPJ	11752288696	
Deadline	amandasc@gmail.com	
Contato	34991420744 34991420744	
Endereço	Rua Sao Judas Tadeu, 33 Carajas - Uberlandia/ MG 38408562	

Locação de Espaço

Editar

Espaços	Auditório A	
	Auditório A	R\$ 1,500.00
Preço de Venda		R\$ 1,500.00
Desconto		R\$ 0.00
Valor Total		R\$ 1,500.00

Figura 3.5.6 – Exibição de detalhes da proposta.

terminado assunto. Além disso, há duas principais sessões: autenticação e *dashboard*. Na autenticação tem-se 3 áreas: login, cadastro e recuperação de senha. Já na *dashboard* as áreas implementadas foram: *home*, propostas, agenda, documentos e gerenciamento.

Na autenticação, há formulários para que o usuário se autentique e consiga usar o sistema. Os formulários são para login, cadastro ou recuperação de senha. Além dos formulários há o nome da empresa que está usando o sistema, instruções, mensagens de confirmação, mensagens de erro e links para as outras áreas. Em dispositivos que o monitor apresenta resolução de largura maior que 767 *pixels*, também é apresentada uma área, na parte direita da tela, contendo o nome do sistema, como mostrado na Figura 3.6.1. As outras áreas podem ser vistas nas figuras 3.6.2a e 3.6.2b.

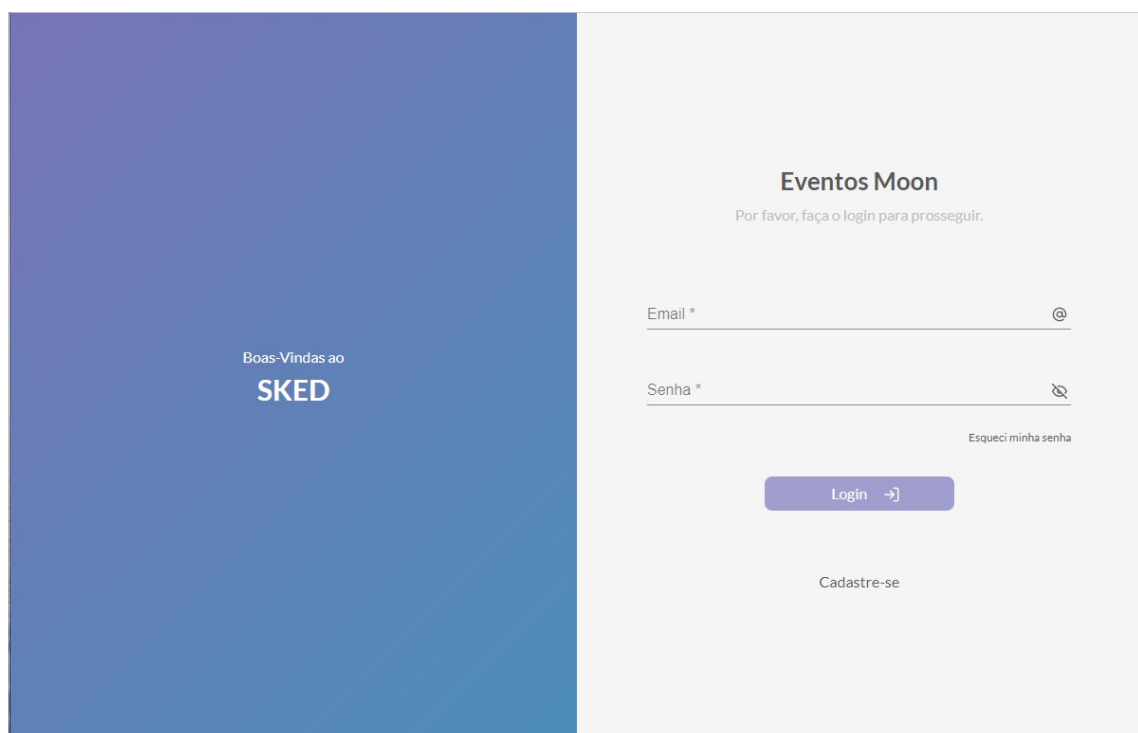


Figura 3.6.1 – Tela de login.

Uma vez que o login é efetuado, o usuário tem acesso à *dashboard*. Nesta área há um menu de navegação no topo da página. Tal menu permite acessar outras áreas da *dashboard*, assim como acessar a sessão de autenticação por meio da opção de “sair”, disponível no menu *drop-down* referente ao usuário logado. Além disso, também é comum a todas as áreas da *dashboard* uma barra superior que mostra o caminho que o usuário percorreu para estar na página atual.

Na *home* (Figura 3.6.3) são apresentadas notificações de acordo com as permissões do usuário em questão. Nas imagens a seguir o usuário que estava acessando o sistema era o administrador, logo tem acesso geral às informações. As notificações implementadas são acerca de propostas que estão com *deadline* próximos de vencer ou vencidas, além de notificar sobre usuários que se cadastraram e ainda não foram aprovados. Nas notifica-

(a) Formulário de Recuperação de Senha

(b) Formulário de Cadastro de Usuário

Figura 3.6.2 – Áreas de cadastro e recuperação de senha.

ções há *links* de redirecionamento para as áreas responsáveis pelos respectivos assuntos presentes nelas.

Figura 3.6.3 – Página de *home*.

Em propostas encontra-se um botão para cadastrar uma nova proposta e uma tabela para exibir as principais informações das propostas já cadastradas. Há também um filtro e controles de paginação para a tabela (Figura 3.6.4). Quando o usuário opta por cadastrar uma nova proposta, ele é redirecionado para uma sequência de formulários referentes aos dados daquela nova proposta. Tais formulários são similares ao protótipo e são apresentados no Apêndice I.

Já na área de documentos, é mostrado os modelos de documentos já cadastrados e a opção de cadastrar um novo documento (Figura 3.6.7). O conteúdo dos modelos de

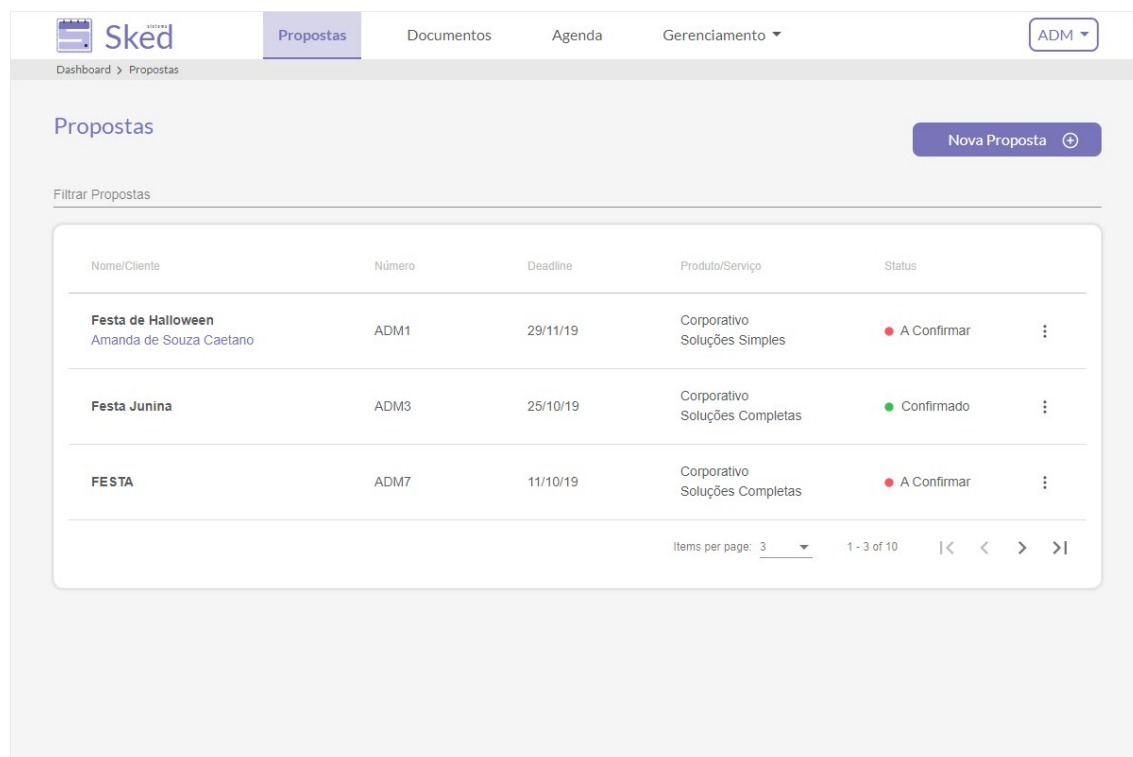


Figura 3.6.4 – Área de propostas.

documentos que já estão cadastrados é exibido em forma de prévia, com título e menu de opções. Quando o usuário opta por editar ou criar um modelo, ele é redirecionado para uma nova página (Figura 3.6.6), onde um campo de *Rich Text HTML* e um título são exibidos.

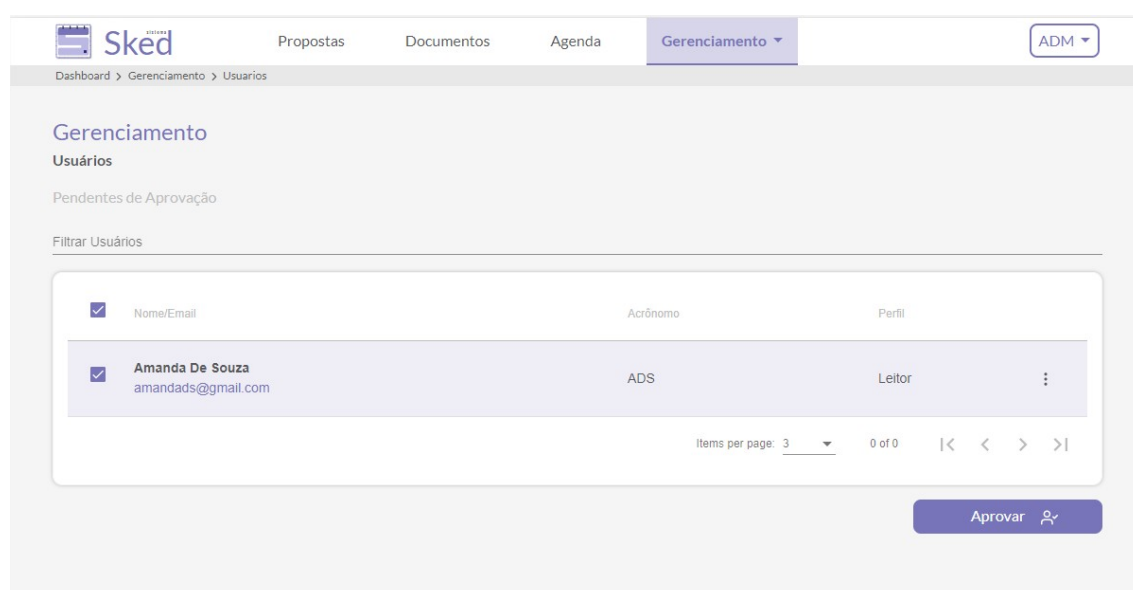


Figura 3.6.5 – Tela de gerenciamento e aprovação de usuários.

No que diz respeito à área de gerenciamento, esta é dividida em 4 subáreas: usuários, clientes, espaços e itens. É no gerenciamento que o administrador pode aprovar um

Dashboard > Documentos > Modelo

Propostas Documentos Agenda Gerenciamento ADM

Documentos

Novo Documento

Título *

Proposta

B I U S " / H₁ H₂ x₂ x² Normal Normal

A Sans Serif

Eventos Moon

Proposta de Locação de Espaço

Uberlândia, \${this.date.day} de \${this.date.month} de \${this.date.year}.

Prezada \${this.client.name},

Conforme sua solicitação, segue dados da proposta para realização de "\${this.proposal.name}" na Eventos Moon, conforme dados abaixo:

Dados do Evento:

Nome: \${this.proposal.name}
 Convidados: \${this.proposal.pax}
 Produto: \${this.proposal.product.name}
 Categoria: \${this.proposal.product_category.name}
 Cronograma: \${this.schedule}

Seus Dados:

Nome: \${this.client.name}
 CPF/CNPJ: \${this.client.cpf_cnpj}
 Email: \${this.client.email}
 Telefone: \${this.client.phone}
 Endereço: \${this.client.address}, \${this.client.number}
 Bairro: \${this.client.neighborhood}
 Cidade: \${this.client.city}/\${this.client.state}

O valor total é de \${this.proposal.price} (\${this.extended_price}), com as seguintes condições de pagamento: \${this.proposal.payment_conditions}.

Essa proposta é válida até o dia \${this.proposal.deadline}, e para aceitá-la basta entrar em contato no email \${this.user.email}.

Atenciosamente,
 \${this.user.name}.

Cancelar Salvar

Figura 3.6.6 – Área de propostas.

cadastro de usuários (Imagem 3.6.7), administrar dados dos clientes, itens e espaços da empresa.

Na agenda é exibida uma tabela, onde as linhas são os espaços disponíveis para a realização de eventos e as colunas são dias do mês ou semana (Figura 3.6.8). Um esquema de cores é usado para indicar o status da proposta.

Dessa forma, uma relação entre a aplicação e os 7 requisitos definidos na etapa de desenvolvimento (Capítulo 3.1) pode ser feita, como mostrado na Tabela 3.

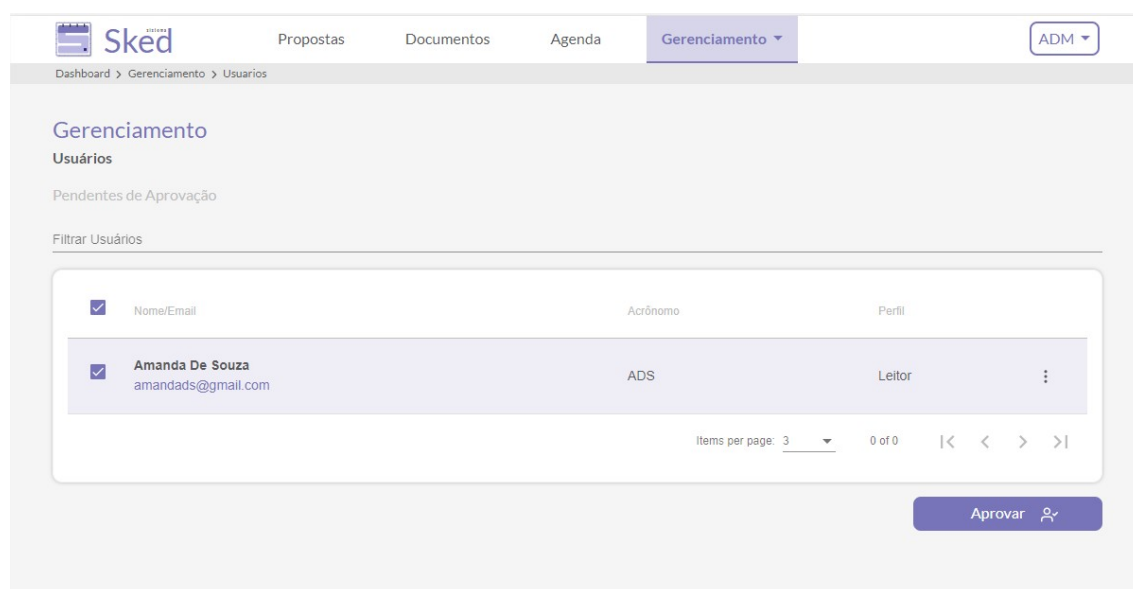


Figura 3.6.7 – Gerenciamento de Usuário: usuários pendentes de aprovação.

Requisito	Nome	Implementação
R1	Cadastro do Usuário	Completa
R2	Aprovação ou Reprovação de Usuário	Parcial
R3	Login	Completa
R4	Cadastro e Edição de Propostas	Parcial
R5	Cadastro e Edição de Modelos de Documentos	Completa
R6	Geração de Documentos	Parcial
R7	Visualização de Agenda	Completa

Tabela 3 – Relação entre implementação e requisitos.

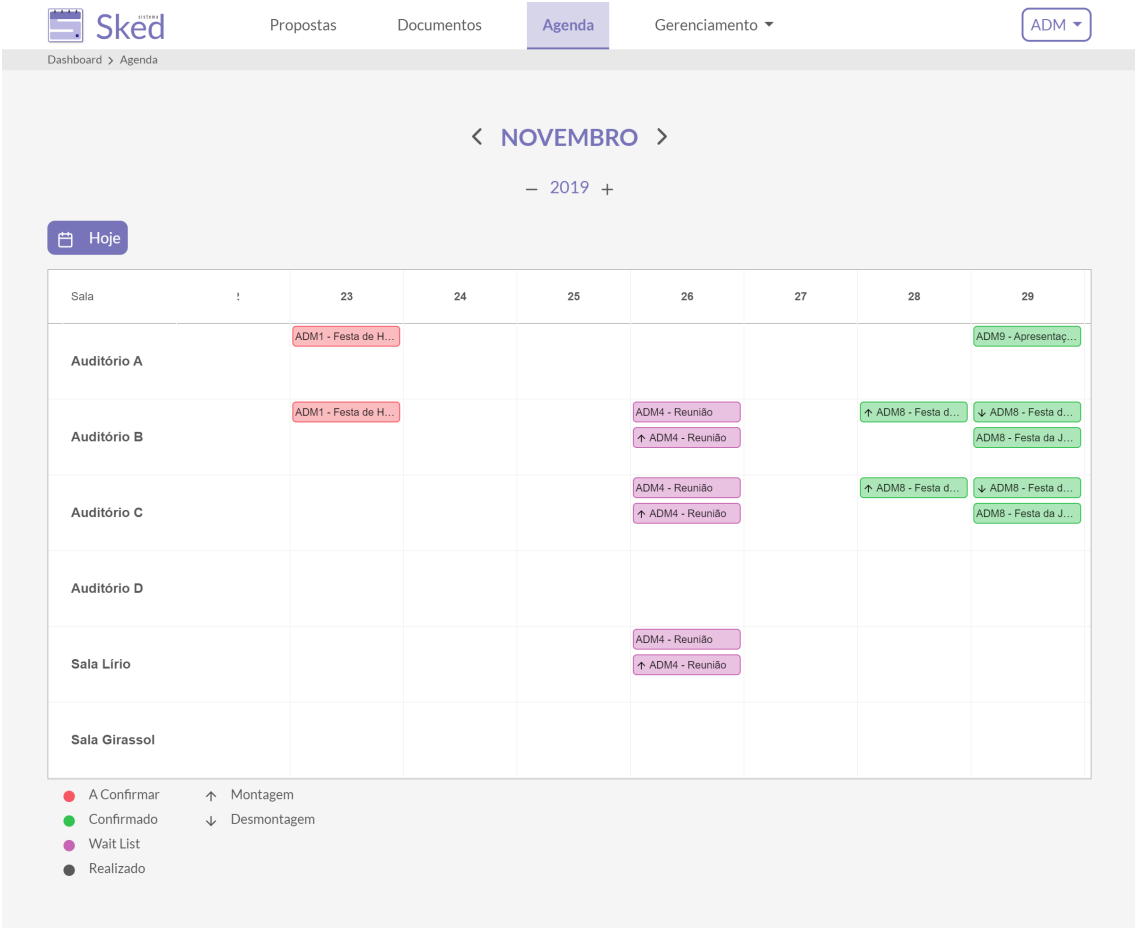


Figura 3.6.8 – Agenda de eventos.

4 Considerações Finais

No decorrer do desenvolvimento desse trabalho foi idealizado e implementado um sistema Web para auxiliar empresas e agências organizadoras de eventos. Seu objetivo é de unificar, de forma simples, funcionalidades como gerenciamento de propostas, controle de agenda e confecção de documentos. Tais funcionalidades são frequentemente usadas por empresas desse ramo, e também é de comum ocorrência o uso de múltiplas aplicações e plataformas para realizar tais tarefas acerca do mesmo evento.

Tal objetivo foi cumprido através do estudo e desenvolvimento do sistema Web SKED, uma aplicação que permite cadastro de propostas, usuários, clientes, entre outros itens que tornam possível cadastrar, editar e deletar propostas, um modelo de dados capaz de agrupar as características de um evento. Também inclui agenda e produção de documentos sobre as propostas, como contratos e orçamentos.

Dessa forma, a aplicação tem potencial para auxiliar empresas e agências organizadoras de eventos em suas atividades diárias, eliminando o uso de múltiplas plataformas e, assim, auxiliando os funcionários a cumprir suas tarefas.

Tal feito deu-se com a junção de conhecimentos adquiridos no decorrer do curso de graduação em Ciências da Computação, relacionados às disciplinas como Programação para Internet, Banco de Dados, Programação Orientada a Objetos 1 e 2, Interação Humano-Computador, Modelagem de Software, Engenharia de Software e Sistemas de Banco de Dados.

4.1 Desafios Encontrados

Durante o estudo e desenvolvimento do projeto algumas dificuldades foram enfrentadas, como esperado. A etapa inicial de entendimento do problema e modelagem de requisitos foi uma delas, onde foi necessário refazer o protótipo e reescrever os requisitos algumas vezes. Apesar do esforço, essa foi uma etapa de suma importância para as etapas posteriores, e também foi uma oportunidade de rever e colocar em práticas conceitos presentes nas disciplinas de Modelagem de Software e Engenharia de Software.

Outra dificuldade superada foi a de elaborar o *layout* do sistema. Tal processo deu-se com auxílio dos princípios de usabilidade, que ditavam alguns conceitos, elementos e características que eram necessárias na parte visual do sistema. Porém, escolhas como esquema de cores, fonte, disposição e tamanho de elementos na tela representaram uma dificuldade por se aproximarem à área de Design. Contudo, esta foi uma oportunidade de explorar tal área, além de pesquisar mais sobre Web design, estudar sobre componentes

prontos, como Angular Material, e expandir conhecimentos sobre usabilidade.

Também se fez necessário o estudo dos *frameworks* e tecnologias utilizadas, e como estas poderiam coexistir em um só sistema. Ter conhecimento prévio de Programação Orientada a Objetos foi de grande ajuda na etapa do desenvolvimento, mas ainda assim houve momentos de dificuldade para implementar algumas funcionalidades. A comunicação entre a API e o banco de dados foi uma delas, assim como encontrar a hierarquia ideal de componentes para a aplicação Web.

4.2 Trabalhos Futuros

Como próximas etapas deste trabalho compreendem-se:

- Elaboração de documentação detalhada acerca dos requisitos;
- Testar usabilidade de outras áreas do sistema;
- Continuar desenvolvendo a aplicação, adicionando novas funcionalidades, como por exemplo:
 - Adicionar área do cliente ao sistema, onde o cliente poderia visualizar seus eventos, dados e propostas.
 - Implementar envio automático dos documentos para o cliente.
 - Tornar o sistema mais inteligente com a implementação de novas funcionalidades, como por exemplo, recomendação de itens e datas de acordo com o histórico de eventos similares e criação de um ambiente virtual para montagem virtual dos espaços.
- Estudar e implementar mecanismos de segurança para troca segura de dados entre as partes do sistema.
- Fazer *deploy* da aplicação;
- Estudo de possibilidade de migrar a API monolítica para microsserviços;
- Efetuar testes com a aplicação e comparação tempo de execução de tarefas com a ferramenta e sem ela;
- Elaboração de um plano de negócio para avaliar possibilidade de transformar a aplicação em um produto e vendê-lo.

Referências

ABEOC; SEBRAE. II Dimensionamento Econômico da Indústria de Eventos no Brasil - 2013. p. 40, 2013. Citado na página 13.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. Disponível em: <<http://agilemanifesto.org/>>. Citado na página 17.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. Uniform Resource Identifier (URI): Generic Syntax, STD66, RFC 3986. 2005. Disponível em: <<https://www.rfc-editor.org/info/rfc3986>>. Citado na página 20.

CASTELEYN, S. et al. *Engineering Web Applications*. Springer Berlin Heidelberg, 2009. (Data-Centric Systems and Applications). ISBN 9783540922018. Disponível em: <<https://books.google.com.br/books?id=uOJJCubH9EC>>. Citado na página 21.

DIX, A. et al. *Human-computer Interaction*. Pearson/Prentice-Hall, 2003. ISBN 9780130461094. Disponível em: <<https://books.google.com.br/books?id=luQxui8GHDcC>>. Citado 2 vezes nas páginas 19 e 37.

GARCIA-MOLINA, H.; ULLMAN, J. D.; WIDOM, J. *Database Systems: The Complete Book*. Pearson Education, 2011. ISBN 9780133002010. Disponível em: <<https://books.google.com.br/books?id=gaEuAAAAQBAJ>>. Citado na página 18.

GOMES, A. F. *Desenvolvimento Ágil com Kanban*. 2010. Disponível em: <<https://www.devmedia.com.br/desenvolvimento-agil-com-kanban-java-magazine-84/18235>>. Citado 2 vezes nas páginas 14 e 17.

HAMMARBERG, M.; SUNDEN, J. *Kanban in Action*. 1st. ed. Greenwich, CT, USA: Manning Publications Co., 2014. ISBN 1617291056, 9781617291050. Citado na página 17.

HAZZAN, O.; DUBINSKY, Y. *Agile Software Engineering*. Springer London, 2009. (Undergraduate Topics in Computer Science). ISBN 9781848001992. Disponível em: <<https://books.google.com.br/books?id=kNFqQW3uZuC>>. Citado na página 17.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. 1990. 1-84 p. Citado na página 17.

JACOBSON, D.; BRIL, G.; WOODS, D. *APIs: A Strategy Guide: Creating Channels with Application Programming Interfaces*. O'Reilly Media, 2011. ISBN 9781449321642. Disponível em: <<https://books.google.com.br/books?id=om5tNwKW4xkC>>. Citado na página 20.

Leite Pereira, R. *SISTEMA GERENCIADOR DE EVENTOS CORPORATIVO: SIGEC*. 61 p. Tese (Doutorado) — Monografia Especialização em Tecnologia Java - Universidade Tecnológica Federal do Paraná, 2012. Citado na página 24.

MASSE, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, 2011. ISBN 9781449319908. Disponível em: <<https://books.google.com.br/books?id=eABpzyTcJNIC>>. Citado na página 20.

NIELSEN, J. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 0125184050. Citado 2 vezes nas páginas 15 e 32.

NIELSEN, J. *Designing Web Usability: The Practice of Simplicity*. Thousand Oaks, CA, USA: New Riders Publishing, 1999. ISBN 156205810X. Citado na página 15.

NIELSEN, J. *Usability 101: Introduction to Usability*. 2012. Disponível em: <<https://www.nngroup.com/articles/usability-101-introduction-to-usability/>>. Citado na página 19.

PRESSMAN, R. S. *Engenharia de Software: Uma Abordagem Profissional*. 7ª edição. ed. São Paulo: AMGH Editora Ltda, 2011. 780 p. ISBN 978-85-8055-044-3. Citado 2 vezes nas páginas 17 e 18.

Revista Eventos. *Abra festa empossa nova diretoria e anuncia que o setor fatura R\$17 bilhões*. 2017. Disponível em: <[https://www.revistaeventos.com.br/Entidades/Abra-festa-empossa-nova-diretoria-e-anuncia-que-o-setor-fatura-R\\$17-bilhoes/41](https://www.revistaeventos.com.br/Entidades/Abra-festa-empossa-nova-diretoria-e-anuncia-que-o-setor-fatura-R%2417-bilhoes/41)>. Citado na página 13.

W3C. *Introduction to HTML*. Disponível em: <https://www.w3schools.com/html/html_intro.asp>. Citado na página 21.

Apêndices

APÊNDICE A – Teste de Usabilidade:

Termo de Consentimento Livre e Esclarecido

Eu _____,
CPF _____, declaro que concordo em participar como voluntário nas atividades de testes de usabilidade do sistema web SKED, em apoio ao Projeto de Conclusão de Curso da aluna Amanda de Souza Caetano, do curso de Ciência da Computação, da Faculdade de Computação da Universidade Federal de Uberlândia, sob orientação do Prof. Dr. Rafael Dias Araújo.

Ao assinar esse documento você concorda com as seguintes afirmações:

1. Declaro também que fui informado(a) sobre os detalhes da pesquisa e seus objetivos, além do total anonimato dos participantes no resultado dos testes.
2. Estou ciente que os testes tem como objetivo avaliar o sistema, e não minha habilidade de usá-lo.
3. Entendo que posso desistir da atividade em qualquer momento que desejar, e que desta forma nenhuma dado sobre mim ou meu teste será avaliado.
4. Entendo que, caso eu conclua o teste, os dados produzidos por mim serão analisados e publicados anonimamente, e autorizo tais ações.
5. Entendo que participo como voluntário e não receberei dinheiro ou nenhuma recompensa de valor material pela minha participação.
6. Declaro que após todos esclarecimentos e tendo entendido completamente o que me foi informado, dou meu consentimento total para participar do teste.

Assinatura do Avaliador

Assinatura do Participante

Uberlândia, ____ de _____ de _____

APÊNDICE B – Teste de Usabilidade:

Instruções e Informações Importantes

Este teste foi desenvolvido para testar o sistema de gerenciamento de eventos SKED, produto do Trabalho de Conclusão de Curso da aluna Amanda de Souza Caetano da Faculdade de Computação da Universidade Federal de Uberlândia, sob orientação do Prof. Dr. Rafael Dias Araújo.

O objetivo principal é avaliar se o sistema tem uma boa usabilidade, ou seja, se é simples, limpo e intuitivo. É importante ressaltar que você não está sendo avaliado, e sim o sistema.

Por favor, leia atentamente as instruções abaixo antes de começar o teste.

Instruções:

1. O seu código de participante é _____.
2. O teste é individual e o instrutor não poderá te ajudar em momento algum.
3. Você poderá desistir a qualquer momento, basta informar ao instrutor que pretende parar.
4. Leia atentamente as descrições das tarefas antes de executá-las.
5. O teste tem duração de 25 a 40 minutos.
6. Evite rasurar a folha de tarefas de modo que prejudique a leitura da mesma, pois você precisará dela em momentos futuros.
7. Você deverá ter disponível **4 itens**, referentes à lista abaixo:
 - a) 1 computador com o sistema SKED, Questionário Demográfico e Teste de Usabilidade abertos;
 - b) 1 folha de “Termo de Consentimento Livre e Esclarecido”;
 - c) 1 folha de “Instruções e Informações”;
 - d) 1 folha de “Cenário e Tarefas”;
8. Os questionários serão respondidos por meio do computador, mas se preferir, você poderá responde-los usando folhas impressas.

APÊNDICE C – Teste de Usabilidade:

Cenário e Tarefas

Cenário: Você é um funcionário de uma empresa de eventos sociais, que presta serviços planejando e executando eventos como formaturas, casamentos, festas de confraternização de empresas, entre outros. Seu cargo consiste em entrar em contato com um cliente que deseja realizar um evento e propor ao cliente serviços e produtos possam atender a sua necessidade. Para fazer isso você usa um sistema web que gerencia tais propostas.

Tarefa 1: Um cliente entra em contato interessado em realizar um evento de confraternização no final do ano. Acesse a agenda e verifique os eventos já cadastrados entre os dias 17 e 26 de dezembro.

Tarefa 2: Volte para a sessão de propostas e inicie o processo de cadastrar nova proposta. Primeiramente, verifique os dados do cliente e siga para a próxima etapa.

Tarefa 3: Complete as informações do serviço, escolhendo por definir as datas do mesmo. O evento deverá ter o cronograma abaixo, e se uma das datas listadas não constar no sistema, você deverá adicioná-la:

Montagem: dia 22/12/2019, das 14 às 18 horas;

Diária 1: dia 22/12/2019, das 17 às 23 horas e 59 minutos;

Diária 2: dia 23/12/2019, das 0 às 12 horas;

Desmontagem: dia 23/12/2019, das 12 às 15 horas;

Tarefa 4: Na etapa de "Locação de Espaço" simule qual seria o preço total se a sala "Varanda de Ouro" fosse uma cortesia. Volte ao preço sem a cortesia e prossiga.

Tarefa 5: Agora você deverá adicionar serviços de alimentos e bebidas e depois de estacionamento no formulário de "Serviços e Produtos". Depois prossiga para a próxima etapa.

Tarefa 6: Tente excluir a proposta, mas opte por manter os dados e salve-os.

Tarefa 7: Verifique o código de sua proposta na tela de "Finalização". Diretamente dessa tela, acesse a agenda e use o código para identificar a proposta recém cadastrada na agenda. Depois finalize voltando a sessão de propostas.

APÊNDICE D – Teste de Usabilidade:

Questionário Demográfico

Questionário Demográfico

Esse é um formulário para coletar informações sobre os participantes do teste de usabilidade do sistema web SKED, como parte do Trabalho de Conclusão de Curso da aluna Amanda de Souza Caetano da Faculdade de Computação da Universidade Federal de Uberlândia, sob orientação do Prof. Dr. Rafael Dias Araújo.

***Obrigatório**

1. Código de participação *

2. Você é um profissional ligado à alguma empresa ou agência de eventos sociais? *

Marcar apenas uma oval.

- ☐ Sim
- ☐ Não
- ☐ Não sei o que são eventos sociais

3. Qual o seu grau de escolaridade? *

Marcar apenas uma oval.

- ☐ Ensino Fundamental
- ☐ Ensino Médio
- ☐ Ensino Superior Incompleto
- ☐ Ensino Superior Completo

4. Com qual frequência você utiliza o computador para acessar a internet? *

Marcar apenas uma oval.

- ☐ Raramente ou nunca
- ☐ 1 a 3 dias por semana
- ☐ 4 ou 5 dias por semana
- ☐ Mais que 5 dias por semana

5. Quando você mais utiliza o computador para acessar a internet? *

Marcar apenas uma oval.

- ☐ Horas livres
- ☐ Horas de trabalho
- ☐ Outro: _____

6. Qual seu nível de dificuldade ou facilidade ao usar o computador para acessar a internet? *

Marcar apenas uma oval.

- ☐ Muita dificuldade
- ☐ Pouca dificuldade
- ☐ Nem dificuldade nem facilidade
- ☐ Pouca facilidade
- ☐ Muita facilidade

7. Você já teve contato com algum sistema voltado para empresas e agências de eventos sociais?

*

Marcar apenas uma oval.

- ☐ Sim
- ☐ Não
- ☐ Não sei o que são eventos sociais

8. Se você respondeu 'sim' na pergunta anterior, descreva em poucas palavras o sistema e suas principais funcionalidades.

Avaliação das Tarefas

Esse é um formulário para coletar informações sobre as tarefas executadas durante o teste de usabilidade do sistema web SKED, como parte do Trabalho de Conclusão de Curso da aluna Amanda de Souza Caetano da Faculdade de Computação da Universidade Federal de Uberlândia, sob orientação do Prof. Dr. Rafael Dias Araújo.

8. Marque o nível de dificuldade que você teve para realizar cada tarefa. *

Marcar apenas uma oval por linha.

	Fácil	Médio	Difícil
Tarefa 1	☐	☐	☐
Tarefa 2	☐	☐	☐
Tarefa 3	☐	☐	☐
Tarefa 4	☐	☐	☐
Tarefa 5	☐	☐	☐
Tarefa 6	☐	☐	☐
Tarefa 7	☐	☐	☐

9. Marque a(s) funcionalidade(s) que você acha que precisam ser melhoradas. *

Marque todas que se aplicam.

- ☐ Navegar no menu superior.
- ☐ Visualizar agenda.
- ☐ Começar nova proposta.
- ☐ Inserir dados do cliente.
- ☐ Inserir informações do produto.
- ☐ Adicionar diárias à proposta.
- ☐ Inserir locação de espaço.
- ☐ Inserir produtos e serviços.
- ☐ Visualizar dados da proposta.
- ☐ Excluir ou manter proposta.
- ☐ Exibição do código da proposta.
- ☐ Outro: _____

10. Descreva o que você achou mais difícil sobre o teste (opcional).

11. Deixe uma sugestão de melhoria (opcional).

APÊNDICE F – Teste de Usabilidade: Ficha de Observações

Código: _____

Início: _____

Fim: _____

Tarefa 1:

Erros: ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Início: _____

Tarefa 2:

Erros: ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Início: _____

Tarefa 3:

Erros: ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Início: _____

Tarefa 4:

Erros: ☐☐☐☐☐☐☐☐☐☐

Início:_____

Tarefa 5:

Erros: ☐☐☐☐☐☐☐☐☐☐

Início:_____

Tarefa 6:

Erros: ☐☐☐☐☐☐☐☐☐☐

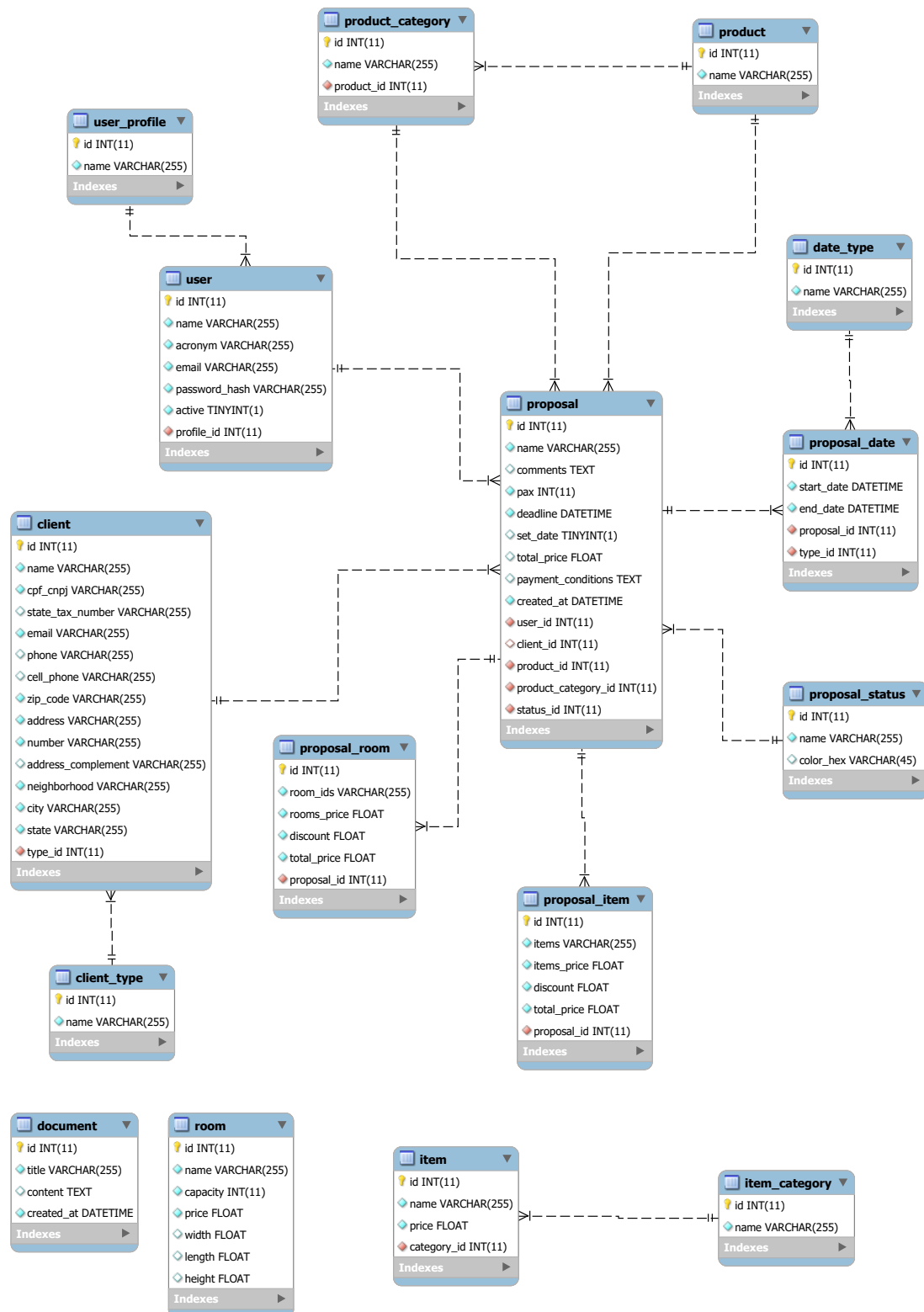
Início:_____

Tarefa 7:

Erros: ☐☐☐☐☐☐☐☐☐☐

Início:_____

APÊNDICE G – Diagrama Entidade-Relacionamento



APÊNDICE H – Documentação da API

Sked

proposal

Proposal data requests.

The requests that contains the param 'complete' returns proposal objects that has data from other models, such as room, itens, dates, user, client and product.

status

Proposal status requests.

GET /proposal-status

```
http://localhost:4080/proposal-status
```

Returns all proposal status registered in the data base.

POST /proposal-status

```
http://localhost:4080/proposal-status
```

Body raw (text)

```
{
  "name": "Status A",
  "color_hex": "#FFFFFF"
}
```

dates

Proposal dates requests.

type

Date types requests.

GET /date-type

```
http://localhost:4080/date-type
```

Returns all date types registered in the data base.

POST /date-type

```
http://localhost:4080/date-type
```

Body raw (text)

```
{
  "name": "Type B"
}
```

GET /proposal-date

```
http://localhost:4080/proposal-date
```

If year or/and month are indicated, it will return only the proposal dates that has start_date in the month or year informed, otherwise it will return all proposal dates.

GET /proposal-date/{{proposal.id}}

```
http://localhost:4080/proposal-date/{{proposal.id}}
```

Returns data from the document model where proposal id is the same as the one in the request URL.

POST proposal-date

```
http://localhost:4080/proposal-date
```

Body raw (text)

```
{
  "proposal_id": 8,
  "dates": [
    {
      "proposal_id": 8,
      "start_date": "2019-01-12T14:00:00.000Z",
      "end_date": "2019-01-12T20:00:00.000Z",
      "type_id": 2
    },
    {
      "proposal_id": 8,
      "start_date": "2019-01-12T08:00:00.000Z",
      "end_date": "2019-01-12T12:00:00.000Z",
      "type_id": 1
    }
  ]
}
```

PUT proposal-date/{{proposal_date.id}}

```
http://localhost:4080/proposal-date/{{proposal_date.id}}
```

Updates data from the proposal date where id is the same as the one in the request URL.

Body raw (text)

```
{
  "proposal_id": 8,
  "dates": [
    {
      "proposal_id": 8,
      "start_date": "2019-01-12T14:00:00.000Z",
      "end_date": "2019-01-12T20:00:00.000Z",
      "type_id": 2
    },
    {
      "proposal_id": 8,
      "start_date": "2019-01-12T08:00:00.000Z",
      "end_date": "2019-01-12T12:00:00.000Z",
      "type_id": 1
    }
  ]
}
```

rooms

Proposal rooms requests.

GET /proposal-room

```
http://localhost:4080/proposal-room
```

Returns data from all proposal rooms registered in the data base.

GET /proposal-room/{{proposal.id}}

```
http://localhost:4080/proposal-room/{{proposal.id}}
```

Returns data from proposal room where proposal_id is same as the one informed in the request URL.

POST /proposal-room

```
http://localhost:4080/proposal-room
```

Creates a new relations between room and proposal. The field 'room_ids' is a string that represents a list of ids from the rooms that will be related to the proposal. It is important keep the JSON format inside the string content.

Body raw (text)

```
{
  "room_ids": "[1, 2]",
  "rooms_price": 3500,
  "discount": 200,
  "total_price": 3300,
  "proposal_id": 1
}
```

PUT /proposal-room/{{proposal-room.id}}

```
http://localhost:4080/proposal-room/{{proposal-room.id}}
```

Updates data from proposal room where id is the same as the one informed in the URL.

Headers

Content-Type application/json

Body raw (application/json)

```
{
  "room_ids": "[1, 2]",
  "rooms_price": 3500,
  "discount": 600,
  "total_price": 2900,
  "proposal_id": 1
}
```

item

Proposal items requests.

GET /proposal-item

http://localhost:4080/proposal-item

Returns data from all proposal items registered in the data base.

GET /proposal-item/{{proposal.id}}

http://localhost:4080/proposal-item/{{proposal.id}}

Returns data from proposal itens where proposal_id is same as the one informed in the request URL.

POST /proposal-item

http://localhost:4080/proposal-item

Creates a new relations between items and a proposal. The field 'items' is a string that represents a list of categories and it's selected itens that will be related to the proposal. It is important keep the JSON format inside the string content.

Body raw (application/json)

```
{
  "items": "[{category_id: 1, items_id: [1,2]}, {category_id: 5, items_id: [30]}]",
  "items_price": 8220,
  "discount": 20,
  "total_price": 8200,
  "proposal_id": 1
}
```

PUT /proposal-item/{{proposal-item.id}}

http://localhost:4080/proposal-item/4

Updates data from proposal item where id is the same as the one informed in the URL.

Body raw (application/json)

```
{
  "items": "[{category_id: 1, items_id: [1,2]}, {category_id: 5, items_id:
[30]}]",
  "items_price": 8220,
  "discount": 220,
  "total_price": 8000,
  "proposal_id": 1
}
```

GET /proposal

http://localhost:4080/proposal?complete={{complete}}

Returns all proposals registered in the data base.

Params

complete	{{complete}}
	(OPTIONAL) Boolean value. If 'true', request returns proposal object that has data from other models, such as room, itens, dates, user, client and product.

GET /proposal/deadline

http://localhost:4080/proposal/deadline?complete={{complete}}&days={{days_range}}

Returns proposals where deadline are going to expired in a range of {{days_range}} from request date.

Params

complete	{{complete}}
	(OPTIONAL) Boolean value. If 'true', request returns proposal object that has data from other models, such as room, itens, dates, user, client and product.
days	{{days_range}}
	(Required)

GET /proposal/expired

http://localhost:4080/proposal/expired?complete={{complete}}&user_id={{user.id}}

Returns proposals where deadline is expired (before request date). If param 'user_id' is informed, it return only the expired proposals where user_id is the same.

Params

complete	{{complete}}
	(OPTIONAL) Boolean value. If 'true', request returns proposal object that has data from other models, such as room, itens, dates, user, client and product.

user_id	{{user.id}}
----------------	-------------

GET /proposal/{{proposal.id}}

```
http://localhost:4080/proposal/{{proposal.id}}?complete=true
```

Returns data from proposal where id is the same as the one informed in the URL

Params

complete	true
-----------------	------

(OPTIONAL) Boolean value. If 'true', request returns proposal object that has data from other models, such as room, itens, dates, user, client and product.

POST /proposal

```
http://localhost:4080/proposal
```

Body raw (text)

```
{
  "name": "Apresentação de TCC",
  "pax": "150",
  "deadline": "2019-12-31",
  "user_id": 4,
  "product_id": 1,
  "product_category_id": 1
}
```

PUT /proposal/{{proposal.id}}

```
http://localhost:4080/proposal/{{proposal.id}}?complete=false
```

Updates data from proposal where id is the same as the one informed in the URL

Params

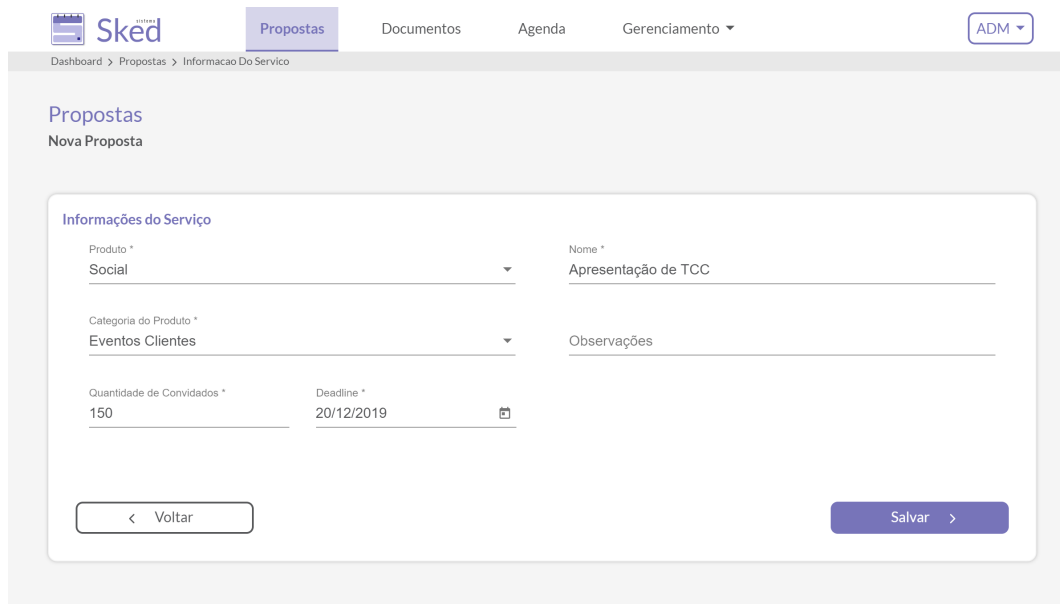
complete	false
-----------------	-------

(OPTIONAL) Boolean value. If 'true', request returns proposal object that has data from other models, such as room, itens, dates, user, client and product.

Body raw (text)

```
{
  "pax": "150",
  "deadline": "2019-12-31",
}
```

APÊNDICE I – Apresentação de Formulários Implementados

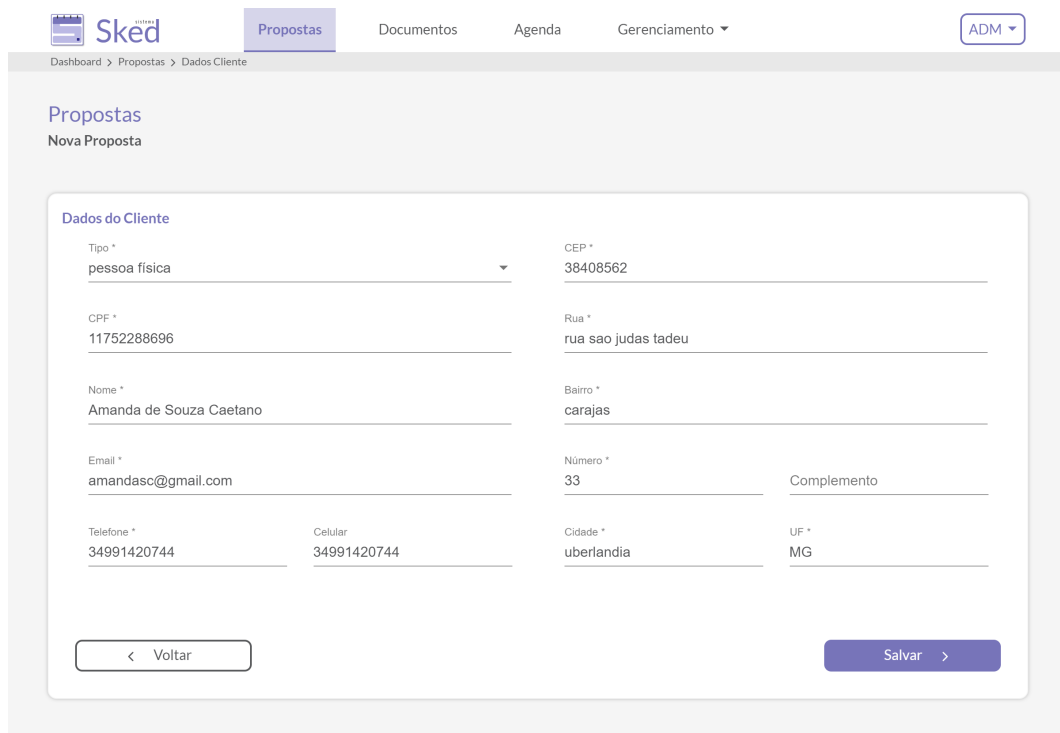


The screenshot shows the 'Propostas' (Proposals) section of the Sked system. The breadcrumb trail is 'Dashboard > Propostas > Informacao Do Serviço'. The page title is 'Propostas' and the subtitle is 'Nova Proposta'. The form is titled 'Informações do Serviço' and contains the following fields:

- Produto ***: Dropdown menu with 'Social' selected.
- Nome ***: Text input with 'Apresentação de TCC'.
- Categoria do Produto ***: Dropdown menu with 'Eventos Clientes' selected.
- Observações**: Text area.
- Quantidade de Convidados ***: Text input with '150'.
- Deadline ***: Text input with '20/12/2019' and a calendar icon.

At the bottom of the form are two buttons: '< Voltar' (Back) and 'Salvar >' (Save).

Figura I.0.1 – Formulário 1: Informações do Serviço.



The screenshot shows the 'Propostas' (Proposals) section of the Sked system. The breadcrumb trail is 'Dashboard > Propostas > Dados Cliente'. The page title is 'Propostas' and the subtitle is 'Nova Proposta'. The form is titled 'Dados do Cliente' and contains the following fields:

- Tipo ***: Dropdown menu with 'pessoa física' selected.
- CEP ***: Text input with '38408562'.
- CPF ***: Text input with '11752288696'.
- Rua ***: Text input with 'rua sao judas tadeu'.
- Nome ***: Text input with 'Amanda de Souza Caetano'.
- Bairro ***: Text input with 'carajas'.
- Email ***: Text input with 'amandasc@gmail.com'.
- Número ***: Text input with '33'.
- Complemento**: Text input.
- Telefone ***: Text input with '34991420744'.
- Celular**: Text input with '34991420744'.
- Cidade ***: Text input with 'uberlandia'.
- UF ***: Text input with 'MG'.

At the bottom of the form are two buttons: '< Voltar' (Back) and 'Salvar >' (Save).

Figura I.0.2 – Formulário 2: Dados do Cliente.

Sked

Propostas Documentos Agenda Gerenciamento ADM

Dashboard > Propostas > Datas

Propostas
Nova Proposta

Datas

Definir datas?
Sim Não

Montagens 1 Diárias 1 Desmontagens

Diária 1

Data *	Início *	Fim *
30/11/2019	09:00	11:00

2h

Adicionar Diária

Voltar Salvar

Figura I.0.3 – Formulário 3: Datas.

Sked

Propostas Documentos Agenda Gerenciamento ADM

Dashboard > Propostas > Locacao Espaco

Propostas
Nova Proposta

Locação de Espaços

Espaços * Sala Lirio, Sala Girassol Desconto 80 \$

Detalhamento

Sala Lirio	R\$ 1,000.00
Sala Girassol	R\$ 1,000.00
Valor de Venda	R\$ 2,000.00
Desconto	- R\$ 80.00
Valor Total	R\$ 1,920.00

Voltar Salvar

Figura I.0.4 – Formulário 4: Locação de Espaço.