UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

DANILO JOSE DA SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA CONVERSÃO DE MODELOS UTILIZANDO A METODOLOGIA BDUF

Joinville
2022

DANILO JOSE DA SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA CONVERSÃO DE MODELOS UTILIZANDO A METODOLOGIA BDUF

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Dr. Gian Ricardo Berkenbrock

Joinville
2022

AGRADECIMENTOS

Ao Criador, por sempre me dar forças e permitir vencer todos os obstáculos encontrados.

Aos meus pais, que sempre me apoiaram e deram suporte.

A Universidade Federal de Santa Catarina que me proporcionou um ensino de qualidade com políticas de permanência estudantil que agregaram na minha conclusão da graduação.

Ao Dr. Alexandre Miers Zabot que foi meu orientador de iniciação cientifica durante parte de minha graduação e que sem dúvidas agregou no meu processo de desenvolvimento. Também ao Laboratório de Computação Cientifica - LCC que me proporcionou uma infraestrutura com equipamentos de qualidade em todo o período de minha iniciação cientifica.

Ao meu orientador de trabalho de conclusão de curso, Dr. Gian Ricardo Berkenbrock que sempre se mostrou paciente e disponível em todo o processo de elaboração deste trabalho.

A coordenadora do curso de Engenharia Mecatrônica, Dr.(a) Tatiana Renata Garcia, que além de ter sido minha orientadora de estágio, sempre se mostrou humana e presente no esclarecimento de dúvidas.

Aos meus queridos amigos: Acir Marconato Júnior, Daniela Rodrigues Ferreira, Jéssica Medalha Ferri, Thais Baena Moura, Samuel Possamai e entre outros aqui não listados que tornaram meu período de graduação mais agradável, leve e feliz.

E por fim, a mim mesmo.

RESUMO

Desde o advento da web 2.0 surgiu uma preocupação em construir softwares cada vez mais próximo às expectativas do cliente e que tornasse a experiência ao usuário cada vez mais útil, agradável e relevante. O tradicional método em cascata surgiu nesse intuito, porém, devido à natureza mutável dos softwares e uma crescente necessidade por ambientes mais dinâmicos onde a interação entre as partes interessadas se tornou fundamental no processo de construção do software, esse método passou a ser considerado obsoleto em projetos difíceis de prever. Com isso, algumas metodologias para desenvolvimento de software surgiram no intuito de preencher essa lacuna que o método tradicional em cascata apresenta, sendo uma delas, a BDUF (Big Design Up Front). Essa metodologia é uma evolução do método tradicional em cascata, contudo, aqui se reconhece que no processo de construção de software podem surgir ajustes nas etapas anteriores, de modo que, se torne viável a implementação do software. Neste trabalho, é proposto o uso da metodologia BDUF no intuito de gerar uma documentação rica em detalhes da interface do usuário para uma aplicação web de gestão orientado a modelos, e logo em seguida, seja implementada na etapa de construção de software.

Palavras-chave: Experiência do Usuário. UX. Interface do Usuário. UI. Big Design Up Front. BDUF.

ABSTRACT

Since the advent of web 2.0, there has been a concern to build software closer to customer expectations and to make the user experience more and more useful, pleasant and relevant. The traditional waterfall method emerged for this purpose, however, due to the changing nature of software and a growing need for more dynamic environments where the interaction between stakeholders has become fundamental in the software construction process, this method has come to be considered obsolete in projects difficult to predict. With this, some methodologies for software development have emerged in order to fill this gap that the traditional waterfall method presents, one of them being BDUF (Big Design Up Front). This methodology is an evolution of the traditional waterfall method, however, here it is recognized that in the software construction process, adjustments may arise in the previous steps, so that the implementation becomes viable. In this work, it is proposed to use the BDUF methodology in order to generate a documentation rich in details of the user interface for a model-oriented management web application, which, soon after, is implemented in the software construction stage.

Keywords: User eXperience. UX. User Interface. UI. Big Design Up Front. BDUF.

LISTA DE FIGURAS

Figura 1 – Metodologia tradicional em cascata 14
Figura 2 – Metodologia BDUF 16
Figura 3 – Metodologia ágil 17
Figura 4 – Comparativo entre as metodologias ágil e cascata 18
Figura 5 – Etapas no processo de levantamento de requisitos de software . . . 23
Figura 6 – Padrão de documento SyRS e SRS segundo ISO/IEC/IEEE. . . (2011) 26
Figura 7 – Wireframe e protótipo da plataforma para o caso do usuário não autenticado 30
Figura 8 – Figura exemplo no levantamento e ordenação logica dos eventos . . 36
Figura 9 – Figura exemplo na ordenação dos wireframes 37
Figura 10 – Figura exemplo na identificação das entradas 37
Figura 11 – Figura exemplo na identificação das saídas 38
Figura 12 – Figura exemplo da sintaxe JSX do React 39
Figura 13 – Estrutura da aplicação MOMS 41
Figura 14 – Estrutura base da aplicação 43
Figura 15 – Listagem de possíveis eventos 44
Figura 16 – Acesso sem login 45
Figura 17 – Cadastro de novo usuário 46
Figura 18 – Wireframe da interface do usuário 47
Figura 19 – Wireframe para representação genérica de um elemento 48
Figura 20 – Protótipo para o caso do usuário não logado 49
Figura 21 – Paleta de cores para os principais elementos da plataforma 49
Figura 22 – Protótipo para o caso de troca de tema 50
Figura 23 – Protótipo para o caso de login ou cadastro 51
Figura 24 – Protótipo para o caso de exclusão de um modelo 52
Figura 25 – Protótipo para o caso de cópia de um modelo 52
Figura 26 – Protótipo para o caso de edição de uma sequência 53
Figura 27 – Protótipo para o caso de criação ou edição de um projeto 53
Figura 28 – Protótipo para o caso de conversão de um modelo 54
Figura 29 – Protótipo para o caso de visualizar alertas 54
Figura 30 – Protótipo para o caso de adicionar um modelo ou sequência ao projeto 55
Figura 31 – Protótipo para o caso de consultar ou excluir um membro do grupo . 56
Figura 32 – Protótipo para o caso de incluir um membro ao grupo 56
Figura 33 – Protótipo para o caso de edição ou criação de um grupo 57
Figura 34 – Implementação para o caso do usuário não logado 58

Figura 35 – Implementação para o caso de login ou cadastro 59
Figura 36 – Implementação para o caso de acesso aos modelos públicos e privados 60
Figura 37 – Implementação para o caso de acesso às sequências privadas . . . 61
Figura 38 – Implementação para o caso de acesso aos projetos privados 62
Figura 39 – Implementação para o caso de adicionar um modelo ou sequência de conversão a um projeto 63
Figura 40 – Modelagem de eventos para login de diferentes níveis de usuário . . 77
Figura 41 – Modelagem de eventos para operações básicas com um modelo . . 78
Figura 42 – Modelagem de eventos para operações básicas com um projeto . . 79
Figura 43 – Modelagem de eventos para gerenciamento de grupos e usuários . 80
Figura 44 – Protótipo para o caso de exclusão de sequência, projeto ou grupo . 81
Figura 45 – Protótipo para o caso de consultar os modelos e sequências de um projeto 82
Figura 46 – Implementação para o caso de mudança de tema 83
Figura 47 – Implementação para o caso de mudança de coluna simples para dupla 84
Figura 48 – Implementação para o caso de um novo projeto 85

LISTA DE ABREVIATURAS E SIGLAS

UX	User eXperience
UI	User Interface
ISO	International Organization for Standardization
AI	Arquitetura da Informação
UML	Unified Modeling Language
BDUF	Big Design Up Front
QA	Quality Assurance
XP	Extreme Programming
DSDM	Dynamic Systems Development Method
SADT	Structured Analysis Design Technique
SyRS	System Requirements Specification
SRS	Software Requirements Specification
SPA	Single Page Applications
SEO	Search Engine Optimization
MOMS	Model-Oriented Management System

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Objetivo 12
1.1.1 Objetivo Geral 12
1.1.2 Objetivos Específicos 12
1.2 Estrutura do Trabalho 12

2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 Metodologia BDUF 19
2.1.1 Brainstorm 19
2.1.2 Requisitos de Software 22
2.1.2.1 Elicitação 23
2.1.2.2 Análise 24
2.1.2.3 Especificação 25
2.1.2.4 Validação 26
2.1.2.5 Gerenciamento de Requisitos 27
2.1.3 Projeto 27
2.1.3.1 Wireframes 29
2.1.3.2 Protótipos 30
2.1.4 Construção de Software 30
2.1.4.1 Princípios de Organização 32
2.2 Trabalhos Relacionados 34

3 MATERIAIS E MÉTODOS 35
3.1 Validação de Requisitos 35
3.2 Interface do Usuário 38
3.3 Implementação da aplicação 39

4 RESULTADOS E DISCUSSÕES 43
4.1 Validação de Requisitos 43
4.2 Interface do Usuário 46
4.2.1 Wireframes 46
4.2.2 Protótipo 48
4.2.2.1 Login e Cadastro 50
4.2.2.2 Modelos 51
4.2.2.3 Sequências de conversão 52
4.2.2.4 Projetos 53
4.2.2.5 Grupos 55

4.2.3 Software 57
4.2.3.1 Login e Cadastro 59
4.2.3.2 Modelos 60
4.2.3.3 Sequências de Conversão 61
4.2.3.4 Projetos 62
4.2.3.5 Grupos 64
4.3 Limitações e Discussões 64
4.4 Considerações finais 64

5 CONCLUSÕES 65

REFERÊNCIAS 67

APÊNDICE A 70

APÊNDICE B 77

APÊNDICE C 81

1 INTRODUÇÃO

Desde a sua origem a internet passou por transformações que marcaram a maneira como o usuário interage com as páginas web e segundo Patil (2021) tais transformações evolutivas são definidas como web 1.0, web 2.0 e web 3.0. Na web 1.0 os conteúdos eram apenas estáticos e não era possível a interação do usuário com as páginas web, enquanto na web 2.0 que utilizamos atualmente temos um conceito de colaboração e distribuição de conteúdo, onde páginas antes estáticas passaram a ser dinâmicas e interativas aos usuários permitindo uma gama de possibilidades como postagem de comentários, envio de arquivos, etc.

De acordo com REISSWITZ (2008), estamos caminhando rumo a web 3.0 onde passaremos para um conceito de web “semântica” que virá para proporcionar interatividade entre homem e máquina, melhorando as linguagens de programação com sites e aplicações inteligentes que baseiam suas pesquisas nos comportamentos de seus usuários. Patil (2021) segue na ideia de que a web “semântica” virá para organizar as informações possibilitando pesquisas inteligentes tornando a experiência do usuário mais relevante, útil e agradável.

Para Patil (2021), a web 2.0 é o marco de inovações em relação à experiência do usuário, principalmente no desenvolvimento front-end. Esse termo experiência do usuário (do inglês User eXperience - UX), apesar de ser muito utilizada atualmente em desenvolvimento de software e abordada na norma ISO 9241-210, é uma expressão originalmente criada na década de 90 por Donald Norman na Apple e possui um conceito bem amplo. UX engloba tudo que vise melhorar a experiência do usuário, seja em um aplicativo, em um serviço ou qualquer outra coisa que o usuário possa interagir (AGNI, 2016).

Essa experiência do usuário é subjetiva, sendo que um usuário pode ter uma experiência diferente de outro no uso de uma mesma aplicação, pois temos uma influência de dois agentes principais: o fator humano e os fatores externos (TEIXEIRA, 2014). O fator humano está diretamente relacionado ao usuário, ou seja, se há uma familiaridade no uso daquela tecnologia, se as informações contidas na aplicação são interpretadas de forma correta, de seu interesse no uso da aplicação e dentre outros fatores.

Conforme indicado por TEIXEIRA (2015), o UX designer tem a sua parcela de influência, pois a experiência do usuário é uma soma de fatores que devem ser considerados. Fatores como uma sequência fluida de interação do usuário com a aplicação somados a um layout amigável e suave em seu carregamento são agentes que podem definir se determinada aplicação pode ser mais popular do que outra.

GOMES (2015) aponta que essa experiência do usuário é sustentada por quatro elementos fundamentais: arquitetura de informação, design visual, design de interação e usabilidade. Na arquitetura de informação o objetivo é organizar as informações, de modo que se torne intuitivo ao usuário o uso de determinada aplicação. No design visual a atenção se dá aos elementos visuais que vão desde a paleta de cores, imagens até elementos que permitam o desenvolvimento de um layout agradável.

O design de interação é responsável pela criação e desenvolvimento de conteúdo e sua preocupação está em como as informações serão estruturadas na interface do software. Para GOMES (2015), essa interação entre o software e o usuário deve ser vista como uma conversa onde o design de interação deve exercer controle sobre ela com sequências de falas e quantidade de conteúdos que possibilite ao usuário completar a sua tarefa desejada.

Por fim, a usabilidade é definida por Teixeira (2014) como a “facilidade com que as pessoas podem utilizar uma ferramenta ou objeto para realizarem uma tarefa”. Na norma ISO 9241-210 temos uma definição mais técnica onde usabilidade é tido como “o grau em que um produto é usado por determinados usuários para atingir objetivos específicos com eficácia, eficiência e satisfação em determinado uso específico” (ISO Central Secretary, 2019).

Essa preocupação com a experiência do usuário voltado a aplicações web se deu na web2.0, visto que, páginas antes estáticas passaram a ter iteratividade. Essa evolução foi possível graças a apresentação do JavaScript pela Netscape Communications em 1995. O JavaScript é uma linguagem de programação executada no navegador, ou seja, é uma linguagem do lado do cliente e que permitiu aos programadores melhorar a interface e interatividade do usuário com os elementos dinâmicos (KULESZA et al., 2018).

Essa constante evolução das aplicações web trouxe a necessidade em desenvolver metodologias que tornassem mais simples e sujeito a menos erros o processo de construção do software. O método em cascata é uma dessas metodologias, onde sua principal característica é a divisão das tarefas em etapas predeterminadas, executadas de forma sequencial, sendo que essa metodologia está cada vez mais em desuso devido as suas limitações (ANDERSON et al., 2010).

Anderson et al. (2010) destaca que outras metodologias surgiram de modo a suprir deficiências que o método em cascata apresenta, uma delas é a Big Design Up Front abreviada pela sigla BDUF. Na BDUF se assume que as etapas presentes no método em cascata serão realizadas, porém, devido a imprevistos e a natureza desconhecida do software essas etapas podem ser aperfeiçoadas ou adaptadas na etapa final de desenvolvimento do software.

Em ambientes mais dinâmicos, onde se deseja obter entregas constantes do projeto é empregado uma metodologia mais contemporânea conhecida como métodos ágeis. Essa metodologia difere das anteriores, pois, nela se trabalha com etapas de entregas menores, sendo que, uma etapa não precisa ser totalmente finalizada para que outra comece. Nos métodos ágeis, o importante é a troca de opiniões e informações, de modo a se chegar em um produto mais próximo do desejado (ANDERSON et al., 2010).

1.1 OBJETIVO

Para um melhor delineamento do trabalho é proposto os seguintes objetivos:

1.1.1 Objetivo Geral

Desenvolver uma aplicação web com finalidade de possibilitar o uso de uma plataforma para transformação de modelos suportados. Para tanto, será utilizado a BDUF (do inglês Big Design Up Front) uma metodologia de desenvolvimento de software que concentra os estudos na interface e experiência do usuário.

1.1.2 Objetivos Específicos

- Levantar os requisitos de software;
- Com base nos requisitos de software, realizar a modelagem de eventos de modo a identificar a interação do usuário com a aplicação;
- Projetar a interface do usuário com base nos requisitos de software e modelagem de eventos;
- Utilizar bibliotecas e frameworks para a linguagem de programação javascript;
- Realizar testes com uma API emulada.

1.2 ESTRUTURA DO TRABALHO

O objetivo dessa seção é a de fornecer uma visão geral a respeito da estrutura do trabalho. Assim sendo, no primeiro momento (capítulo 2) é abordado na fundamentação teórica algumas metodologias de desenvolvimento de software, apresentando suas características, seus pontos positivos e negativos quando aplicados a determinadas categorias de projetos. Por consequência, havendo um embasamento teórico a respeito de algumas das principais metodologias utilizadas, é então, apresentado a justificativa na escolha da metodologia BDUF.

No capítulo 3, que trata dos materiais e métodos onde é exposto às ferramentas e tecnologias utilizadas no desenvolvimento da aplicação, sendo contempladas

as etapas de levantamento de requisitos, elaboração da interface do usuário e programação da aplicação.

Na sequência (capítulo 4), os resultados e discussões são expostos ao leitor, assim como o processo empregado no uso da BDUF. Todo esse estudo serviu para a elaboração da interface do usuário para que no fim seja obtido sequencias de wireframes para o desenvolvimento da aplicação.

Por fim, na conclusão são discutidos os resultados obtidos e também sobre o que era esperado entregar após a conclusão deste trabalho. Também as perspectivas de evolução dos recursos presentes na plataforma, onde também algumas funcionalidades que serão implementadas na medida que novas versões da aplicação estiverem disponíveis.

2 FUNDAMENTAÇÃO TEÓRICA

No desenvolvimento do software do presente trabalho são estudadas algumas metologias de modo a encontrar uma melhor abordagem para sua construção. Anderson et al. (2010) enfatiza não haver uma metodologia definitiva para o desenvolvimento de um software, pois uma metodologia pode se adequar ou não para determinada categoria de projeto. Como a variedade e peculiaridade dos projetos, deve-se saber dosar qual metodologia é mais adequada e isso pode variar conforme a experiência dos profissionais de software em considerar o que torna uma metodologia boa ou ruim. Atualmente há diversas metodologias utilizadas pelos profissionais de software dentre elas podemos destacar: metodologia em cascata, BDUF (do inglês Big Design Up Front) e metodologia Ágil.

Na metodologia em cascata (figura 1) é proposto que o software seja construído em uma sequência de etapas principais, sendo: requisitos de negócios, requisitos de projeto, desenvolvimento e implantação sendo que cada uma dessas etapas devem ser totalmente concluída antes do início da próxima. A metodologia é eficiente para produtos que tenham o mínimo de complexidade de projeto e engenharia, ou que são implementações simples de soluções bem definidas (ANDERSON et al., 2010).

Figura 1 – Metodologia tradicional em cascata

Fonte: Anderson et al. (2010) pág. 103

Anderson et al. (2010) destaca que apesar de a metodologia em cascata ser mais intuitiva para as pessoas e a mais amplamente empregada, ela apresenta algumas deficiências. Um delas é explicar as incertezas ou eventos desconhecidos relacionados ao projeto, pois, aqui se presume que cada etapa pode ser completa e perfeitamente concluída antes que a próxima etapa comece. Como as etapas são totalmente isoladas entre si, os grupos de colaboradores são isolados uns dos outros, ou seja, o grupo responsável pelo brainstorming que elaboram os requisitos de software não colaboram com o grupo que arquitetam e projetam (ANDERSON et al., 2010).

> Essa abordagem força os estágios de engenharia e garantia de qualidade (quality assurance - QA) a absorver quase todos os efeitos dos riscos e incógnitas que surgem durante o projeto. Como

> o planejamento, a arquitetura e o design do produto já estão ostensivamente completos, não há opção de alterá-los porque o dinheiro para eles já foram gasto e os recursos foram alocados para outras coisas. Isso deixa para os engenheiros descobrir como explicar os inevitáveis problemas e incógnitas – e fazê-lo no orçamento e cronograma alocados antes que os problemas e incógnitas fossem identificados. (Anderson et al. (2010), tradução nossa)

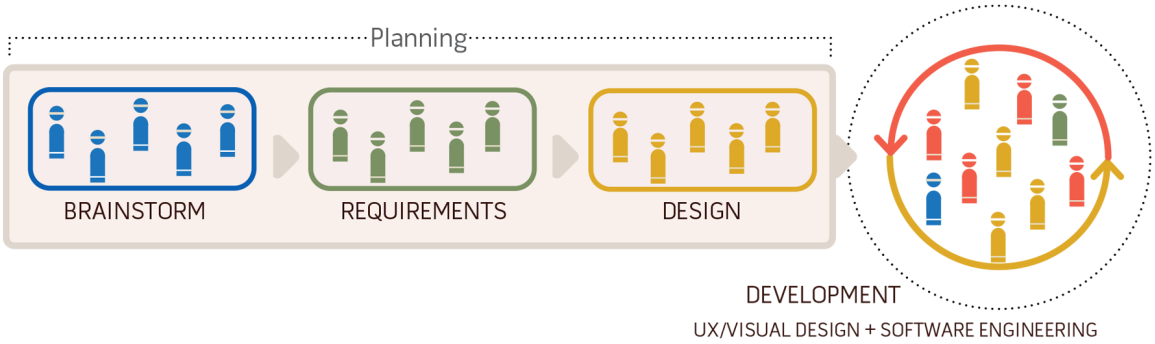
A metodologia Big Design Up Front (BDUF) se assemelha ao método em cascata, porém como o nome sugere há um esforço maior no processo de design antes das etapas de engenharia e controle de qualidade. Na BDUF temos um pensamento central que cada etapa pode ser aperfeiçoada antes que a próxima comece, nela se sugere que a etapa de design pode ser finalizada antes do início do desenvolvimento do produto. Nessa fase do design inicial normalmente não há assistência da equipe de desenvolvimento, que devem estar presentes para avaliar os custos e a viabilidade de certas ideias e contribuindo com mais ideias e perspectivas (ANDERSON et al., 2010).

Para Anderson et al. (2010) a BDUF oferece uma oportunidade de se trabalhar intensamente com os stakeholders, ou seja, com as partes interessadas, de modo a traduzir suas necessidades de negócios e usuários em um conjunto de requisitos de design de experiência visual. Toda essa documentação gerada torna os projetos BDUF mais fáceis de vender, pois, permitem que as partes interessadas aprovem um projeto de design menor antes de se comprometerem com o projeto completo maior. Porém, assim como no método em cascata, aqui temos a forte tendência de se chegar a conclusões apenas com esforços iniciais no design do produto, isso acarreta um efeito de sobrecarga na equipe de engenharia, pois terão que absorver essa carência de detalhes não explorados nos processos anteriores.

> O problema com o BDUF é que ele geralmente mantém os engenheiros na bancada, no escuro e fora da conversa por muito tempo. Um grande número de incógnitas, problemas e oportunidades podem ser identificados e resolvidos por um exaustivo processo de projeto inicial, mas até que a engenharia comece, um vasto e rochoso mar do desconhecido permanece inexplorado. Além disso, sem o benefício da contribuição dos engenheiros, podem ser feitas promessas e estimativas que mais tarde se mostrarão impossíveis ou irreais, levando a decepções e aumento tensões. (Anderson et al. (2010), tradução nossa)

Contudo, conforme pode ser observado na figura 2, temos um avanço significativo da BDUF em relação ao método em cascata. Na etapa de desenvolvimento do software se reconhece que até certo ponto os projetos iniciais precisarão ser adaptados e/ou modificados e de que incógnitas e problemas poderão surgir ao longo do processo. Com isso, há uma colaboração entre os engenheiros de requisitos, designers de UX e da equipe de desenvolvimento nesse último estágio (ANDERSON et al., 2010).

Figura 2 – Metodologia BDUF



Fonte: Anderson et al. (2010) pág. 105

A metodologia ágil é um conjunto de metodologias que vieram como alternativas ao modelo tradicional em cascata para construção de suas aplicações. Ela se popularizou em 2001 quando diversos especialistas em desenvolvimento de software que representavam metodologias já existentes (XP do inglês extreme programming, SCRUM, DSDM do inglês dynamic systems development method e etc) estabeleceram objetivos comuns a elas (PRIKLADNICKI; WILLI; MILANI, 2014). Com isso surgiu o Manifesto Ágil (BECK et al., 2001b), onde seus conceitos, chaves são:

- **Indivíduos e interações** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação de contratos;
- **Responder a mudanças** mais que seguir um plano.

O manifesto ágil deixa claro que: indivíduos e interações; software em funcionamento; colaboração com o cliente e responder a mudanças, possuem um maior grau de importância. Em um primeiro momento, pode parecer que os métodos ágeis negligenciem alguns processos, no entanto, Anderson et al. (2010) destaca que a metodologia ágil reconhece as incertezas e mudanças inerentes ao projeto, com isso, ter flexibilidade e uma colaboração com todas as partes envolvidas são de extrema importância.

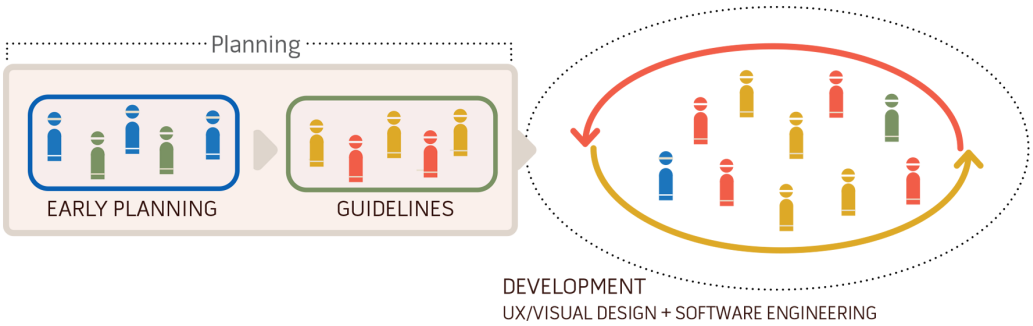
Além disso, os autores do manifesto ágil definiram 12 princípios (BECK et al., 2001a) que visam maximizar o trabalho das equipes e os resultados gerados aos clientes, sendo elas:

1. Ter como prioridade a satisfação do cliente através de entregas de valor contínuas e rápidas;
2. Ser receptivo às alterações nos requisitos em qualquer fase do processo. Aliás, ambientes mutáveis são empregados em toda etapa do projeto para entregar ao cliente vantagem competitiva;
3. Realizar entregas frequentes (do produto ou serviço) no menor período possível;
4. Manter a colaboração das partes envolvidas em todo o projeto, diariamente;

5. Fornecer o ambiente, as ferramentas e o suporte necessários aos indivíduos do projeto, além de acreditar neles para realizar as atividades;
6. Estimular a comunicação pessoal, que transmite as informações necessárias ao time de colaboradores, sendo o meio mais eficiente;
7. Software funcionando é a principal medida de progresso;
8. Processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem conseguir manter um ritmo constante indefinidamente;
9. Manter atenção frequente à excelência de design e técnica eleva ou aprimora a agilidade;
10. Simplicidade – a arte de maximizar a quantidade de trabalho não feito – é essencial;
11. Equipes auto-organizáveis propiciam os melhores designs e arquiteturas, além de atenderem aos requisitos do projeto,
12. Por meio de intervalos regulares, o time de colaboradores do projeto reflete sobre como melhorar a sua eficiência e eficácia para otimizar o seu comportamento.

Conforme pode ser notado nos 12 princípios, não há um único foco na metodologia ágil, embora como seu nome sugere, o principal objetivo de sua aplicação está em aumentar a agilidade dos processos. Amarras burocráticas comuns em projetos que usam o método em cascata, são retiradas na metodologia ágil, isso porque ao flexibilizar as operações, é possível trabalhar em um fluxo de processos mais aderente às mudanças, tornando o processo de desenvolvimento do produto mais fluido. Isso é exemplificado na figura 3, onde a política de envolver as partes interessadas no processo de desenvolvimento do produto, acarretam uma interação de feedbacks contínuos que favorecem o sucesso do projeto.

Figura 3 – Metodologia ágil



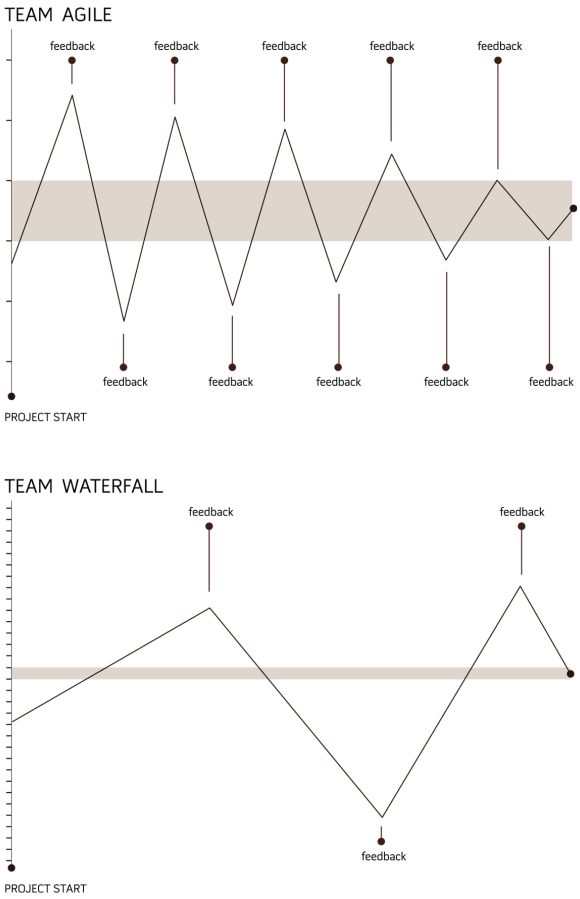
Fonte: Anderson et al. (2010) pág. 108

Para mostrar o quão eficaz a metodologia ágil é em frente ao tradicional modelo em cascata, Anderson et al. (2010) traça um paralelo na figura 4 onde é comparado duas

equipes igualmente qualificadas utilizando essas duas metodologias. Na metodologia tradicional em cascata, logo no início do projeto a equipe responsável fara uma tentativa de prever detalhes do processo de desenvolvimento, caso ocorra do projeto ter sido levado para uma direção errada é feito um novo planejamento mais amplo que o anterior de modo a identificar como o produto precisa ser modificado para alcançar o sucesso. Mesmo que esse novo planejamento favoreça chegar mais perto do sucesso, não será atingido totalmente, pois haverá apenas mais duas etapas de planejamento no ciclo de construção do produto.

Por outro lado, com métodos ágeis, temos um comportamento que vai alternando ao longo do caminho de construção do produto que tentem próximo à linha de sucesso. Em cada parada e avaliação da situação, caso seja notado que as diretrizes do projeto estão fora do rumo é feito uma correção. Com isso, cada vez que é feito uma correção há um avanço no projeto e há uma melhor capacidade nas tomadas de decisões sobre ajustes de curso.

Figura 4 – Comparativo entre as metodologias ágil e cascata



Fonte: Anderson et al. (2010) pág. 112

2.1 METODOLOGIA BDUF

Para o desenvolvimento do software do presente trabalho é utilizado a metodologia BDUF, conforme mostrado na Fig. 2 nesse método antes da implementação da aplicação são realizados sequências de estudos e levantamentos com passos bem definidos, de modo a se chegar em um design funcional. Nessa metodologia, inicia-se com o levantamento de ideias onde são discutidas as premissas básicas da aplicação. Toda essa discussão e levantamentos resultará em um business case, um documento que fornece informações relevantes sobre a aplicação, como seus benefícios, recursos necessários, prazos, etc (TOTVS, 2021).

A etapa seguinte trata no desenvolvimento dos requisitos que compreendem em: levantamentos, analises, especificação e validação dos requisitos de software (SAWYER; KOTONYA, 2001). De acordo com Fraser et al. (2007) o desenvolvimento dos requisitos de software é a etapa mais complexa, pois caso seja mal realizada é a mais difícil de se corrigir e também a que mais impacta nos demais processos. Um dos recursos utilizados nessa etapa de análise é sua a modelagem conceitual, onde é empregado uma modelagem de eventos como recurso. Nesse processo, se mostra intuitivamente como ocorre a interação do usuário com o sistema, sendo que, na modelagem de eventos, o objetivo é analisar apenas o que o usuário vê em qualquer momento específico (LAU, 2020).

O processo seguinte trata do projeto para desenvolvimento da interface gráfica onde a experiência do usuário é um dos enfoques. Na BDUF, inicialmente, temos a característica de dedicar mais tempo na etapa de design. Essa abordagem tem suas vantagens, pois fornece um estudo rico em detalhes que contribuem para uma imagem mais definida de como será o produto final.

Com a documentação de requisitos de software e a interface do usuário finalizadas, se incia o processo de desenvolvimento. Conforme já citado, a BDUF passou por algumas evoluções em relação ao método tradicional em cascata, pois, na fase de desenvolvimento do software, caso seja necessário, podem ocorrer ajustes nas documentações fornecidas. Isto ocorre, no intuito de tornar viável a implementação da aplicação, quando considerado prazos e custos envolvidos.

Para um melhor entendimento da metodologia utilizada neste trabalho, nos parágrafos que se sucedem é explicado com mais detalhes cada etapa da BDUF (consultar 2).

2.1.1 Brainstorm

Brainstorm é um estrangeirismo, que pode ser traduzido para o português como “tempestade de ideias”. Nessa técnica é explorado a capacidade criativa de um grupo previamente selecionado de pessoas, no intuito de atingir algum objetivo ou solucionar

alguma categoria de problema. O brainstorming possui várias aplicações, sendo frequentemente usadas no(a): desenvolvimento de novos produtos; geração de novas ideias para campanhas de publicidade; resolução de problemas; gerenciamento de processos; gerenciamento de projetos, etc. Na técnica é proposto que o grupo reunido utilize a diversidade de pensamentos e experiências para gerar ideias inovadoras, sugerindo qualquer ideia ou pensamento que vier a mente (DEBASTIANI, 2015).

Aqui se espera reunir o maior número de ideias, propostas e visões possíveis que levem ao objetivo comum para solução do problema, onde os participantes desse grupo deve ser diversificado e composto por pessoas de diferentes setores e especialidades. Debastiani (2015) aponta que a premissa base do brainstorming é que nenhuma ideia deve ser descartada ou considerada absurda, onde todas ideias devem ser ouvidas, discutidas e registradas para se chegar em um compilado de ideias que levem a uma solução efetiva. O brainstorming normalmente possui três fases principais, sendo: encontrar os fatos, gerar ideias e definir uma solução (DEBASTIANI, 2015).

A fase que compreende em encontrar os fatos, define o problema, explorando todas as informações que se conhecem a seu respeito. Debastiani (2015) destaca que em algumas vezes será necessário subdividir o problema em partes menores no intuito de realizar uma análise mais aprofundada. Na segunda fase, temos a geração de ideias, aqui a proposta é que os participantes coloquem sob discussão as suas ideias e visões a respeito do problema proposto. Por fim, na última fase é gerado uma avaliação de cada sugestão, até que se chegue em um consenso quanto a solução a ser adotada (DEBASTIANI, 2015).

Segundo Debastiani (2015) é vital que o brainstorming seja realizado sem pressa, com tempo suficiente para que cada etapa do processo seja finalizado corretamente. Também, é importante deixar claro o proposito da sessão, onde cinco regras devem ser respeitadas durante o período de trabalho:

Suspensão de julgamentos: é deixado claro que criticas as ideias apresentadas estão proibidas, pois, esse tipo de comportamento gera inibições ou perdas de objetivo da sessão.

Quantidade é importante: quanto maior o número de ideias maior a quantidade de hipóteses e que por consequência pode-se levar a uma solução, aqui a ideia é que quantidade gera qualidade.

Liberdade total: nenhuma ideia é considerada absurda ao ponto de ser ignorada ou desprezada, já que pode servir de base para criação de novas ideias.

Modificar e combinar ideias: é permitido gerar novas ideias a partir de outras já apresentadas por qualquer membro do grupo, no entanto, as ideias originais devem ser mantidas mesmo após a apresentação da nova.

Igualdade de oportunidades: aqui o objetivo é que o mediador assegure que

todos os membros tenham as mesmas oportunidades de apresentar suas ideias, sem que membros mais extrovertidos acabem se sobressaindo.

Para Debastiani (2015), o mediador da sessão deve deixar claro para os membros da sessão de brainstorming o tema e lhes dar um pleno entendimento do problema tratado, visto que, essa ação evitara que caminhos equivocados sejam trilhados durante as discussões. Durante a sessão tudo deve ser devidamente anotado para que nada se perca e também o facilitador deve considerar algumas recomendações durante o brainstorming:

- A etapa de geração de ideias deve possuir um tempo máximo de duração;
- Após a geração de ideias é preciso definir como será a estrutura de apresentação delas, que pode ser de forma estruturada ou não estruturada. A forma não estruturada é a mais utilizada, nela os membros da sessão apresentam suas ideias na medida que vão surgindo, nessa modalidade o moderador deve assegurar que as regras estão sendo respeitadas e que todos poderão participar. Na forma estruturada o moderador define uma sequência para a manifestação, de modo que, cada participante contribua com uma ideia em cada rodada.
- O tópico a ser analisado dever ser colocado sempre na forma de pergunta e se deve assegurar que todos os membros da sessão entenderam a pergunta;
- Após o questionamento ser colocado para o grupo, deve ser concedido um tempo para que os membros pensem a respeito, de modo que, em seguida apresente as suas ideias;
- As ideias devem estar expostas a todos, de modo que, gere um estímulo no pensamento criativo do grupo e também para se evitar duplicidade ou desvios delas.
- A sessão de brainstorming deve ser encerrada apenas quando os participantes estiverem com as ideias esgotadas ou quando o tempo estipulado para a sua duração estiver acabado.

Com a etapa de geração de ideias finalizada, Debastiani (2015) ressalta a importância de deixar claro a abrangência e significado de todas as hipóteses levantadas. Isso no intuito de que todos os membros da sessão entendam o real proposito do brainstorming realizado. Para isso ocorrer, o moderador da sessão deve abordar cada ideia apresentada e questionar se os membros possuem dúvida, também caso seja necessário, o próprio autor da ideia pode ficar encarregado de esclarecer eventuais dúvidas aos membros.

Após o término da sessão de brainstorming, o moderador ou um grupo encarregado estará com toda lista de ideias e questionamentos. Esse responsável deverá organizar o material coletado e isso pode ocorrer com uma classificação de acordo com seu tema e categoria. Para cada categoria de ideias, procurar as similares e combiná-las para eliminar eventuais duplicidades. Ao fim deste processo, as melhores

ideias permanecerão, de modo que, sejam posteriormente melhoradas e aproveitadas efetivamente (DEBASTIANI, 2015).

2.1.2 Requisitos de Software

Em sua definição formal, requisitos de software é tudo que vise expressar as necessidades e restrições colocadas em um produto de software, de modo que, contribua para a satisfação de determinada aplicação (SAWYER; KOTONYA, 2001) (MACHADO, 2018) (BRACKETT, 1990). Em relação aos requisitos, Sawyer e Kotonya (2001) aponta que temos uma distinção de propriedade que se caracterizam como: parâmetros do processo e os parâmetros do produto. Parâmetros do processo trata essencialmente de restrições de desenvolvimento do sistema. Os parâmetros do produto são requisitos do sistema, sendo classificados como: requisitos funcionais e não funcionais. Um requisito funcional é tido como uma capacidade do sistema e o requisito não funcional atua na restrição da solução ou como um requisito de qualidade.

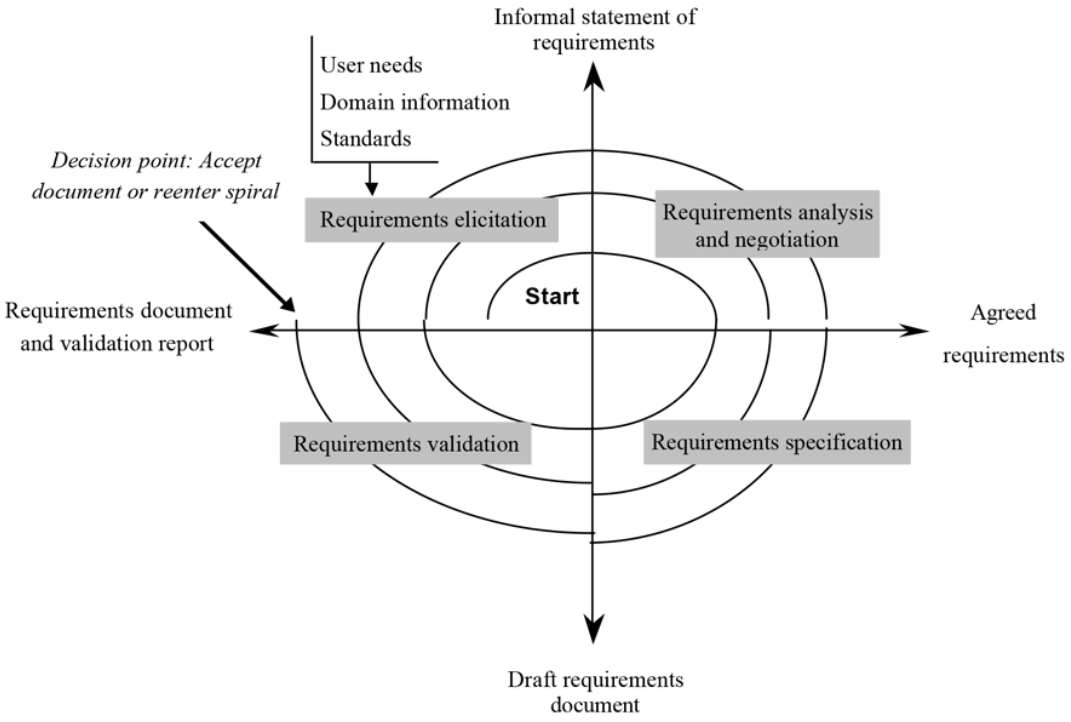
Nos Requisitos de sistema são englobados as partes interessadas, sendo algumas delas: os usuários que irão utilizar o sistema, normalmente se tratam de um grupo heterogêneo de pessoas; o cliente, ou seja, as pessoas que encomendaram o sistema ou que representam o mercado-alvo; os analistas de mercado que atuam como um cliente substituto de modo a estabelecer o que o mercado precisa. Domínios de aplicação de bancos ou transporte publico, por exemplo, são regulamentados, com isso, dependendo do requisito ou aplicação é necessário ter aprovação de outra parte interessada que são as autoridades reguladoras e por fim o desenvolvedor do sistema (SAWYER; KOTONYA, 2001).

Sawyer e Kotonya (2001) enfatiza que uma propriedade essencial em todos os requisitos é que eles devem ser verificáveis e que tais requisitos devem ser declarados explicitamente e, quando apropriado, quantitativamente, sendo importante evitar que a descrição do requisito seja vaga para não dar margem para uma interpretação subjetiva. Os requisitos devem ser guiados pela necessidade de atingir os objetivos gerais do produto, e para que esse foco não se desvie é necessário ter um business case que norteie objetivamente quais os benefícios que serão proporcionados. Business case é um documento que norteia a tomada de decisão e fornece todas as informações relevantes sobre uma determinada ideia, nesse documento é esclarecido os benefícios, recursos necessários, prazos e como se dará a implementação do projeto (TOTVS, 2021).

Com o business case guiando os objetivos gerais do projeto estabelecido, se inicia o trabalho de elicitação, análise, documentação e validação dos requisitos do sistema, sendo essas etapas cruciais para a compreensão do problema e seu provável custo de implementação (SAWYER; KOTONYA, 2001). Na Fig. 5 é mostrado conceitualmente como essas diferentes etapas se compõe iterativamente, Sawyer

e Kotonya (2001) destaca que encerrar o processo de engenharia de requisitos prematuramente pode resultar em um efeito prejudicial nas próximas fases do projeto, sendo que, as etapas que compõe o processo de engenharia de requisitos devem se repetir até que seja gerado um documento de especificação aceitável.

Figura 5 – Etapas no processo de levantamento de requisitos de software



Fonte: Sawyer e Kotonya (2001) pág. 38

2.1.2.1 Elicitação

Segundo Belgamo e Martins (2000), a elicitação de requisitos é a primeira etapa na engenharia de requisitos, porém com uma característica de ser iterativo, de modo que, as demais fases podem conter essa etapa no seu processo. Para início da elicitação é necessário identificar os usuários finais, eles são importantes, pois poderão agregar no processo com seus conhecimentos e experiências. Conforme destacado por Belgamo e Martins (2000), a elicitação de requisitos é uma fase delicada, pois caso seja efetuada uma escolha errada do usuário, pode resultar em tempo gasto de forma desnecessária com prejuízo financeiro para os clientes. Uma vez, que as fontes de requisitos forem identificadas as partes interessadas podem articular seus requisitos, Sawyer e Kotonya (2001) aponta algumas técnicas de elicitação que podem ser utilizadas, sendo as principais:

Entrevistas: amplamente utilizada na elicitação de requisitos, aqui o cuidado que se deve ter é em entender suas vantagens, limitações e sua condução no processo.

Cenários: são importantes, pois, fornecem um contexto na elicitação de requisitos do usuário ao estruturar questionamentos sobre as suas tarefas desempenhadas.

Protótipos: quando alguns requisitos não estão definidos intuitivamente, os protótipos se mostram uma ferramenta valiosa, pois assim como os cenários, eles podem fornecer um contexto sobre quais informações os usuários precisam fornecer.

Reuniões facilitadoras: nas reuniões facilitadoras, o objetivo é desempenhar um trabalho em grupo, afim que gere uma soma de ideias. Sawyer e Kotonya (2001) enfatiza que essa abordagem pode resultar em um conjunto mais rico e consistente de ideias do que em entrevistas, por exemplo, isso ocorre, pois, os requisitos conflitantes são identificados desde o incio pelas partes interessadas. No entanto, deve-se ater a ideia de que as reuniões devem possuir um facilitador, para evitar situações onde algumas opiniões tenham maior peso do que outras dependendo do cargo ou senioridade.

Observação: ela permite que o engenheiro de requisitos possa aprender sobre as tarefas realizadas pelo usuário, pois, permite uma observação de sua interação com o sistema.

2.1.2.2 Análise

Para Sawyer e Kotonya (2001) a visão tradicional da análise de requisitos era limitada apenas em uma modelagem conceitual, onde alguns métodos de análises como SADT (do inglês Structured Analysis Design Technique) são alguns dos métodos utilizados. Como a premissa base da análise de requisitos está em detectar e resolver conflitos entre requisitos, além é claro, em descobrir os limites do sistema e como ele deve interagir com seu ambiente, Sawyer e Kotonya (2001) lista que além da modelagem conceitual, mais algumas etapas importantes no processo de análise de requisitos devem ser consideradas, sendo:

Classificação dos requisitos: requisitos podem ser classificados como funcionais ou não funcionais, também podem derivar de outros requisitos e possuir um grau de prioridade, onde, quanto maior sua prioridade mais essencial é o requisito para atender aos objetivos gerais do sistema. Também podem ser classificados em relação ao seu escopo, ou seja, o quanto o requisito afeta o sistema e seus componentes, além é claro, que requisitos são mutáveis, pois estão sujeitos a mudanças durante o ciclo de vida do software ou mesmo durante o processo de desenvolvimento.

Modelagem conceitual: modelos conceituais visam auxiliar na compreensão do problema, existem vários modelos que podem ser desenvolvidos, isso inclui: fluxos de dados, modelos de estado, rastreamentos de eventos, interações

do usuário, modelos de objetos e entre outros. Fatores como a natureza do problema, requisitos de processo impostos pelo cliente, a experiência do engenheiro de requisitos e a disponibilidade de métodos e ferramentas são alguns dos fatores que influenciam na escolha do modelo conceitual.

Arquitetura de software e alocação de requisitos: a arquitetura de software se trata de um modelo sob o qual um sistema pode ser desenvolvido e abrange como as suas partes estão organizadas e quais serão os seus componentes responsáveis por realizar determinado conjunto de tarefas. A alocação de requisitos permite uma análise mais detalhada dos requisitos, pois dado que um conjunto de requisitos tenha sido alocado a um determinado componente, eles podem ser melhor analisado para descobrir como ocorre sua interação com os demais componentes.

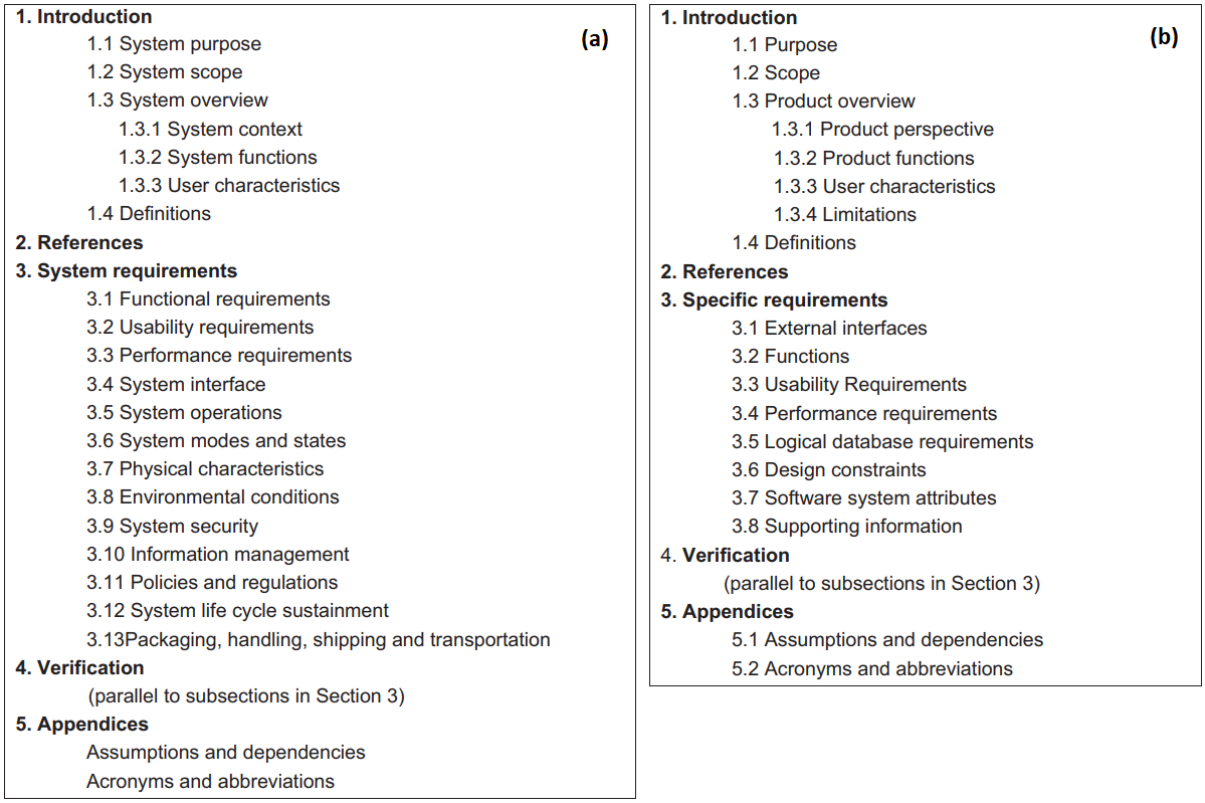
Negociação de requisitos: aqui há uma preocupação em como resolver os conflitos entre requisitos, ou mesmo, algumas das partes interessadas exigirem um recurso que seja incompatível para aquela determinada realidade. Nesses casos, o engenheiro de requisitos tem que se ater a ideia de não tomar medidas unilaterais quando tais conflitos ocorrem, por isso, é sensato sempre consultar as partes interessadas para se chegar em um consenso.

2.1.2.3 Especificação

Na especificação de requisitos a preocupação está na estruturação, qualidade e verificabilidade do documento de requisitos. Sendo que pode tomar a forma de dois documentos, ou duas partes do mesmo documento com diferentes leitores e propósitos, os quais são: especificação de requisitos do sistema e a especificação de requisitos de software (SAWYER; KOTONYA, 2001). A especificação de requisitos do sistema abreviado pela sigla SyRS (do inglês System Requirements Specification), também conhecido como documento de requisitos do usuário ou conceito de operações visa fornecer uma descrição do que o sistema deve fazer em termos de interações ou interfaces do sistema, sendo que, seus leitores podem incluir usuários/clientes (SAWYER; KOTONYA, 2001) (ISO/IEC/IEEE. . . , 2011).

Sawyer e Kotonya (2001) destaca que essa categoria de documento além de listar os requisitos do sistema, também deve conter informações básicas sobre os objetivos gerais do sistema, seu ambiente de destino, restrições, suposições e requisitos não funcionais. Não há um padrão ideal para a organização dos requisitos no documento SyRS, no entanto, conforme pode ser verificado na Fig. 6(a) os requisitos devem estar organizados de forma que auxilie na sua melhor compreensão, para isso é recomendado um consenso das partes interessadas em como os requisitos estarão melhor organizados ISO/IEC/IEEE. . . (2011).

Figura 6 – Padrão de documento SyRS e SRS segundo ISO/IEC/IEEE... (2011)



Fonte: Adaptado de ISO/IEC/IEEE... (2011) pág. 44-45

A especificação de requisitos de software - SRS (do inglês Software requirements specification), é um documento cujo objetivo esta em fornecer uma descrição mais abrangente do produto a ser desenvolvido onde é definido as condições e restrições sob as quais o software deve funcionar e as abordagens de verificação pretendidas para os requisitos (ISO/IEC/IEEE..., 2011). Lane e Krüger (2021) aponta que apesar de o SRS e SyRS serem equivocadamente indicado como algo equivalente, visto que na Fig. 6 notamos similaridades em sua estruturação é importante notar que o SRS em relação ao SyRS apresenta uma descrição mais detalhada do software que será desenvolvido.

2.1.2.4 Validação

Na validação de requisitos a preocupação está no processo de análise do documento de requisitos para garantir que ele descreva o sistema corretamente de modo a corresponder as expectativas do usuário (SAWYER; KOTONYA, 2001). Sawyer e Kotonya (2001) destaca haver quatro subtópicos importantes nessa etapa de validação, sendo: revisão dos requisitos, prototipagem, validação do modelo e testes de aceitação. Na revisão dos requisitos o objetivo é identificar erros, suposições equivocadas, falta de clareza e desvio do padrão utilizado, sendo a inspeção ou

revisões formais os meios mais comuns de validação dos documentos. Para Sawyer e Kotonya (2001), a composição do grupo responsável pela revisão deve ser composta por ao menos um representante do cliente, pois, isso pode fornecer uma análise mais abrangente.

A prototipagem é empregada para validar a interpretação do engenheiro de requisitos a respeito dos requisitos do sistema propostos. A vantagem dos protótipos é que eles podem facilitar a interpretação das suposições do engenheiro de requisitos e dar feedback útil sobre por que estão errados (SAWYER; KOTONYA, 2001). Nos testes de aceitação temos a validação os requisitos do sistema, para isso, é importante planejar como se dará a verificação de cada requisito. Identificar e projetar testes de aceitação para requisitos não funcionais segundo Sawyer e Kotonya (2001) é uma tarefa mais complexa, pois devido a sua natureza, é requerido que sejam expressos quantitativamente.

2.1.2.5 Gerenciamento de Requisitos

Gerenciar as mudanças e manutenção dos requisitos para refletir com precisão o software a ser ou que foi construído é uma tarefa que abrange o gerenciamento de requisitos e isso é realizado durante todo ciclo de vida de um software (SAWYER; KOTONYA, 2001). Sawyer e Kotonya (2001) destaca que três subtópicos compõe essa etapa sendo: seus atributos, rastreamento e o seu gerenciamento de mudanças. Os atributos dos requisitos não devem conter apenas uma especificação do que é necessário, mas também informações auxiliares que ajudem no seu gerenciamento e interpretação. Essas informações auxiliares devem incluir as várias dimensões de classificação de um requisito, métodos usados em sua verificação, testes de aceitação, uma justificativa resumida para o requisito, sua fonte e um histórico de alterações.

No rastreamento de requisitos a preocupação está centrada em rastrear sua origem e prever seus efeitos. Destaca-se que tanto o requisito quanto a parte interessada que o motivou seja rastreável, pois, isso é um facilitador no momento da análise do requisito que estiver passando por alterações. Por fim, o último tópico importante no gerenciamento de requisitos trata do gerenciamento das suas mudanças. Nesse tópico é descrito o papel do gerenciamento de mudanças, os procedimentos que precisam ser implementados e as análises que devem ser aplicadas nas mudanças propostas (SAWYER; KOTONYA, 2001).

2.1.3 Projeto

Desde o advento da web 2.0, o design da interface do usuário foi um ponto critico na experiência de uso de determinada aplicação, independente de qual seja a experiência pretendida. Para isso, é essencial definir a estratégia de projeto para o

produto através das ideias levantadas, bem como, planejar as suas funcionalidades, evolução e realizar testes com os usuários. Isso ajuda o UX designer e os demais membros da equipe a capturar insights, de modo que, possibilite desenhar um produto relevante para o cliente final (TEIXEIRA, 2014) (ANDERSON et al., 2010).

Apesar da palavra “UX design” conter a palavra “design” isso não significa que esse profissional se limite a criação da aparência do produto final, mas em como as pessoas irão interagir com o produto, quais tarefas conseguirão realizar, a ordem em que as telas serão apresentadas, etc (TEIXEIRA, 2014). Teixeira (2014) aponta que esses profissionais começaram mais como ‘arquitetos da informação‘, onde eram responsáveis por organizar o conteúdo de um site e representar como uma interface deveria funcionar, utilizando os famosos “wireframes”. No entanto, a medida que o ecossistema digital das empresas foi evoluindo e se tornando cada vez mais complexo, esses profissionais acabaram assumindo um papel mais estratégico no processo criativo. Atualmente, esses profissionais transitam no planejamento e no desenho do produto, ajudando a definir quais ideias tomarão forma.

Para Teixeira (2014) esse profissional de UX deve ser visto como um “camaleão” se adaptando aos diferentes contextos, conhecendo o vocabulário dos diferentes profissionais envolvidos, como o do programador front-end e também dos designers, sendo fundamental entender um pouco de cada uma dessas áreas. Teixeira (2014) enfatiza que todos os profissionais que de alguma forma contribuam na construção da interface do usuário, podem ser considerado um UX designer, pois, essa função não precisa ser desempenhada por uma categoria de profissional com formação específica. Essa área de estudo voltada a experiência do usuário, é ampla, sendo que, alguns UX designers podem estar concentrados em algumas das sub-áreas de UX, sendo elas (TEIXEIRA, 2014):

Arquitetura de informação: na arquitetura de informação o propósito está em como organizar as informações da aplicação, de modo que, seja acessada mais facilmente pelos usuários. Nessa sub-área também é investigado o perfil do usuário e qual informação ou recurso que ele busca, sendo também investigado como a informação está ordenada, agrupada e organizada na estrutura do software ou site.

Usabilidade: a usabilidade garante que as interfaces sejam fáceis de usar, sendo investigado métodos que visem garantir um acesso a determinado recurso como o mínimo de passos, transtorno no entendimento de utilização ou demora no acesso.

Design de interação: no design de interação se procura entender e definir o comportamento das interfaces quando o usuário interage com elas. Aqui questionamentos como: o que ocorre após um clique em determinado botão? como a interface responde ao acessar determinados recursos? de que modo

a aplicação pode ser utilizada, para que, o usuário tenha uma experiência mais relevante? são alguns dos questionamentos abordados nessa sub-área.

Taxonomia: na taxonomia o estudo está concentrado em organizar e rotular a informação de forma que faça sentido para o usuário. Isso vai desde investigar o que ocorre quando o usuário busca por algum item com variações dessa mesma busca, como também se o perfil demográfico do usuário que acessa a aplicação está habituado a sua linguagem.

Estratégia de design: a estratégia de design gera o entendimento e definição dos porquês do produto. Questionamentos como: para quem está sendo desenvolvido a aplicação; qual o seu retorno esperado; quais os objetivos de negócio e como eles serão alcançados; como medir o sucesso do produto visto que ele foi entregue e quais ferramentas podem ser utilizadas para mensurar esse sucesso, são alguns dos pontos levantados.

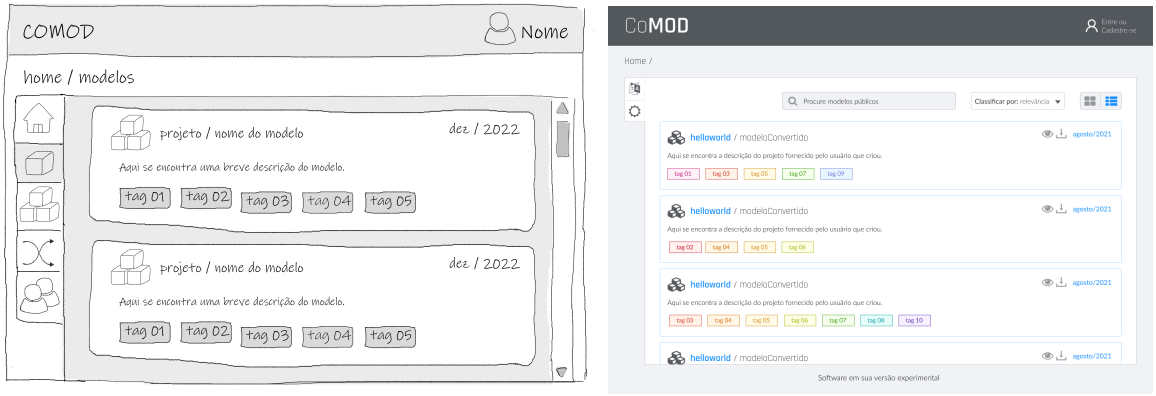
Pesquisa com usuários: na pesquisa com usuários o objetivo é gerar um entendimento do público alvo. Nessa sub-área é investigado as necessidades e anseios que levem o usuário a utilizar determinado produto, sendo aqui considerado as tarefas que serão desempenhadas, seu comportamento e influência no produto final a ser desenvolvido.

Conforme destacado por Teixeira (2014), nem todo material gerado pelo profissional de UX é observado pelo consumidor final, visto que, esses entregáveis e processos utilizados tem por objetivo facilitar a comunicação entre os membros da equipe, documentar as decisões tomadas e também colher opiniões, de modo a, garantir que todos estejam alinhados a respeito do que está sendo criado. Esses entregáveis produzidos pelos profissionais de UX podem variar conforme o projeto, expectativas do cliente, objetivos de design e membros da equipe envolvidos no momento da elaboração dos detalhes da aplicação. Os entregáveis mais comuns são aqueles que documentam a interface do usuário por meio de wireframes e protótipos (TEIXEIRA, 2014).

2.1.3.1 Wireframes

O wireframe é um desenho básico da estrutura de determinada interface de usuário que demonstra de forma simples como o produto deve funcionar, sendo normalmente desenvolvidos em tons de cinza, sem imagens e identidade visual da marca do cliente. Essa documentação é útil, pois ajuda não somente os designers que são os responsáveis por desenvolver os elementos visuais essenciais para o layout, quanto também podem servir como um estimador para os desenvolvedores do quanto de esforço pode ser desprendido no desenvolvimento da aplicação (TEIXEIRA, 2014). Na Fig. 7(a) é ilustrado um exemplo de wireframe para o caso do usuário não autenticado da plataforma deste trabalho.

Figura 7 – Wireframe e protótipo da plataforma para o caso do usuário não autenticado



Fonte: De autoria própria

Teixeira (2014) indica que na criação dos wireframes o profissional de UX deve considerar alguns fatores, como: os objetivos de negócios do cliente; requisitos técnicos do sistema; o conceito criativo do produto; as possibilidades e limitações das tecnologias em que a aplicação será desenvolvida e também as possibilidades e limitações de hardware em que a interface é acessada, como computadores desktop, smartphones, tablets, etc.

2.1.3.2 Protótipos

Um protótipo é uma representação da interface do usuário onde é demonstrado como essa interface deve responder após à interação do usuário. Diferente dos wireframes os protótipos são mais dinâmicos e apresentam os detalhes e elementos visuais que a aplicação final deve conter. Os protótipos possuem uma grande vantagem em relação aos wireframes estáticos, pois, por serem dinâmicos são mais intuitivos, visto que, dependem apenas da iteração do usuário. E assim, ao contrário dos wireframes que a cada interação é representado em uma imagem, gerando assim, uma quantidade excessiva de arquivos que pode acarretar confusão no primeiro momento de análise (TEIXEIRA, 2014). Conforme ilustrado na Fig. 7(b) é proposto um protótipo baseado no wireframe da figura anterior.

2.1.4 Construção de Software

O processo de construção de software é o último dos estágios da metodologia BDUF. Conforme destacado por Bollinger, Gabrini e Martin (2001), isso não implica que o processo de construção de software esteja isolado das etapas anteriores, pelo contrário, as etapas de brainstorming, requisitos e projeto passam a trabalhar em conjunto, visto que, algum requisito, design ou percepção se torne inviável quando implementado na prática.

Segundo Bollinger, Gabrini e Martin (2001), o processo de construção de software deve ser visto como uma simbiose entre o computador e o programador. Sendo o computador responsável por oferecer confiabilidade e velocidade de desempenho, e o programador em fornecer sua criatividade e insights sobre como resolver os problemas na medida que surgem. Além é claro, da sua capacidade em expressar essas soluções com precisão para serem entendidas pelo computador.

Bollinger, Gabrini e Martin (2001) destaca que o processo de construção de software está intimamente ligada a etapa de projeto, visto que, essa etapa analisa os requisitos de software dividindo um problema maior em partes menores mais fáceis de lidar, produzindo uma estrutura de design que sirva de base para a sua construção. Em alguns casos, essa fronteira entre projeto e construção de software pode não ser muito definida, pois esse processo de construção é influenciado pela escala ou tamanho do produto de software que esta sendo construído. Assim, projetos grandes podem exigir uma relação muito mais interativa entre projeto e construção.

De acordo com Bollinger, Gabrini e Martin (2001), as ferramentas de construção de software exercem um papel fundamental nesta última etapa, pois essas ferramentas permitem que o programador expresse seus pensamentos com facilidade e impõem um nível adequado de rigor, melhorando a qualidade do software e permitindo que as pessoas evitem trabalhos repetitivos para os quais um computador é mais adequado. Exemplos comuns de ferramentas de construção de software incluem compiladores, sistemas de controle de versão, depuradores, geradores de código, editores de código, ferramentas para documentação, etc.

Para haver um pleno entendimento entre os desenvolvedores do que esta sendo implementado é imprescindível o estabelecimento de padrões para seu processo de construção de software. Padrões podem ser selecionados pelo cliente ou pela própria organização, sendo que esses padrões podem mudar conforme a característica de cada projeto influenciado diretamente pelas linguagens de programação, banco de dados, métodos de comunicação, plataformas e ferramentas (BOLLINGER; GABRINI; MARTIN, 2001).

O processo de construção de software pode ser feito de forma manual ou automatizada. A construção manual de um software implica em resolver um problema complexo em uma linguagem de programação em que o computador pode executar. Nesse método os programadores devem conseguir subdividir um problema complexo em partes menores, além de prever como a construção do software pode passar por atualizações futuras, de modo que a sua estrutura tenha o mínimo de alterações possíveis (BOLLINGER; GABRINI; MARTIN, 2001).

Uma construção automatizada de software segundo Bollinger, Gabrini e Martin (2001) refere-se a uma ferramenta ou ambiente automatizado que é o principal responsável pela coordenação geral do processo de construção de software. Essa

remoção do usuário no controle do processo tem suas vantagens, pois pode ter um grande impacto na complexidade do processo de construção do software. Isso permite que as contribuições humanas sejam divididas em tarefas menores e menos complexas, que exigem diferentes habilidades de resolução de problemas a serem solucionados.

Visto que a tarefa fundamental de um software está em comunicar a intenção entre dois agentes, as pessoas e o computador, essa interface entre os dois é normalmente expressa em linguagens de programação. As linguagens de programação são criadas em resposta a uma necessidade em um campo de aplicação específico e segundo Bollinger, Gabrini e Martin (2001) é importante que a construção de software não seja muito amarrada a qualquer linguagem de programação ou metodologia de programação, visto que, em algum momento essa linguagem ou metodologia pode se tornar obsoleta.

2.1.4.1 Princípios de Organização

Para Bollinger, Gabrini e Martin (2001), um dos pontos importantes no processo de construção de um software ao subdividi-lo em processos menores com menos complexidade, está em reconhecer quatro princípios fundamentais de organização, sendo: redução da complexidade, antecipação de mudança, estruturação para validação e uso de padrões externos.

Como o ser humano possui uma capacidade limitada em trabalhar com sistemas complexos que possuem muitas partes e interações, surge a necessidade em simplificar essa interface entre o homem e computador no processo de construção de software, nasce assim um dos princípios de organização mais importantes, o princípio da redução de complexidade. De acordo com Bollinger, Gabrini e Martin (2001), existem três técnicas principais para redução de complexidade, sendo: remoção, automação e localização da complexidade.

Remoção da complexidade: um dos meios de se reduzir a complexidade de um software está em remover recursos ou capacidades que não são absolutamente necessários, embora nem sempre seja o método mais correto de se lidar, sempre ter em mente o princípio da parcimônia é fundamental, ou seja, não adicionar recursos e capacidades que não serão necessários na construção do software.

Automação da complexidade: outra forma de se reduzir a complexidade de um software é migrar funções que antes consumiam tempo e são propensas a erros pelo fator humano, para serem desempenhadas pelo computador. Um exemplo disso esta nas linguagens de programação visuais, onde, ao invés do usuário programar utilizando elementos textuais, se utiliza elementos gráficos.

Localização da complexidade: se essa complexidade não pode ser removida e muito menos automatizada, a única opção restante está em localizar essa complexidade e subdividi-las em pequenos módulos, pequenos o suficiente para ser possível um entendimento da sua totalidade. No entanto, conforme destacado por Bollinger, Gabrini e Martin (2001) subdividir uma sequência muita longa de código em sequências menores de forma arbitraria pode torná-lo mais complexo, visto que, fica mais difícil identificar essas relações entre módulos.

No segundo princípio, que trata da antecipação de mudança, o objetivo esta em prever como as pessoas podem utilizar o software, pois dado a sua natureza mutável, qualquer aplicação que seja útil passa por modificações futuras e antecipar essas mudanças impulsiona o seu processo de construção. Existem três técnicas principais que visam antecipar as mudanças no software, sendo: generalização, experimentação e localização (BOLLINGER; GABRINI; MARTIN, 2001).

Generalização: é comum que os estágios iniciais do desenvolvimento do software estejam concentrados em problemas específicos, pois nos estágios iniciais é mais óbvio a sua identificação. Na generalização a proposta está em tornar esses problemas específicos em algo mais amplo que se encaixe em uma estrutura maior para a solução de outros possíveis problemas.

Experimentação: na experimentação é utilizado construções do software inicial submetendo-os à diferentes contextos no proposito de coletar dados sobre como generalizar a sua construção, sendo que, na experimentação é reconhecido essa dificuldade em antecipar todas as possíveis mudanças que podem ocorrer.

Localização: na localização o objetivo é manter as mudanças antecipadas o mais localizada possível sem afetar a estrutura global do software.

Independente do cuidado que se possa ter no processo de desenvolvimento de um software, devido a sua natureza não ser trivial, erros e omissões podem ocorrer por parte do programador. A estruturação para validação propõe construir software de tal forma que tais erros e omissões possam ser descobertos mais facilmente durante o teste de unidade e atividades de teste posteriores. A estruturação para validação geralmente esta atrelada com a antecipação de mudança, visto que, qualquer erro encontrado na validação representa uma importante mudança no software (BOLLINGER; GABRINI; MARTIN, 2001).

Obviamente, é difícil desenvolver um software que possa ser testado em toda a sua totalidade, visto que, alguns componentes podem cobrir uma grande gama de saídas, que exigiria vários testes para todas as saídas possíveis, tornando assim, o processo de teste longo e exaustivo. Com isso, mesmo que a estruturação para validação seja importante no processo de construção do software, ela deve ser

executada dentro de um prazo razoável estipulado pela equipe, pois validar o software por completo pode se tornar inviável (BOLLINGER; GABRINI; MARTIN, 2001).

É comum um software compartilhar dados e até mesmo módulos de trabalho com outro software em qualquer lugar, diante disso, compartilhar as mesmas linguagens e métodos são fundamentais para essa comunicação. Essa importância surgiu desde o advento da internet, onde selecionar e usar padrões externos apropriados e estáveis para construir um software se tornou algo fundamental para poder existir uma compreensão do que foi feito por outras pessoas (BOLLINGER; GABRINI; MARTIN, 2001). Conforme destacado por Bollinger, Gabrini e Martin (2001) a seleção de um padrão externo pode considerar questões como a sua estabilidade e viabilidade econômica de longo prazo de uma determinada empresa de software ou organização que promove esse padrão.

2.2 TRABALHOS RELACIONADOS

Este trabalho utiliza a metodologia BDUF, que apresenta a característica de desprender um esforço maior na fase de prototipação e estudo voltado a experiência do usuário. Na fase de projeto, é utilizado o GitHub (2022) como referência na elaboração dos wireframes. GitHub (2022) é uma plataforma amplamente conhecida e difundida pela comunidade de desenvolvedores e apresenta uma interface intuitiva e convidativa ao usuário. Na plataforma, é possível realizar a hospedagem de código-fonte e de arquivos em repositórios, com controle de versão por meio do Git.

Além disso, usuários cadastrados na plataforma podem contribuir em repositórios públicos ou privados, de qualquer lugar do mundo. A plataforma MOMS (Model-Oriented Management System), proposta neste trabalho, apresenta similaridades com o GitHub. Uma delas é o compartilhamento, sendo que no MOMS usuários cadastrados na plataforma podem compartilhar seus modelos e sequências de conversão, sejam eles por meio de projetos públicos ou privados.

Outro recurso presente no GitHub e proposto no MOMS, está em organizar itens em diferentes níveis. No GitHub, por exemplo, diferentes repositórios podem ser agrupados em uma organização e compartilhados com usuários convidados. No MOMS, a proposta é que diferentes projetos sejam agrupados em um grupo, sendo que, para fazer parte de um grupo é necessário um convite.

3 MATERIAIS E MÉTODOS

Para ser possível implementar uma aplicação funcional com o mínimo de modificações possíveis na etapa de desenvolvimento do software é utilizado algumas ferramentas de modo que mitigue correções que possam surgir. Com isso, na etapa de validação de requisitos é utilizado a modelagem de eventos para fins de validação de requisitos voltados a interface do usuário, essa ferramenta se mostra importante, pois as partes interessadas conseguem uma visão mais ampla de como ocorre a interação do usuário na aplicação para o requisito em questão.

Um recurso importante utilizado na etapa de desenvolvimento da interface do usuário são os chamados protótipos. Protótipos são importantes, pois além de gerar uma documentação muito útil utilizada na implementação da aplicação também é uma ferramenta utilizada no encantamento do cliente, pois com ela é possível apresentar um estudo de design e iterações que ocorrem ao longo de cliques do usuário na aplicação. Por fim, com toda a documentação de requisitos de software e de interface do usuário disponível é então implementado a aplicação utilizando bibliotecas e tecnologias aderentes a aplicação proposta.

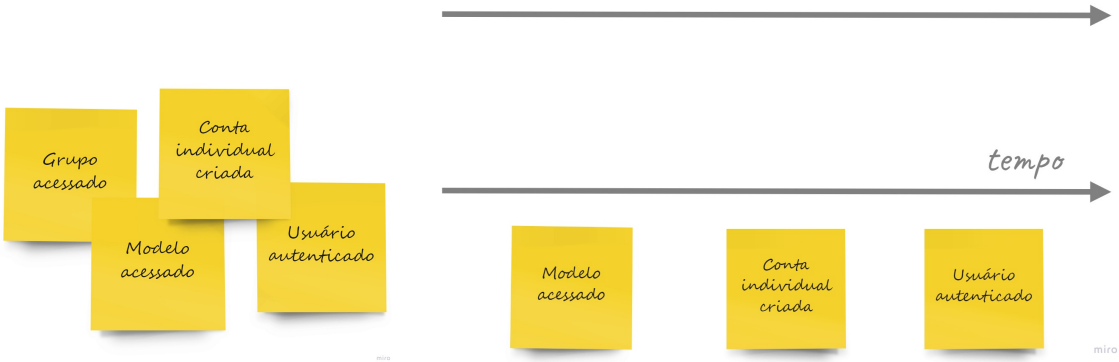
3.1 VALIDAÇÃO DE REQUISITOS

Na etapa de análise dos requisitos de software é realizado uma modelagem conceitual para validar ou tornar mais claro os requisitos focados na experiência do usuário, onde a ferramente utilizada é a modelagem de eventos. O objetivo da modelagem de eventos está em identificar como ocorre a interação do usuário na medida que ele acessa diferentes recursos da aplicação, sendo que, na modelagem de eventos os componentes do software são representados na sua forma mais básica, sem uma preocupação com a etapa de design, visto que, o enfoque está em desenvolver uma sequência fluida com um mínimo de cliques possíveis e trocas de páginas (LAU, 2020).

Segundo Dymitruk (2019) a modelagem de eventos é constituída por alguns passos principais, sendo: brainstorming, ordenação logica dos eventos, ordenação dos wireframes, identificação das entradas e identificação das saídas, sendo que, para representar cada etapa de forma visual e intuitiva é utilizado o Miro(PERMINOVA, 2022). Miro é uma plataforma de lousa interativa digital, onde nela é possível “colar” notas adesivas (post-its) em uma área de trabalho e colaborar com várias pessoas no desenvolvimento de projetos, sendo que, essa ferramenta nos possibilita executar diversas tarefas na criação de um projeto como, por exemplo, o desenvolvimento de esboços e wireframes (PERMINOVA, 2022).

Na etapa do brainstorming para a modelagem de eventos é importante que alguém responsável explique os objetivos gerais do projeto e passe informações relevantes aos participantes do brainstorming, de modo que, os participantes imaginem como o sistema descrito se pareceria ou como se comportaria. Dymitruk (2019) enfatiza que o objetivo aqui está em identificar os eventos de mudança de estado. Um evento de mudança de estado é algo que ocorre após a interação do usuário com determinada recurso da aplicação, ou seja, se um usuário desejar se cadastrar na aplicação deste trabalho e considerando que todos os campos foram preenchidos de forma correta, um agente externo (nesse caso uma API) gera um evento de conta criada.

Figura 8 – Figura exemplo no levantamento e ordenação logica dos eventos

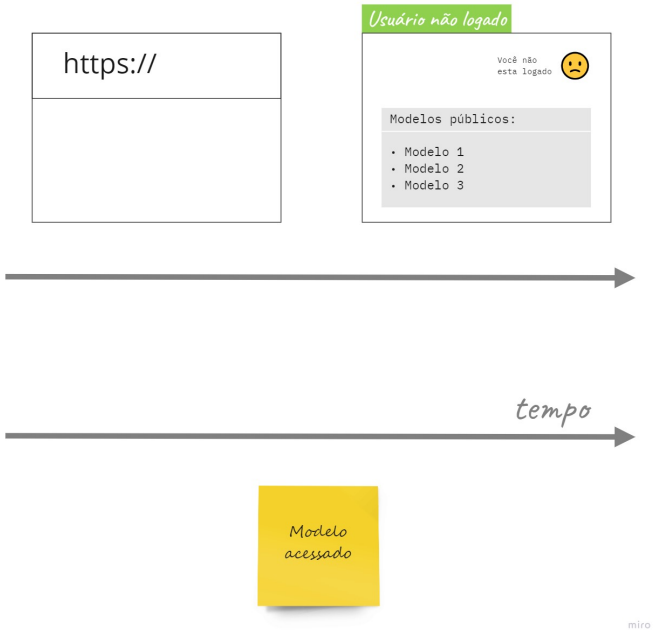


Fonte: De autoria própria

Na Fig. 8(a) é exemplificado como esses eventos podem ser representados no Miro por meio de post-its, no passo seguinte, conforme observado na Fig. 8(b) o objetivo é realizar a ordenação logica dos eventos segundo a sua ocorrência. Após a ordenação dos eventos é então ordenado os wireframes, eles são usados, pois, ilustram à perspectiva do usuário, sendo que, esses wireframes são rascunhos na sua forma mais básica, deixando detalhes de um design mais elaborado em segundo plano, visto que, o objetivo aqui é apenas validar sua interação ao utilizar o sistema.

Na Fig. 9 é mostrado um exemplo de como isso pode ser esboçado no Miro, e conforme observado, dois wireframes compõe o evento de acessar os modelos públicos da plataforma, os quais são: o momento em que o usuário digita o endereço eletrônico da aplicação e quando a aplicação é carregada com a listagem dos modelos públicos compartilhados pelos usuários. Essa figura utiliza apenas um dos eventos listados da Fig. 8(b), isso é feito apenas para fim de simplificação deste exemplo, visto que, quanto mais eventos mais wireframes deverão ser esboçados tornando a figura maior e mais complexa.

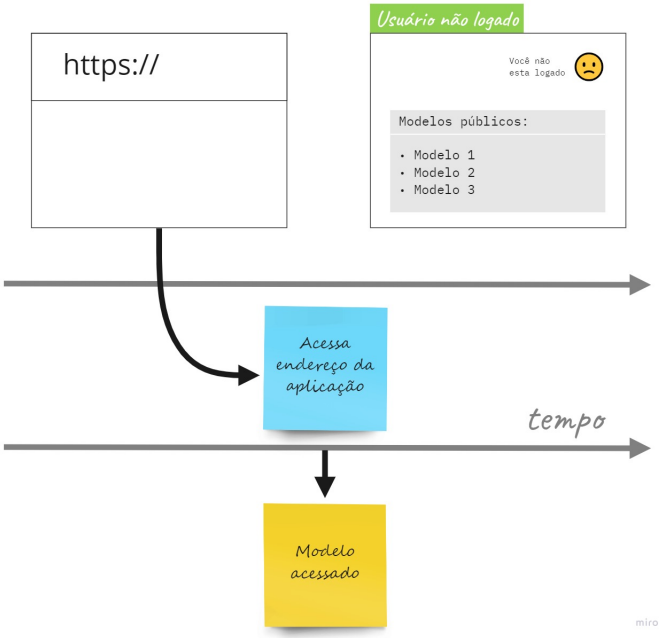
Figura 9 – Figura exemplo na ordenação dos wireframes



Fonte: De autoria própria

A etapa seguinte constitui em identificar as entradas, essas entradas são um resultado da iteração do usuário no momento em que é solicitado algum recurso da aplicação. No exemplo da Fig. 10 é possível verificar que a entrada que aciona o evento de listar os modelos públicos está em apenas acessar o endereço eletrônico da aplicação.

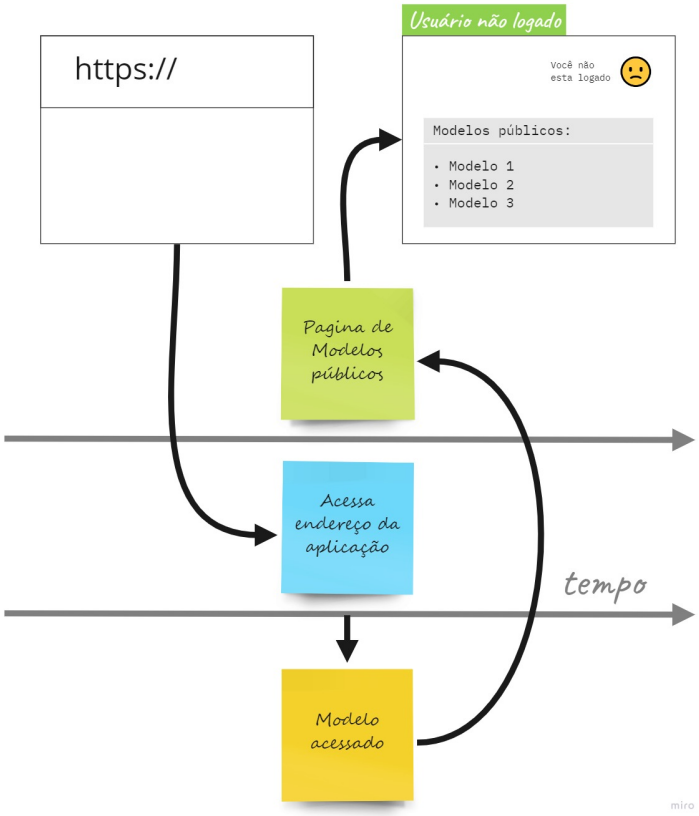
Figura 10 – Figura exemplo na identificação das entradas



Fonte: De autoria própria

Por fim, é identificado a saída que o evento gera, esse evento de saída é um feedback visual para o usuário sendo representado na forma de wireframes. Na Fig. 11 do exemplo abordado a nossa saída corresponde a carregar a tela inicial da aplicação com todos os modelos públicos compartilhados pelos usuários da plataforma.

Figura 11 – Figura exemplo na identificação das saídas



Fonte: De autoria própria

3.2 INTERFACE DO USUÁRIO

Para o desenvolvimento da interface do usuário é utilizado um software para a sua representação e de seus eventos de forma profissional, para isso é utilizado o Figma(GARRETT, 2021). Figma é um editor online colaborativo de gráficos vetoriais com ênfase na prototipagem de interfaces gráficas e estruturas de design voltado para a experiência de usuário, sendo muito utilizado por equipes diversas, pois ela permite que esses profissionais, em locais diferentes, visualizem e trabalhem em modificações do projeto em tempo real. A ferramenta é muito útil, pois além de mostrar o design final da aplicação também permite que o usuário interaja com esse protótipo, dando ao cliente uma visão mais clara de como será o software quando implementado (GARRETT, 2021).

Um ponto importante, é que os protótipos citados aqui, diferem dos wireframes citados na etapa da validação de requisitos, visto que, na etapa de elaboração da interface do usuário o objetivo é representar o design da aplicação na sua forma mais profissional e detalhada possível.

3.3 IMPLEMENTAÇÃO DA APLICAÇÃO

Para a implementação da aplicação web é utilizado algumas ferramentas consideradas aderentes para o seu desenvolvimento, sendo uma delas a biblioteca React(GACKENHEIMER; PAUL, 2015). O motivo da escolha está em seu grande uso no momento, sendo que, a biblioteca vem sendo mantida pelo Facebook, Instagram e uma ampla comunidade de empresas e desenvolvedores individuais, com isso, há uma garantia de atualizações e suporte por parte de toda comunidade. O React é uma biblioteca JavaScript de código aberto baseada na tecnologia SPA com foco em criar interfaces de usuário em páginas para web, onde se utiliza uma sintaxe parecida com XML chamada JSX conforme ilustrado na Fig. 12 (GACKENHEIMER; PAUL, 2015) (FACEBOOK, 2022).

Figura 12 – Figura exemplo da sintaxe JSX do React

Fonte: Facebook (2022)

SPA é uma abreviação do inglês Single Page Applications ou no português Aplicações de Página Única. Apesar do nome permitir uma dedução, isso não significa que a aplicação conterá apenas uma página, e sim a forma com que a página irá ser carregada destoará das outras aplicações. Aplicações que não são baseadas em SPA são páginas renderizadas no lado do servidor, isso acarreta que a cada nova página acessada da aplicação uma nova requisição é feita para o servidor, afim que, se carregue o HTML, CSS e o JavaScript da nova página requisitada. Já em aplicações baseadas em SPA o seu carregamento acontece por inteiro na primeira requisição, onde

todo o HTML, CSS e JavaScript necessários são carregados de uma vez (GUEDES, 2020).

Essa abordagem tem as suas vantagens, uma delas é a otimização do desempenho da aplicação ao deslocar todo o esforço de renderização para o cliente e permitir um tráfego de dados mais leve entre cliente e servidor, no entanto, aplicações SPA possuem um ponto fraco em relação a SEO. SEO é uma abreviação de Search Engine Optimization, traduzido para o português como Otimização para Mecanismos de Busca. Se trata de um conjunto de técnicas cujo objetivo é posicionar as páginas de destino entre os melhores resultados dos mecanismos de busca (GUEDES, 2020) (BENETTI, 2022).

Como aplicações SPA não são renderizadas no lado do servidor, sendo renderizado apenas no lado do cliente, isso faz com que os mecanismos de busca encontrem dificuldades para indexar o conteúdo das páginas, visto que mecanismos de busca não podem indexar conteúdo gerado do lado do cliente (GUEDES, 2020). Como o objetivo da aplicação deste trabalho é ser utilizado por uma variedade de usuários, possuir um bom posicionamento nos motores de busca é fundamental, e para que esse problema seja contornado é utilizado uma ferramenta adicional, o framework Next.js.

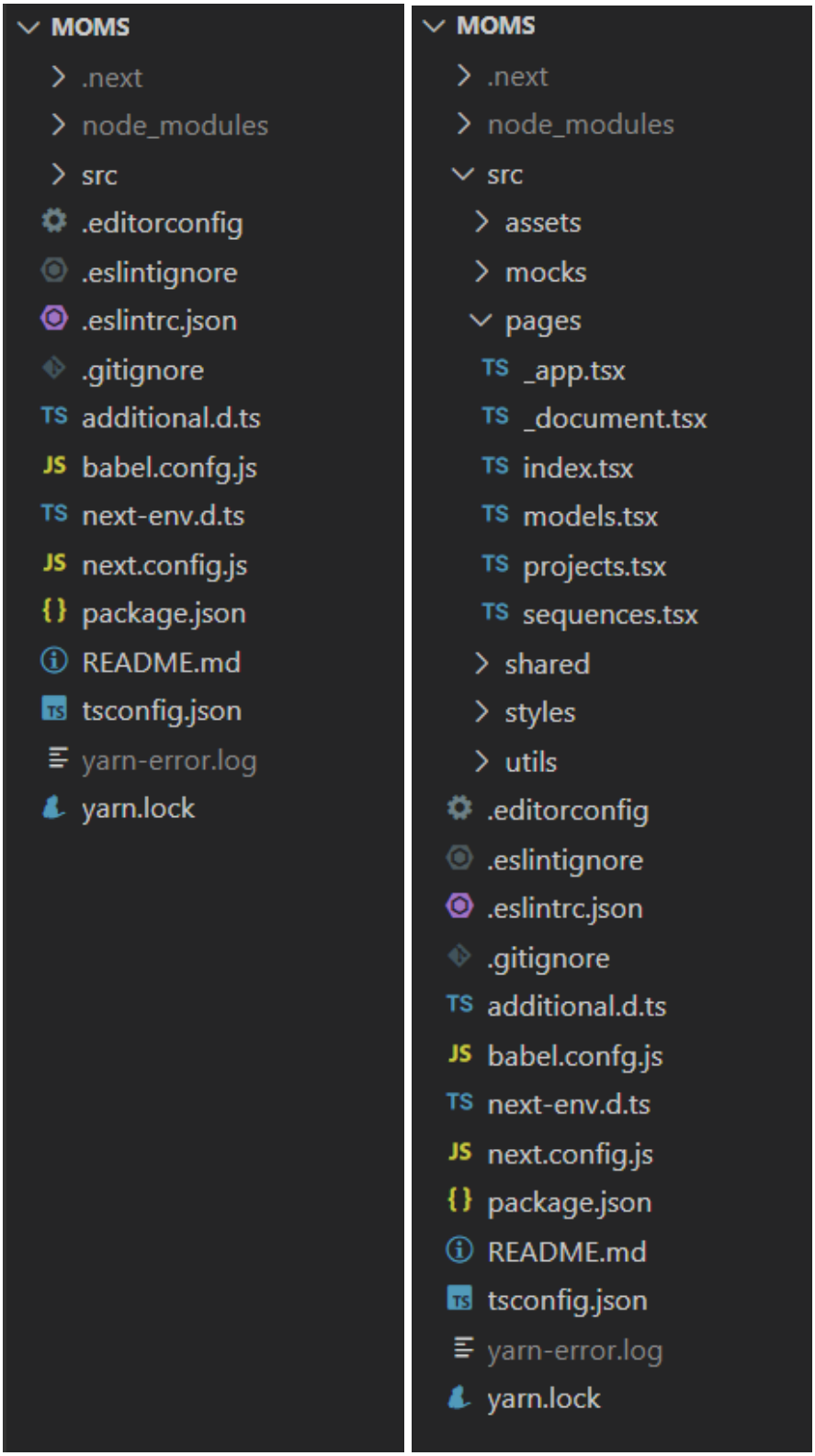
Next.js é um framework para React, ele é considerado um framework, pois adiciona várias outras funcionalidades ao React, sendo elas, a renderização estática pelo lado do servidor, suporte para Typescript e tratamento de rotas. Para gerar toda a página para o navegador o Next utiliza um servidor. Sessa forma, o Next consegue entregar a página pronta para o navegador, ou seja, todo o HTML, CSS e Javascript e isso faz com que a página disponha de conteúdo para ser indexado pelos navegadores, contornando assim o problema com SEO quando se utiliza apenas o React na implementação da aplicação (ROCHA, 2021).

Como o Next oferece suporte para Typescript e o React é uma biblioteca para Javascript é então utilizado essas duas linguagens de programação para o desenvolvimento da aplicação, e visto que, a aplicação precisa realizar requisições para uma API é utilizado o Axios. Axios é um cliente HTTP baseado em promessas para fazer requisições que pode ser utilizado tanto no navegador quanto no Node.js, sendo um projeto open source muito utilizado e mantido por uma ampla comunidade no Github (MARINHO, 2020).

Com isso, é estruturado a aplicação conforme pode ser observado na Fig. 13(a) onde iniciar um projeto no Next.js é gerado uma estrutura de projeto composto por pastas e arquivos que serve como base para implementação da aplicação. Na Fig. 13(b) é possível notar que a pasta “src” possui pastas adicionais além da “pages”, que ela por padrão é adicionada pelo Next, tais pastas são inseridas para fim de estruturação do projeto no intuito de obter uma melhor organização na medida que novos estilos e componentes são adicionados. Não há um padrão definitivo para

estruturar um projeto, onde, nome de arquivos e pastas, bem como, sua localização é algo que vai depender da experiência de cada programador.

Figura 13 – Estrutura da aplicação MOMS



Fonte: De autoria própria

Neste trabalho são adotados alguns padrões para que outros desenvolvedores possam se familiarizar com a estrutura e não perder muito tempo em entender onde cada recurso está localizado. Por padrão qualquer pasta ou arquivo novo que precise ser inserido na aplicação deve estar contido na pasta “src”, sendo que, o nome de todas as pastas que não representem algum componente genérico compartilhado é escrito em lowercase e sem nomes compostos.

Caso a pasta possua algum arquivo que representa um componente genérico, o nome da pasta e do arquivo deve ser escrito no padrão ‘PascalCase’ (consultar Fig. 13(b) na pasta “shared”). Na listagem abaixo é possível conferir a função de cada pasta que se encontra presente na pasta “src”:

assets: contém arquivos de imagem, mídia, etc.

mocks: contém arquivos de configuração de inicialização da aplicação e configuração das funcionalidades que os modelos, sequencias de conversão, projetos e grupos disponibilizam para o usuário. Como, por exemplo: um projeto pode ser: baixado, criado, editado, copiado e excluído.

pages: pasta padrão gerada pelo Next, que contem arquivos que serão transformados em rotas na plataforma. Cada arquivo presente nessa pasta reapresenta uma rota, como por exemplo: o arquivo “models.tsx” redireciona para a URL responsável por listar os modelos, sendo que podem ser públicos ou privados dependendo da ação do usuário.

shared: pasta contendo elementos genéricos utilizados em diferentes paginas da plataforma.

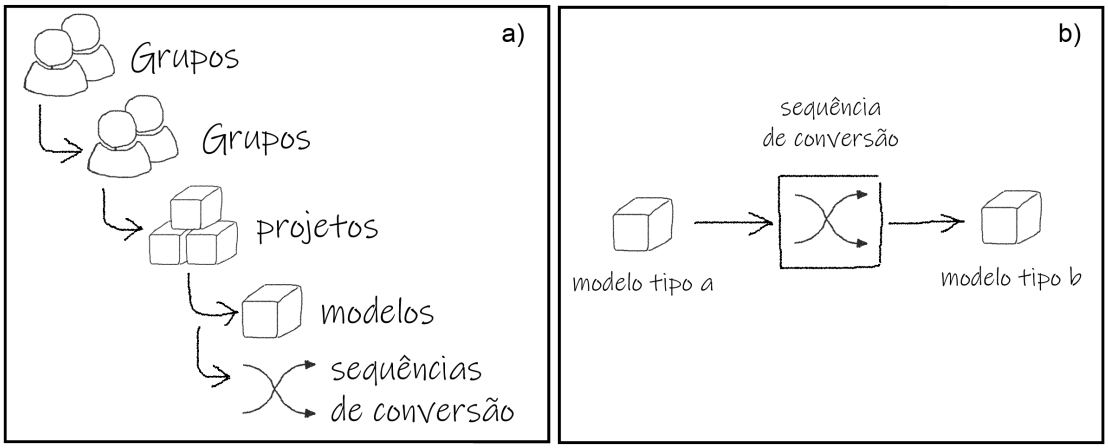
styles: contém os estilos e tema da página.

utils: contém elementos não genéricos utilizados em recursos específicos da aplicação.

4 RESULTADOS E DISCUSSÕES

As características gerais do software estão baseadas em poder cadastrar usuários suportando diferentes níveis de acesso, onde a aplicação deve conseguir suportar o carregamento, download, exclusão e conversão de diferentes categorias de modelos. Conforme indicado na Fig. 14(a) modelos e sequências de conversão são agrupadas dentro de um projeto, sendo que um projeto, modelo ou sequência de conversão pode ser criado, editado, excluído ou copiado. Projetos são públicos ou privados e podem ser agrupados em outros grupos, onde, grupos podem ser compartilhados com outros usuários cadastrados na plataforma. Para ser possível a conversão de modelos o usuário deve utilizar sequências de conversão, de modo que, seja indicado a categoria de modelo de entrada e a categoria de modelo de saída suportado na plataforma conforme ilustrado na Fig. 14(b).

Figura 14 – Estrutura base da aplicação

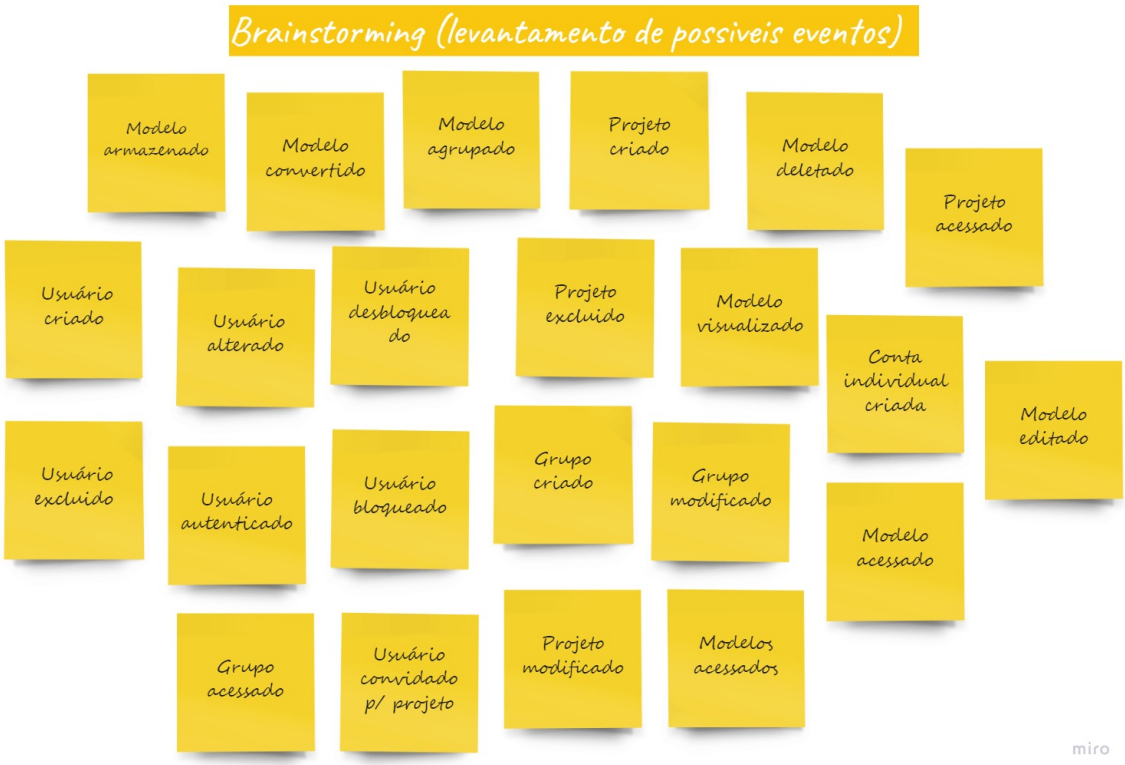


Fonte: De autoria própria

4.1 VALIDAÇÃO DE REQUISITOS

Inicialmente é imaginado e anotado os eventos possíveis de mudança de estado que podem ocorrer quando o usuário interagir com a aplicação. Nessa etapa inicial, não há uma preocupação em como organizar as informações, mas em apenas identificá-las. Como um dos requisitos de software é que usuários, grupos, projetos, modelos e sequências de conversão suportem operações básicas como: criar, editar e excluir é então identificado esses eventos e ilustrado na Fig.15. Também nessa identificação de eventos é incluído alguns relacionados ao usuário, como: sua solicitação para realizar login no sistema, seu bloqueio, desbloqueio e convite para fazer parte de algum projeto ou grupo através de um administrador.

Figura 15 – Listagem de possíveis eventos

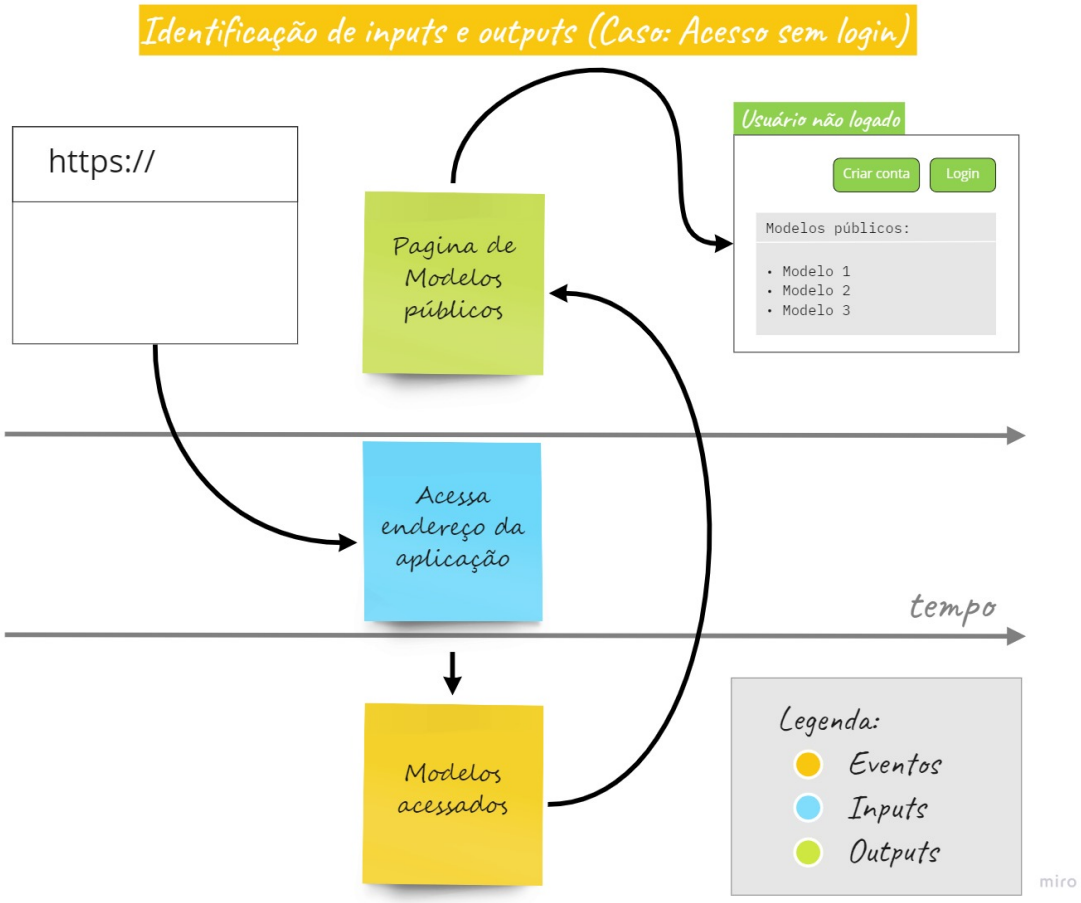


Fonte: De autoria própria

No passo seguinte, é realizado a ordenação dos eventos visando definir a sua ordem de ocorrência. Essa ordenação logica é útil para a identificação dos inputs e outputs relacionados aos eventos. Na modelagem de eventos, inputs tratam das ações do usuário, ou seja, o momento em que o usuário interage com a aplicação no objetivo de acessar algum recurso, já os outputs representam a resposta do sistema a solicitação realizada pelo usuário (LAU, 2020).

Na Fig.16 é realizada a modelagem de eventos para o caso mais simples possível, que seria o acesso do usuário ao endereço eletrônico da aplicação, onde o requisito descrito no Apêndice A define que o usuário não logado terá acesso a uma página contendo todos os modelos públicos compartilhados pela comunidade que utiliza o software. Nesse caso, o input se trata do acesso ao endereço eletrônico da aplicação, sendo o evento desencadeado o de acesso aos modelos públicos disponíveis e que por consequência gera um output que é a visualização da página contendo a listagens de todos os modelos públicos compartilhados pelos usuários.

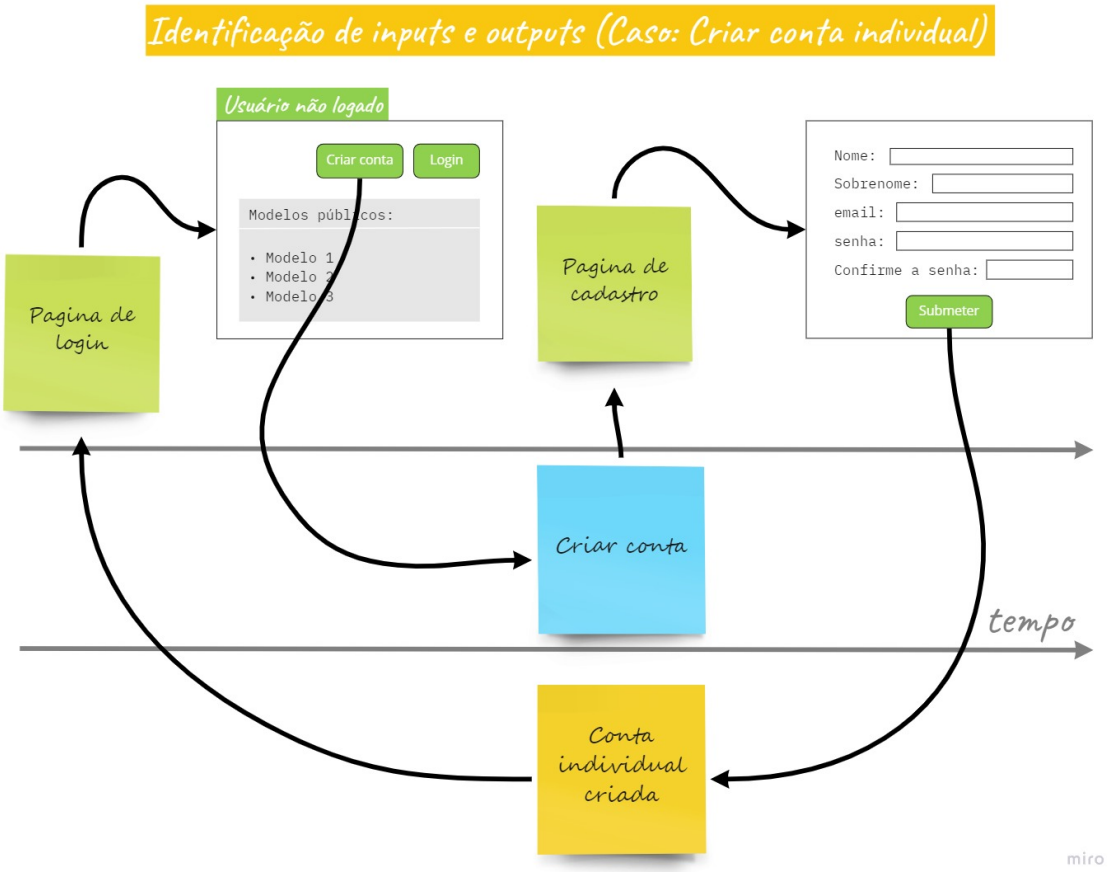
Figura 16 – Acesso sem login



Fonte: De autoria própria

No segundo cenário de interação do usuário com o sistema, temos representado na Fig.17 o cadastro de um novo usuário na plataforma. Assim que é acessado a opção de criar uma conta é gerado um output, que por sua vez, é um redirecionamento para uma página de cadastro. É importante notar que o evento só ocorre quando o usuário submete seu cadastro e obtêm sucesso. Neste trabalho a intenção está em mostrar como ocorre as iterações básicas do usuário com o sistema, visto que, iterações mais complexas aumentam significativamente o número de eventos o que tonaria as figuras mais densas em informação. Outras iterações que apresentam mais complexidade podem ser consultadas no Apêndice B deste trabalho.

Figura 17 – Cadastro de novo usuário



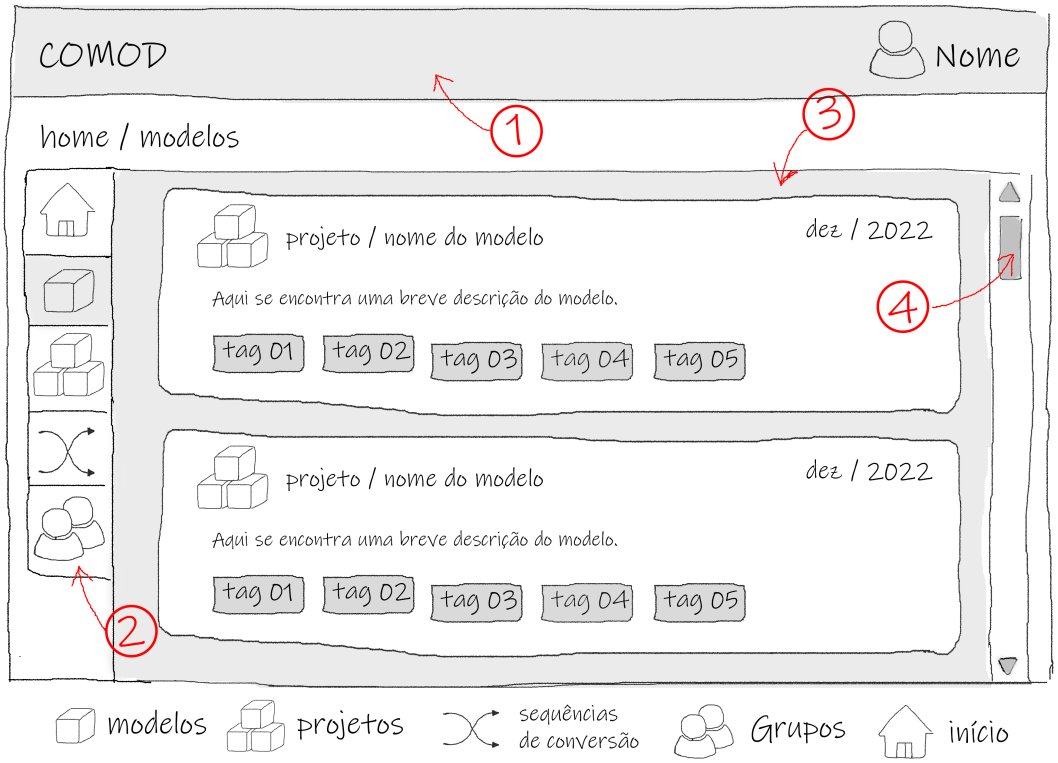
Fonte: De autoria própria

4.2 INTERFACE DO USUÁRIO

4.2.1 Wireframes

Com a etapa de modelagem dos eventos concluída, já é possível ter uma visão mais clara de quais elementos irão compor a aplicação e com isso é esboçado a ideia de como será o software utilizando wireframes. A ideia dos wireframes esta em possibilitar uma visão inicial do que as partes interessadas estão querendo comunicar sem precisar se preocupar com um design bem elaborado, visto que, se trata apenas de um rascunho de como será o conceito da aplicação. É importante destacar que os wireframes são uma espécie de rascunho do protótipo ou wireframe final, pois, logo em seguida dessa rodada de desenhos e discussões, o UX designer revisa esse material em forma de wireframe mais formal feito em algum software (TEIXEIRA, 2014).

Figura 18 – Wireframe da interface do usuário



Fonte: De autoria própria

No software deste trabalho a ideia está em aproveitar conceitos já existentes em outras aplicações web como o github, por exemplo. O GitHub (2022) é tomado como base, pois apresenta uma filosofia similar ao que a plataforma desenvolvida neste trabalho propõe, visto que, a ideia esta em compartilhar projetos. O conceito base da aplicação é que seja listado de forma intuitiva para o usuário seus: modelos, projetos, grupos e sequências de conversão de forma clara e seja possível visualizar suas informações básicas sem a necessidade de cliques adicionais, pois a ideia aqui é que quanto menos cliques e trocas de páginas melhor para a experiência do usuário. Em seu rascunho inicial, conforme observado na Fig.18 são esboçados os elementos que irão compor a aplicação, os quais são:

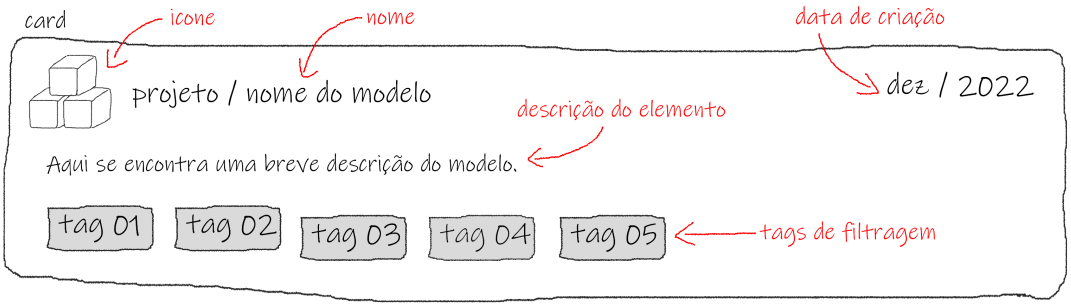
1. Barra de menus com logotipo da aplicação e também um botão para login ou no caso do usuário logado, exibir seu nome de usuário.
2. No caso do usuário logado, possuir uma barra de opções, onde será possível acessar os recursos disponíveis da aplicação de acordo com seu nível de acesso, onde os recursos mais básicos disponíveis são: criar, editar e deletar projetos, modelos e sequências de conversão.
3. Uma área com listagens de card‘s onde é intuitivamente mostrado ao usuário seus elementos disponíveis como: projetos, modelos, sequências de conversão e grupos. Conforme observado na Fig. 19 a ideia é que esses card‘s sejam

genéricos, possuindo um ícone que deixe claro se é um modelo, projeto, grupo ou sequência de conversão. Os card‘s também devem conter uma descrição fornecida pelo usuário, sua data de criação e tags relacionadas a esse elemento, de modo a facilitar sua busca.

4. A página também possui paginação caso a quantidade de elementos ultrapassem a quantidade configurada a ser exibida na tela.

Caso o elemento possua algum "pai", ou seja, um elemento acima dele, o seu card será conforme ilustrado na Fig.19. Modelos fazem parte de um projeto por isso o nome deste projeto vem antes do modelo, de modo que, seja mostrado de forma clara essa relação de dependência. Esses diferentes níveis de dependência podem ser consultado na Fig.14(a).

Figura 19 – Wireframe para representação genérica de um elemento

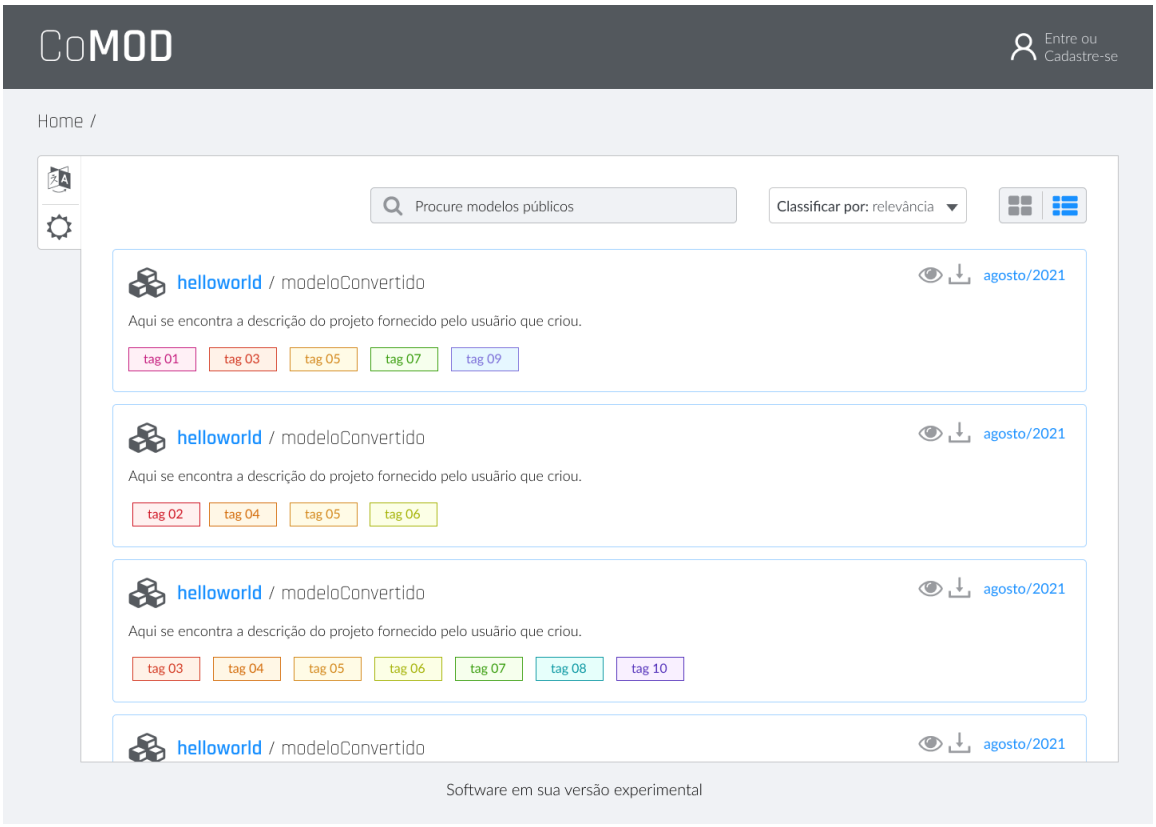


Fonte: De autoria própria

4.2.2 Protótipo

Após essas discussões e rabiscos iniciais todo o material gerado é passado para um wireframe mais formal, o Figma foi a ferramenta utilizada para a sua confecção. Figma é um software para edição gráfica de vetor e prototipagem de projetos de design muito utilizado pelos UI/UX design para seus estudos, prototipagem e validação. Na Fig.20 é mostrado o design final da aplicação baseado da Fig.18 para o caso do usuário não autenticado na plataforma, obviamente, novos elementos foram inseridos na medida que a interface foi desenhada. Algumas delas são: uma caixa de busca por modelos públicos, um select para ordenação dos modelos baseados em sua relevância, data ou nome e também botões com a opção de visualizar os elementos listados em uma ou duas colunas.

Figura 20 – Protótipo para o caso do usuário não logado



Fonte: De autoria própria

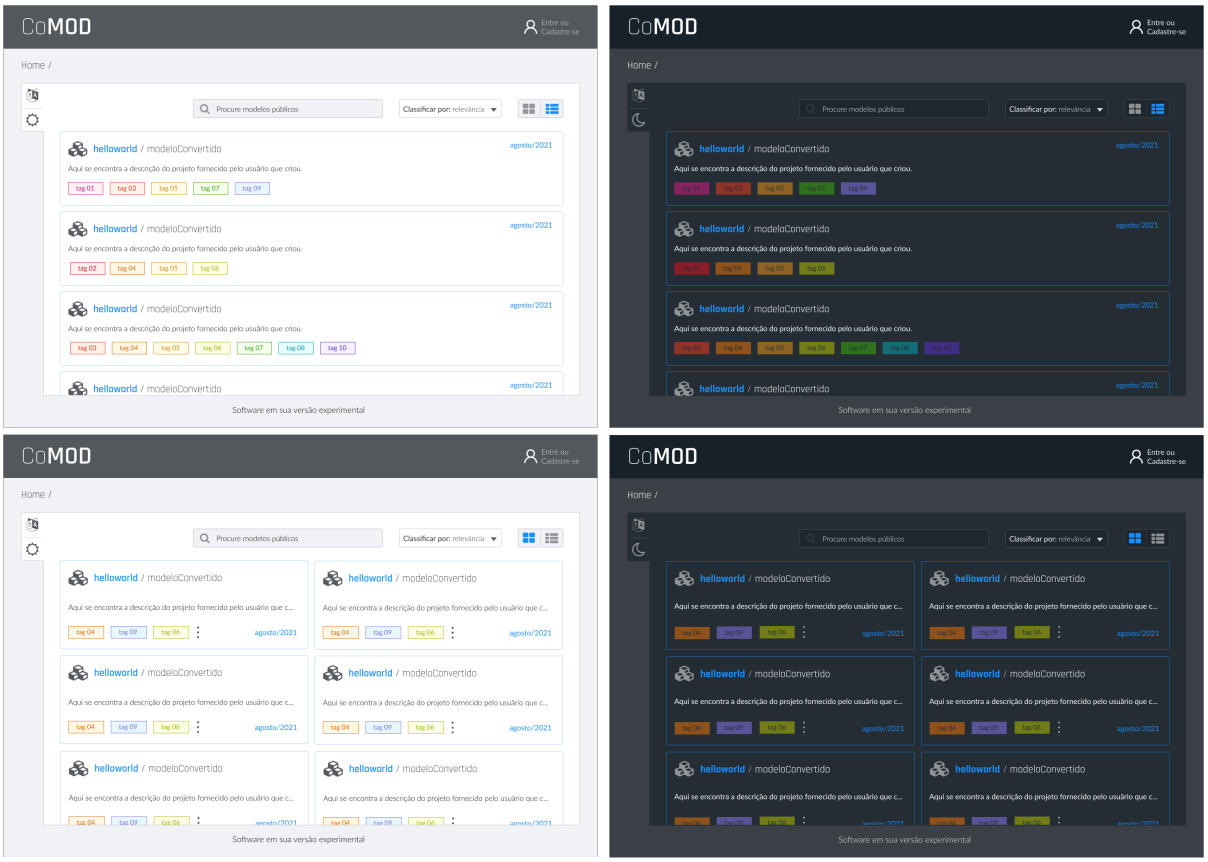
Um ponto importante na experiência do usuário é que a aplicação sempre emita um feedback à medida que os recursos são acessados. Com isso em mente, quando o usuário selecionar alguma opção ou clique em algum botão, que seja visualmente destacado que o elemento está ativo, diante disso, conforme mostrado na Fig.21 é utilizado uma cor que ressalte esses elementos, sendo aqui utilizado o azul-celeste.

Figura 21 – Paleta de cores para os principais elementos da plataforma

Fonte: De autoria própria

Para usuários não autenticados, os requisitos de interface do usuário indicam que a aplicação deve conter as opções básicas como mudança de idioma e tema, com isso, é então inserido esse recurso no wireframe conforme ilustrado na Fig.22.

Figura 22 – Protótipo para o caso de troca de tema



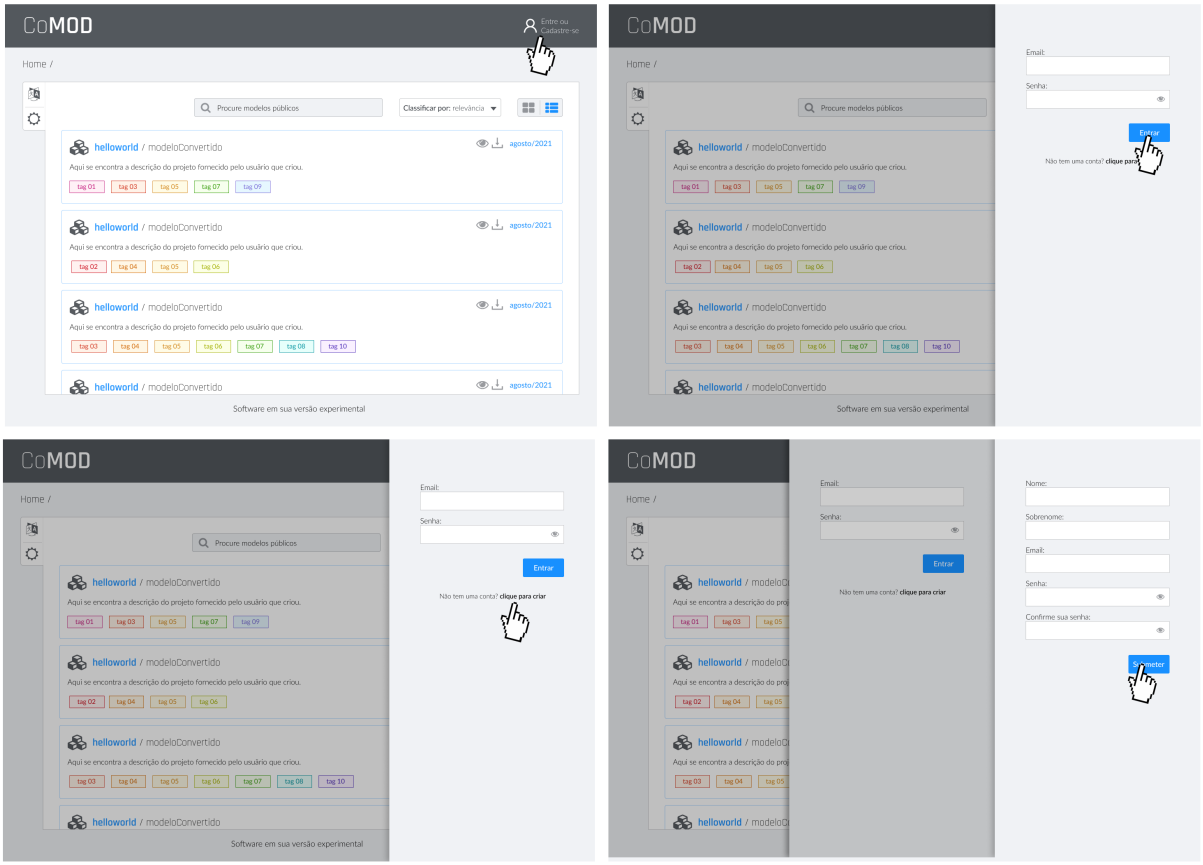
Fonte: De autoria própria

4.2.2.1 Login e Cadastro

Conforme já destacado a premissa base da aplicação está em possibilitar ao usuário o mínimo de cliques e trocas de páginas, diante disso, na maioria dos wireframes que se sucedem é utilizado um recurso conhecido como “drawer”. “Drawer” é um elemento que desliza da borda para tela do usuário, eles são indicados para casos em que se quer evitar sair do contexto da página atual, empregados em formulários ou em processamento de sub-tarefas. Na Fig.23 é representado o wireframe para o processo de login, nessa tela o usuário já cadastrado tera acesso a sua conta através do seu usuário e senha, caso não possua uma conta o processo se dará como mostrado nos wireframes seguintes, onde o usuário ao clicar em um novo cadastro será aberto um segundo drawer que ficará em primeiro plano.

Nesta versão inicial da aplicação, não foi pensado como representar o processo de recuperação de senha do usuário nos wireframes. Isso porque, o objetivo deste trabalho, está em desenvolver uma aplicação funcional com seus recursos básicos de funcionamento, visto que, recursos menos essenciais serão adicionados na medida que surja a necessidade em novas versões do software.

Figura 23 – Protótipo para o caso de login ou cadastro

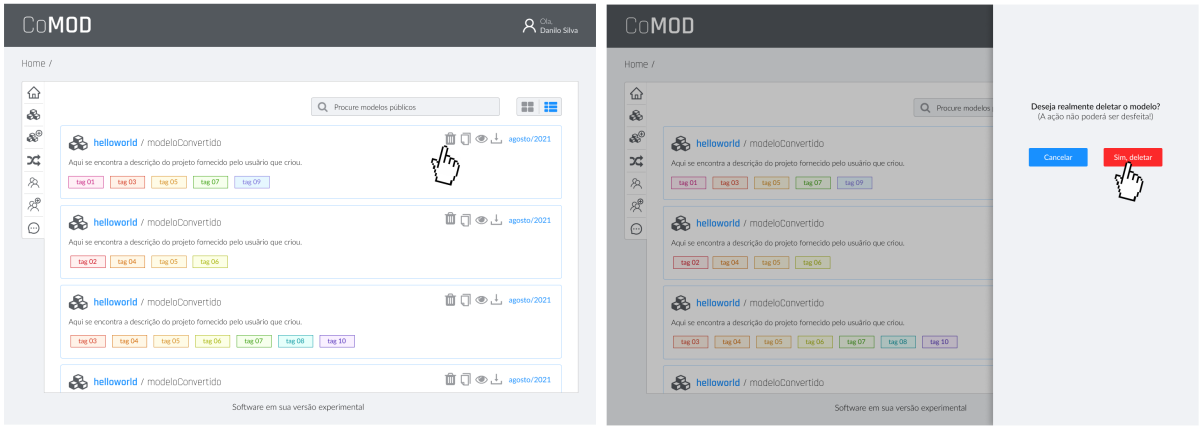


Fonte: De autoria própria

4.2.2.2 Modelos

Considerando que o usuário já está cadastrado no sistema e independentemente do seu nível de permissão, todos terão acesso a funcionalidades básicas do sistema, que no caso dos modelos incluem: deletar, copiar, visualizar e baixar. A Fig.24 ilustra a ação de deletar um modelo, esse mesmo padrão também é adotado para as sequências de conversão, projetos e grupos que estão disponíveis no Apêndice C deste trabalho.

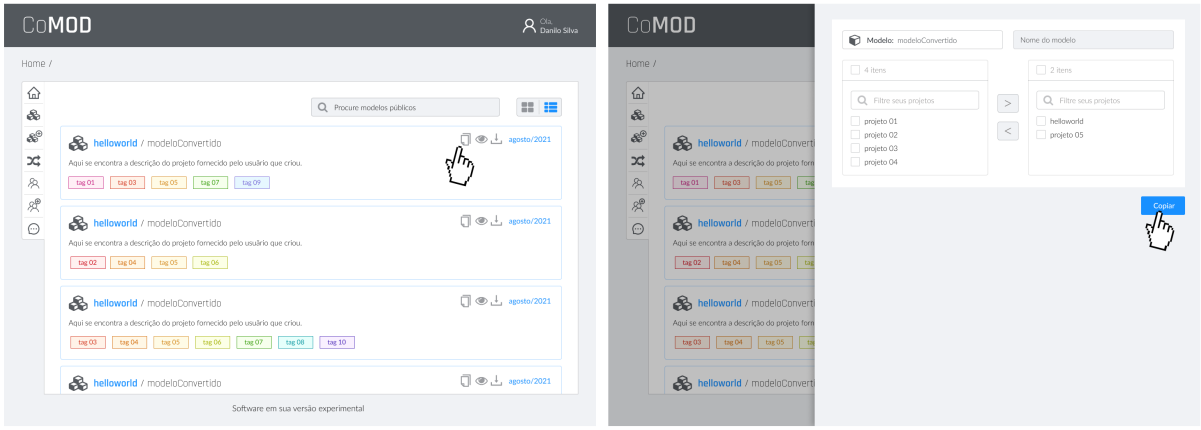
Figura 24 – Protótipo para o caso de exclusão de um modelo



Fonte: De autoria própria

Na Fig.25 é representado o processo de cópia de um modelo, que pode ser feito tanto em modelos públicos ou privados. Para realizar a cópia de um modelo publico o usuário precisa estar logado na plataforma, pois usuários não logados podem apenas visualizar e baixar modelos. No caso de cópia de modelos privados só sera possível realizar pelo criador do modelo ou por algum usuário convidado a participar de um projeto ao qual o modelo está contido.

Figura 25 – Protótipo para o caso de cópia de um modelo



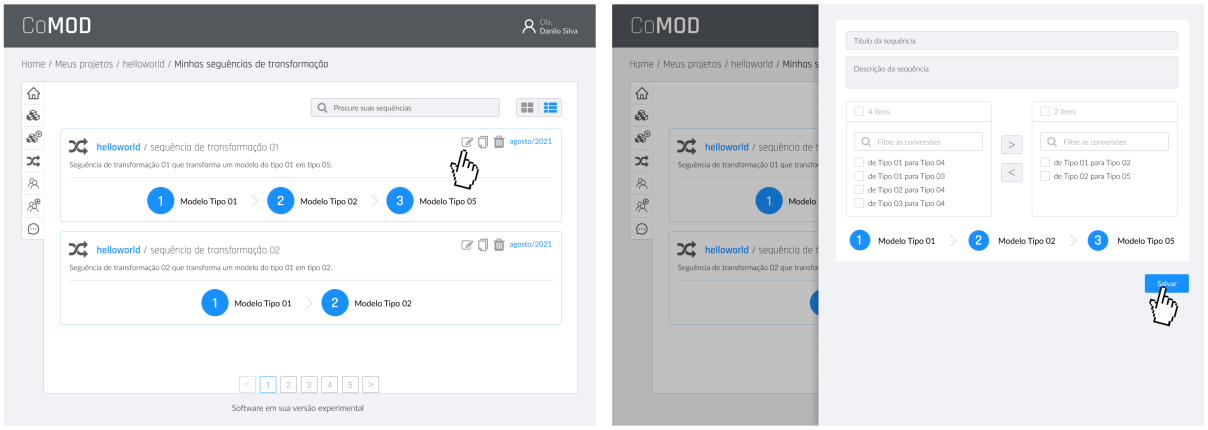
Fonte: De autoria própria

4.2.2.3 Sequências de conversão

As sequências de conversão visam converter um modelo para outro, sendo que, assim como os modelos, as sequências estão associados a um projeto. Uma sequência pode ser editada, copiada ou excluída pelo usuário que criou o projeto. Na Fig. 26 está indicado o processo de edição de uma sequência, onde obrigatoriamente elas devem ter um título, descrição e um pelo menos uma sequência atômica. O termo

sequência atômica é utilizado aqui, no intuito de indicar o processo de conversão mais simples possível, que se trata da conversão de um modelo para outro, sendo possível agrupar várias dessas sequências atômicas para gerar uma sequência de conversão.

Figura 26 – Protótipo para o caso de edição de uma sequência

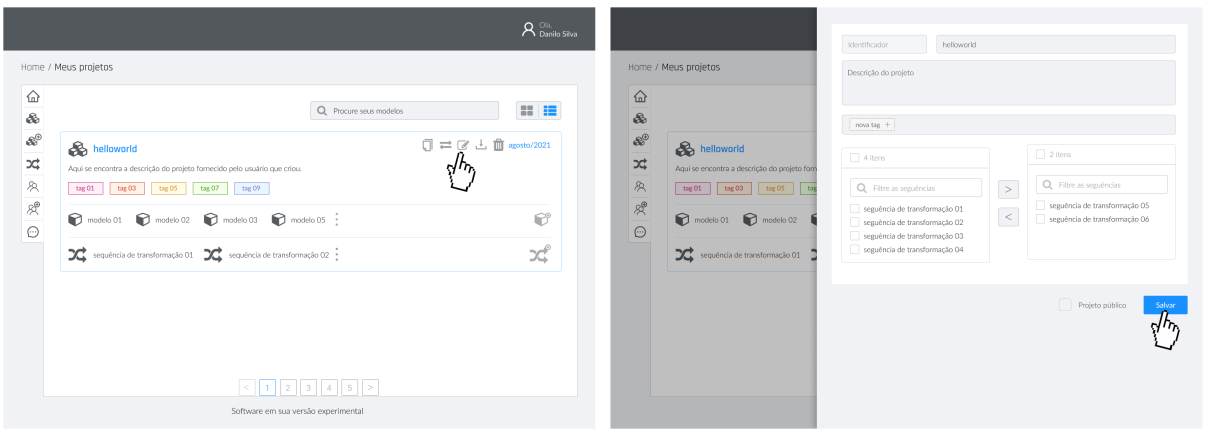


Fonte: De autoria própria

4.2.2.4 *Projetos*

Projetos podem ser copiados, editados, baixados e excluídos. O processo de edição ou criação de um projeto conforme indicado na Fig.27 exige um título, descrição e tags relacionadas ao projeto, onde ele pode ser indicado como publico ou privado. Projetos públicos estarão disponíveis para outros usuários utilizarem, sendo que, projetos privados estão restritos ao usuário que criou ou a usuários convidados a participar de grupo que contemple o projeto.

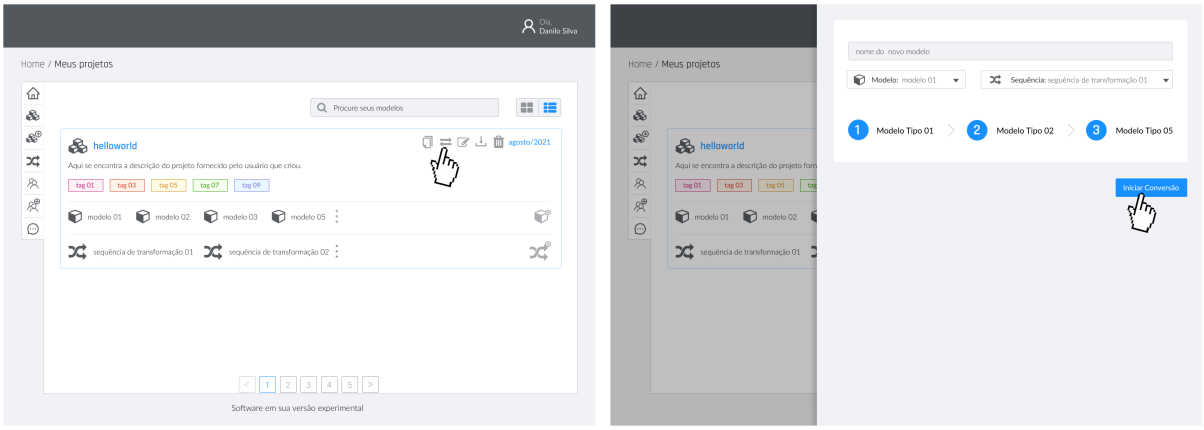
Figura 27 – Protótipo para o caso de criação ou edição de um projeto



Fonte: De autoria própria

Só é possível converter modelos através de projetos, na Fig.28 é mostrado esse processo de conversão de modelos, sendo que, ao clicar na opção de converter é aberto um drawer com a lista de todos os modelos e sequências de conversão atrelados ao projeto. Com o nome do novo modelo preenchido e modelo com sequência de conversão selecionados é então realizado o processo de conversão.

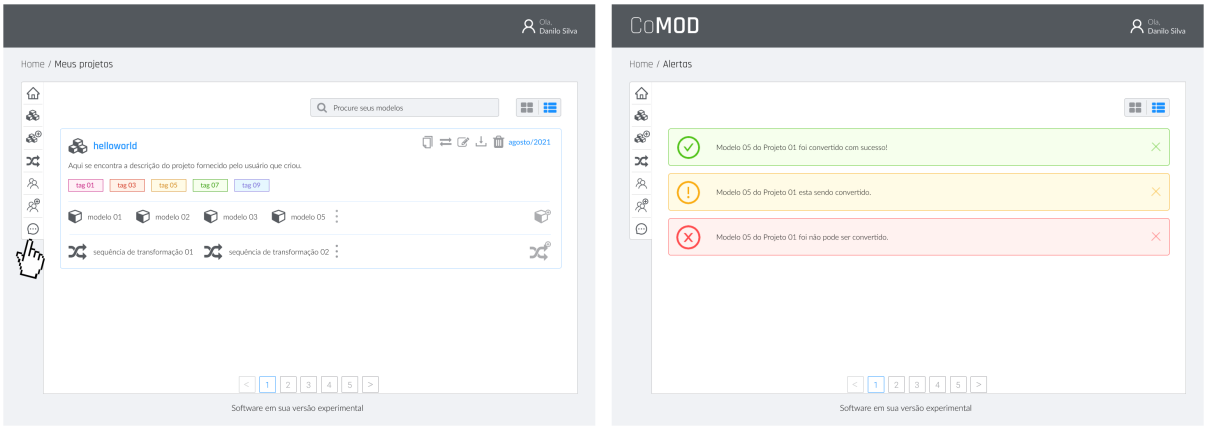
Figura 28 – Protótipo para o caso de conversão de um modelo



Fonte: De autoria própria

Um processo de conversão não ocorre de forma instantânea, pois, cada solicitação é enfileirada de modo que as conversões são processadas conforme a sua ordem de chegada. Esse processo de conversão pode ser acompanhado pelo usuário através no menu de avisos conforme indicado na Fig.29.

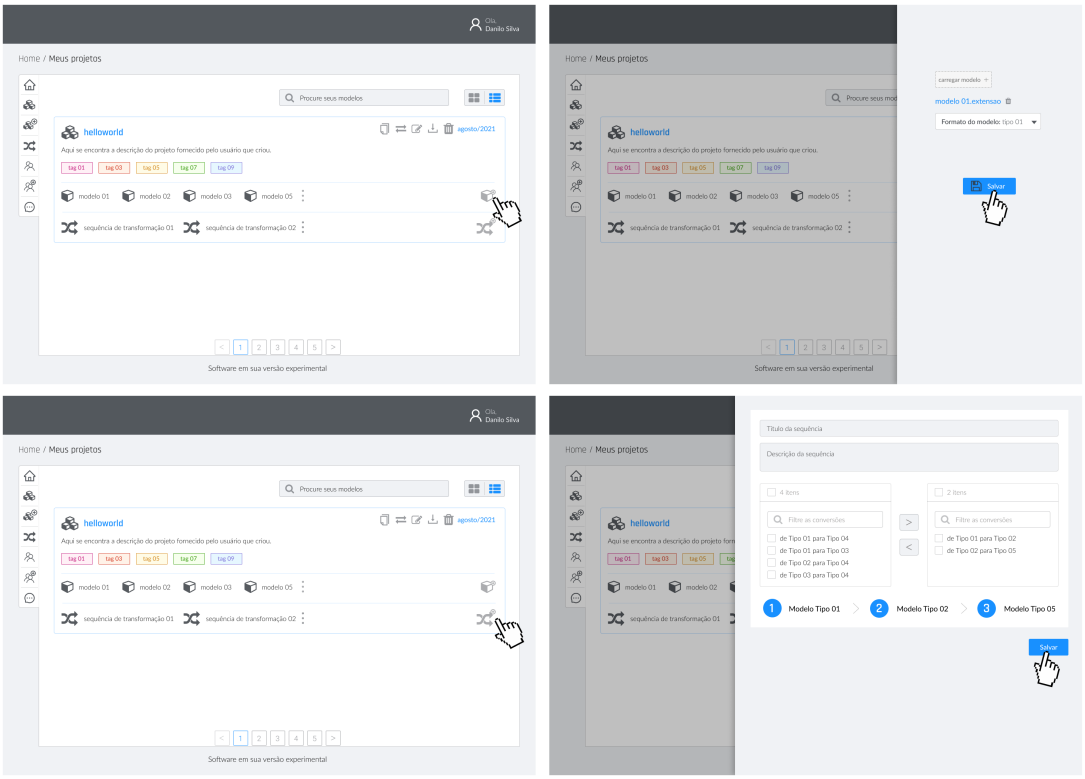
Figura 29 – Protótipo para o caso de visualizar alertas



Fonte: De autoria própria

Um recurso muito importante e deve estar presente está em adicionar novos modelos e sequências de conversão a um projeto. Conforme ilustrado na Fig. 30 o usuário tem disponível a listagem de alguns modelos e sequências contidos no card do projeto, sendo que, a opção de adicionar novos se encontra logos apos estas listagens. Para carregar um novo modelo, o mesmo deve ser de um formato suportado na plataforma, onde na sequência de conversão o processo de criação é o mesmo para o processo de edição que já foi explicado na seção anterior.

Figura 30 – Protótipo para o caso de adicionar um modelo ou sequência ao projeto

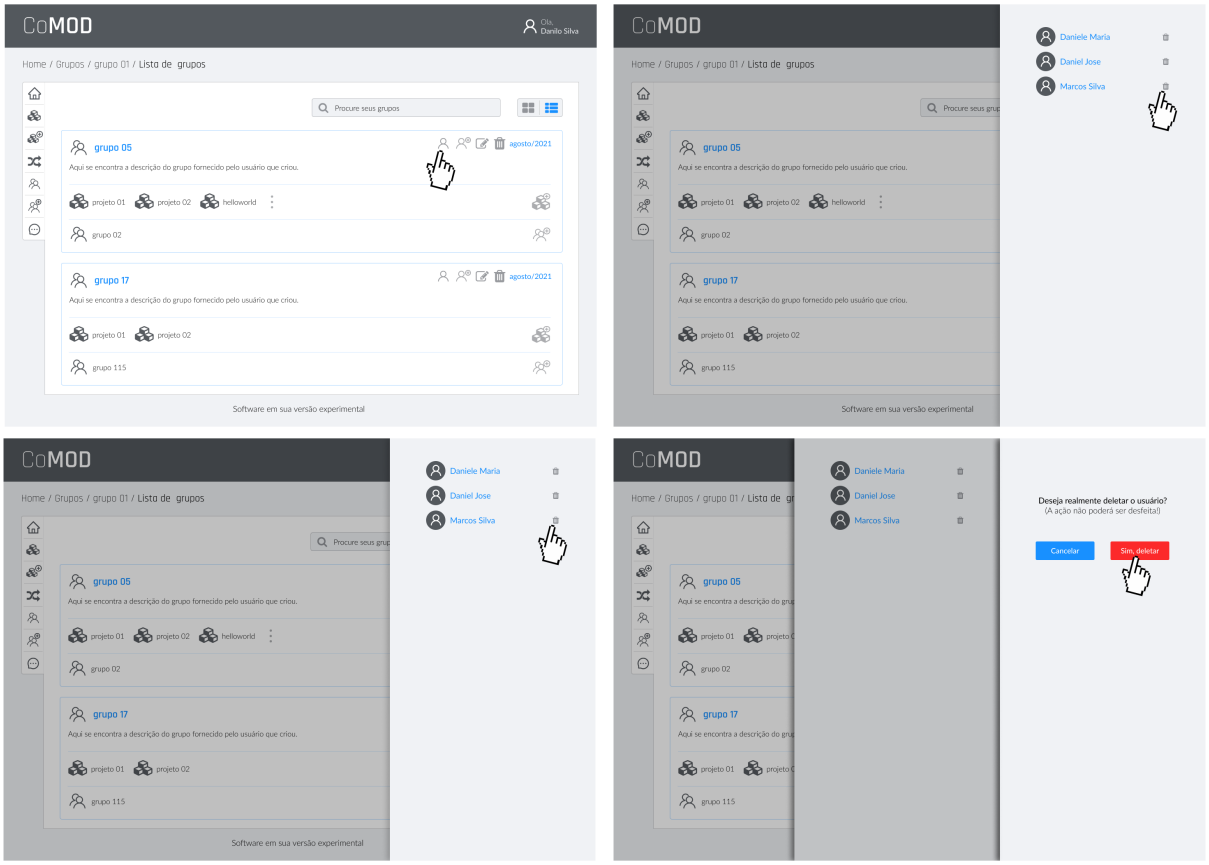


Fonte: De autoria própria

4.2.2.5 Grupos

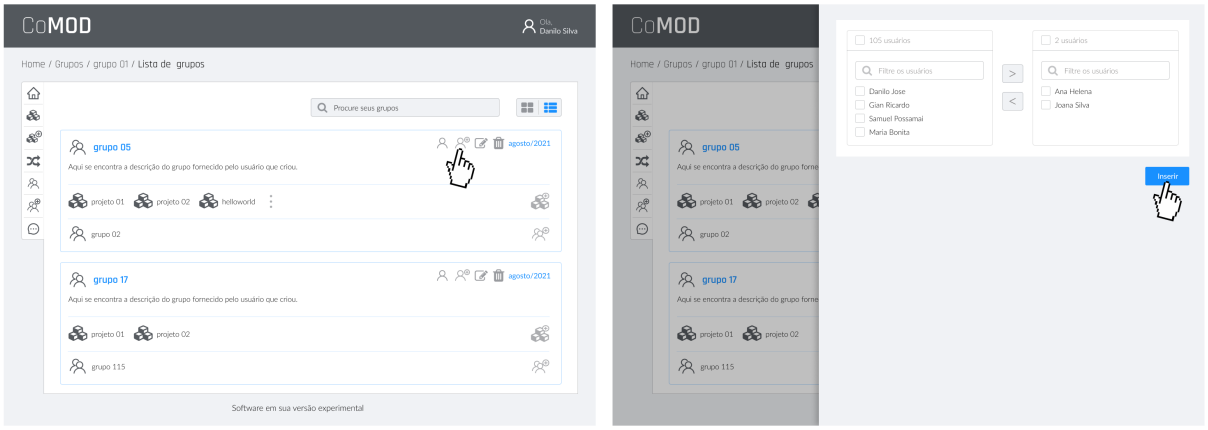
Se um usuário criar um projeto privado e desejar compartilhá-lo com outros usuários, esta ação será possível mediante a associação do projeto a um grupo. Grupos podem ser compostos por outros grupos e também conter projetos, sendo que operações como: criar, editar, excluir, inserir membros ao grupo, deletar membros do grupo são operações previstas a serem suportadas na plataforma. Na Fig. 31 e 32 é ilustrado a operação de remover um usuário do grupo e também a de inserir novos usuários ao grupo. No processo de remoção, é primeiramente acessado a listagem de todos os usuários que membros do grupo, onde a opção de delatar se encontra logo apos o nome de usuário.

Figura 31 – Protótipo para o caso de consultar ou excluir um membro do grupo



Fonte: De autoria própria

Figura 32 – Protótipo para o caso de incluir um membro ao grupo

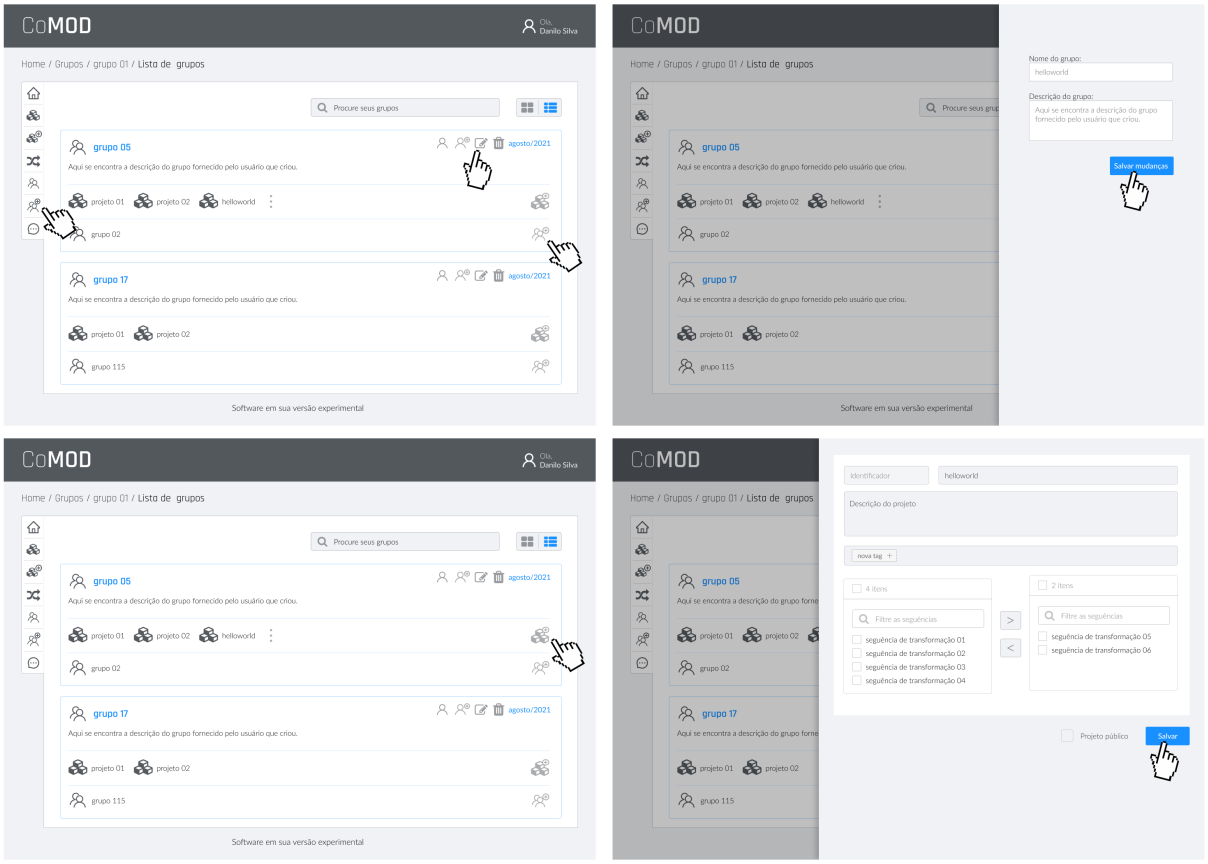


Fonte: De autoria própria

Em relação a inserir um novo usuário a um grupo, é disponibilizado no painel uma lista previa de usuários, sendo que, para inserir um usuário em específico cadastrado na plataforma é preciso fornecer seu endereço de e-mail. No momento em que o usuário começa a digitar um endereço de e-mail é retornado a lista de usuários correspondentes aos caracteres fornecidos. É importante frisar que mesmo que um usuário comece a fazer parte de um grupo e passe a ter acesso aos projetos, modelos

e sequências do grupo, o mesmo possuirá um nível de permissão de convidado, com isso, ações de editar e excluir não são permitidas.

Figura 33 – Protótipo para o caso de edição ou criação de um grupo



Fonte: De autoria própria

Conforme ilustrado na Fig.33 operações de edição e criação de um grupo resultam em um mesmo painel onde sua ação vai depender do contexto em que o drawer foi acionado. Para o caso em que se deseje criar um grupo e que o mesmo não precise fazer parte de outro, a opção deve ser acessado pela barra de menus lateral, no entanto, se o intuito é que um grupo já existente passe a conter outros grupos, então a inserção é realizada dentro do card correspondente ao grupo em questão. O processo de criação de um novo projeto é similar ao que foi mostrado na seção anterior relativo à edição de um projeto.

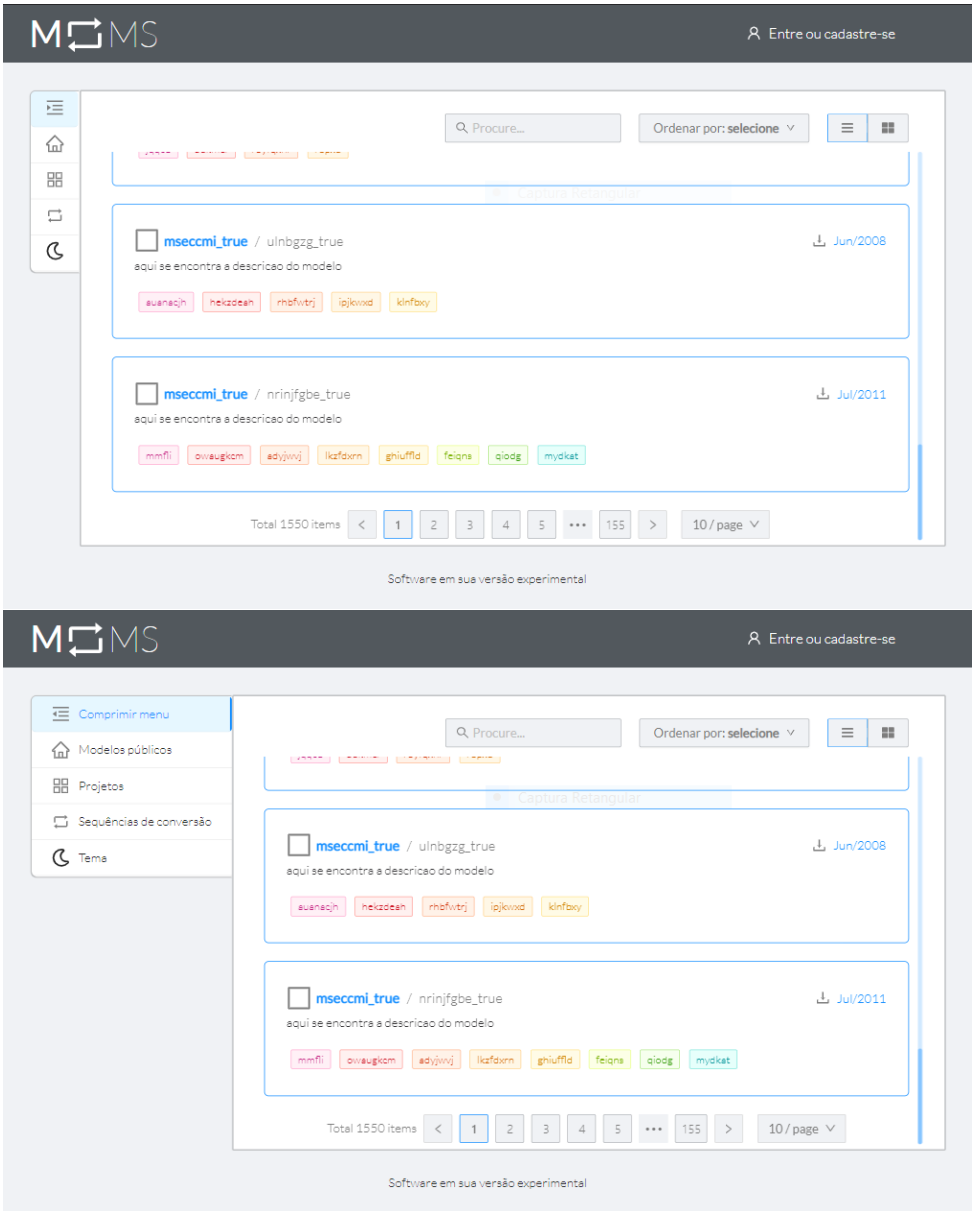
4.2.3 Software

Nesta seção, o objetivo é mostrar algumas telas do software implementado conhecido como MOMS (abreviação do inglês para Model-Oriented Management System), de modo que, possa gerar uma comparação com o protótipo desenvolvido. Detalhes como o passo a passo de cada interação quando o usuário clica em algum botão são omitidos aqui, visto que, todo esse processo detalhado pode ser consultado

na seção anterior que trata do protótipo do software. Algumas telas aqui não listadas podem ser consultadas no Apêndice C deste trabalho.

Na Fig. 34(a) é ilustrado a tela inicial do MOMS quando o usuário não logado acessa a aplicação web, nesta implementação, além do que foi previsto no protótipo é inserido um recurso de expandir a barra de menus lateral, conforme ilustrado na Fig. 34(b). Esse recurso é interessante para a experiência do usuário, pois, mostra de forma explicita a opção que esta sendo acessada. Também, na medida que o software estava sendo implementado e testado, surgiu a necessidade de disponibilizar além dos modelos públicos, as sequências de conversão e os projetos, isso obviamente, para o usuário não autenticado.

Figura 34 – Implementação para o caso do usuário não logado

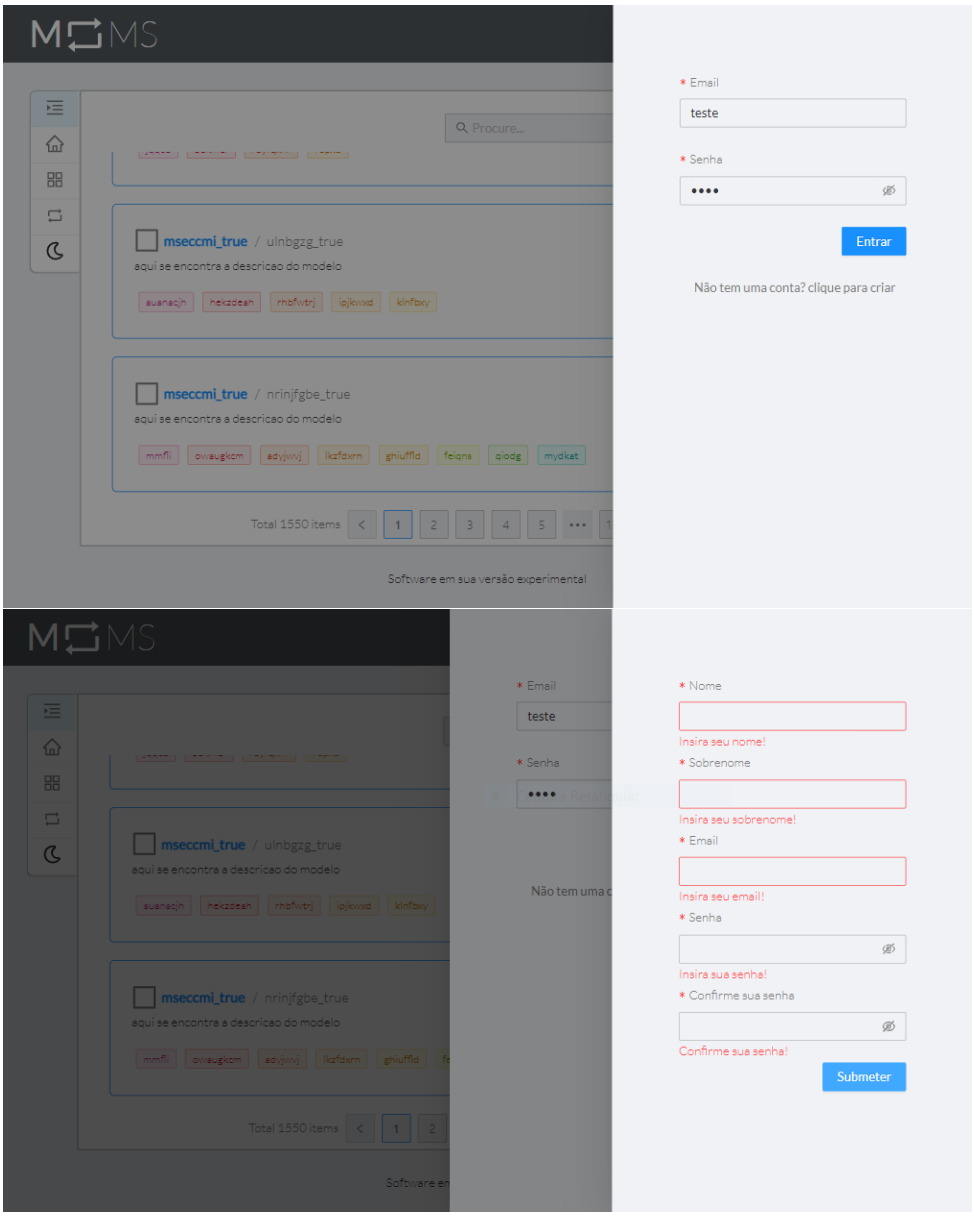


Fonte: De autoria própria

4.2.3.1 Login e Cadastro

O processo de login ou cadastro de usuário conforme ilustrado na Fig. 35 não difere do que é mostrado no protótipo da seção anterior. Um recurso não previsto no wireframe e que também não foi implementado neste trabalho é o processo de recuperação de senha. Obviamente, esse recurso é necessário para que os requisitos de software levantados sejam atendidos, no entanto, como a premissa base deste trabalho é realizar um estudo de interface de usuário que possibilite uma experiência fluida e torne intuitivo seu uso, é considerado que essa proposta foi atendida, visto que, o trabalho aqui realizado é a base para as próximas versões da aplicação.

Figura 35 – Implementação para o caso de login ou cadastro



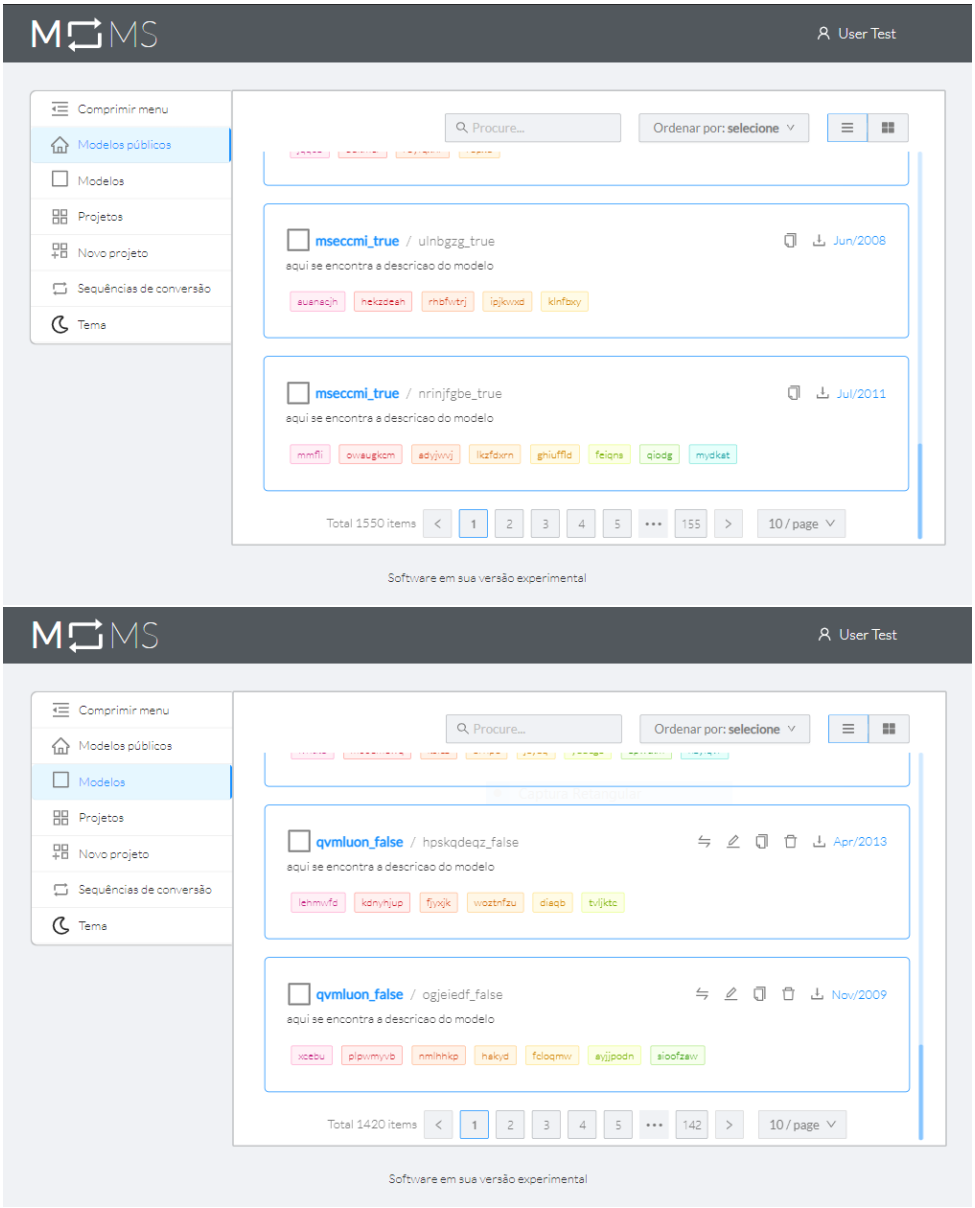
Fonte: De autoria própria

4.2.3.2 Modelos

Conforme indicado na Fig. 36 é mostrado a listagem dos modelos tanto públicos quanto privados para o caso de um usuário logado. É importante notar, que conforme especificado nos requisitos de software, que um usuário logado pode copiar algum modelo listado como publico para algum projeto de sua autoria, sendo que, seus modelos privados além de serem copiados, podem ser editados e excluídos, além é claro que qualquer modelo pode ser baixado como um aquivo zip.

Um ponto importante, que apesar de o ícone para a conversão de modelo estar disponível em cada card, não foi possível implementá-lo em tempo hábil. Com isso, é previsto a inclusão desta funcionalidade em novas versões da plataforma.

Figura 36 – Implementação para o caso de acesso aos modelos públicos e privados

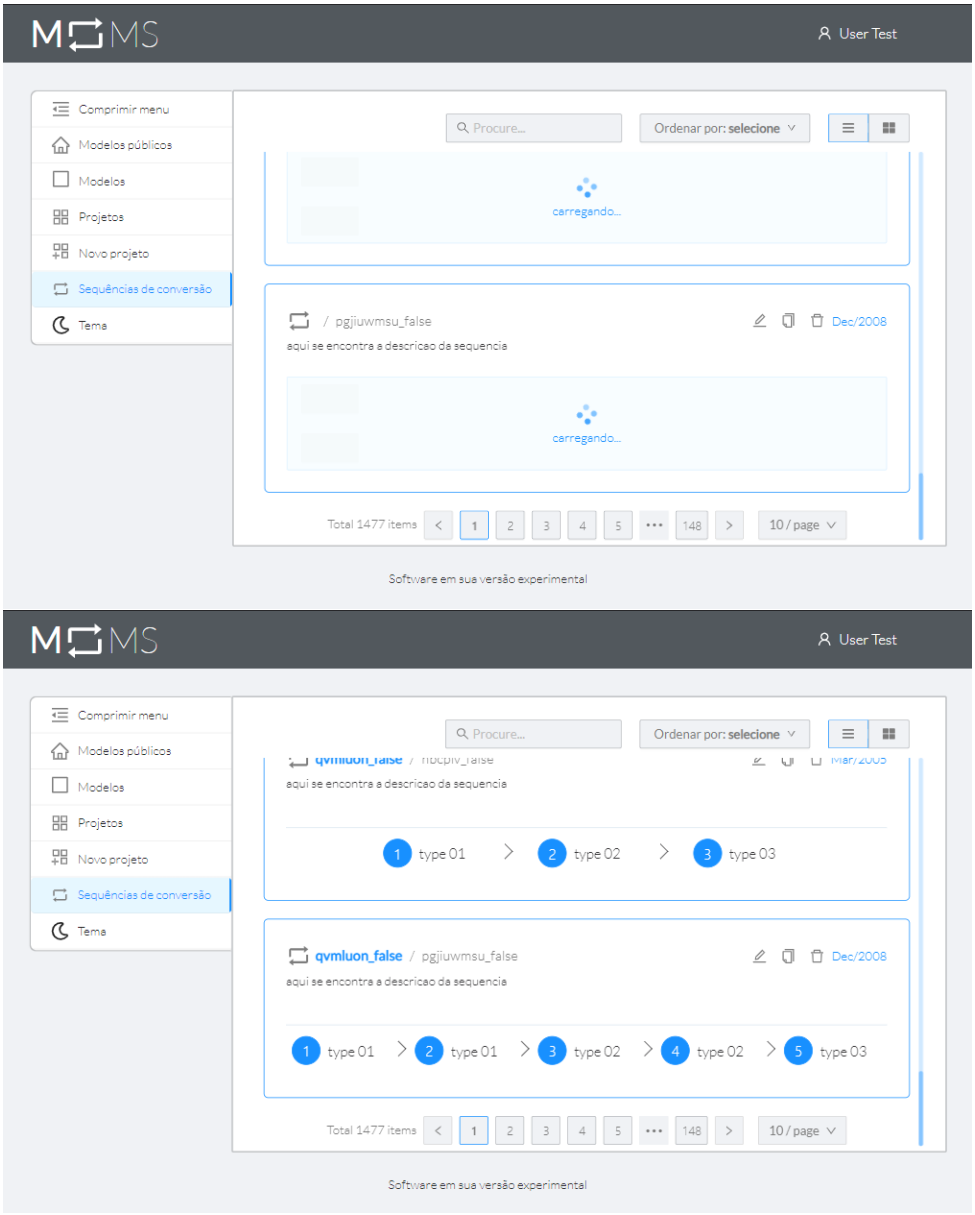


Fonte: De autoria própria

4.2.3.3 Sequências de Conversão

Nas sequências de conversão conforme indicado na Fig. 37 o usuário logado tem acesso as suas sequências de conversão clicando no botão indicado. É importante notar, que as sequências de conversão listadas nesta opção são as de autoria do usuário, visto que, caso seja necessário acessar alguma sequência de conversão pública disponibilizada pela comunidade, o usuário deve efetuar o logout na plataforma e digitar '/sequences' na URI do navegador.

Figura 37 – Implementação para o caso de acesso às sequências privadas

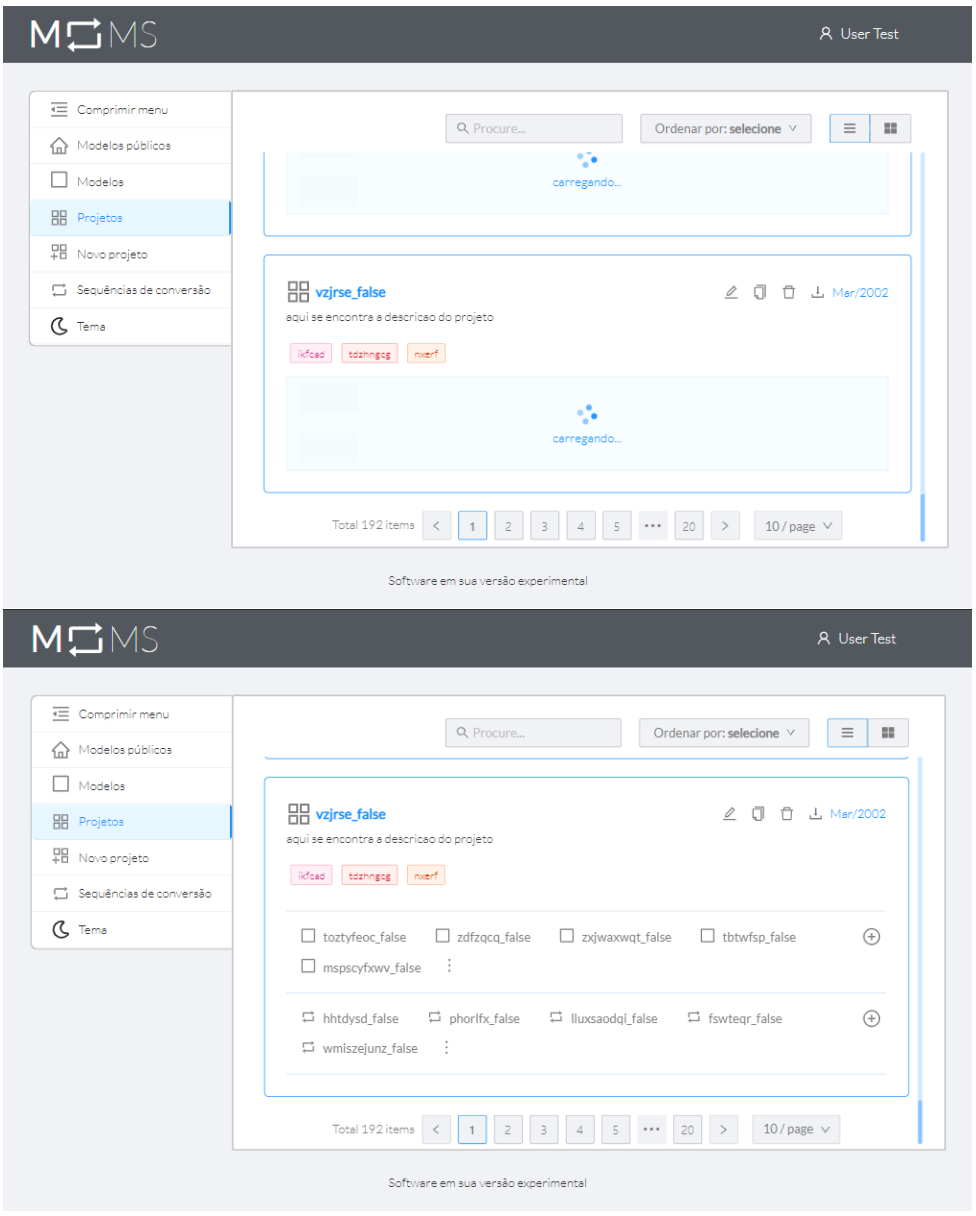


Fonte: De autoria própria

4.2.3.4 Projetos

A Fig. 38 ilustra o caso em que o usuário acessa seus projetos e isso inclui tanto seus projetos públicos quanto privados. Caso o usuário deseje acessar projetos públicos disponibilizado pela comunidade, o mesmo procedimento deve ser seguido como feito nas sequências de conversão públicas, sendo necessário realizar o logout e digitar na URI da plataforma '/projects'.

Figura 38 – Implementação para o caso de acesso aos projetos privados



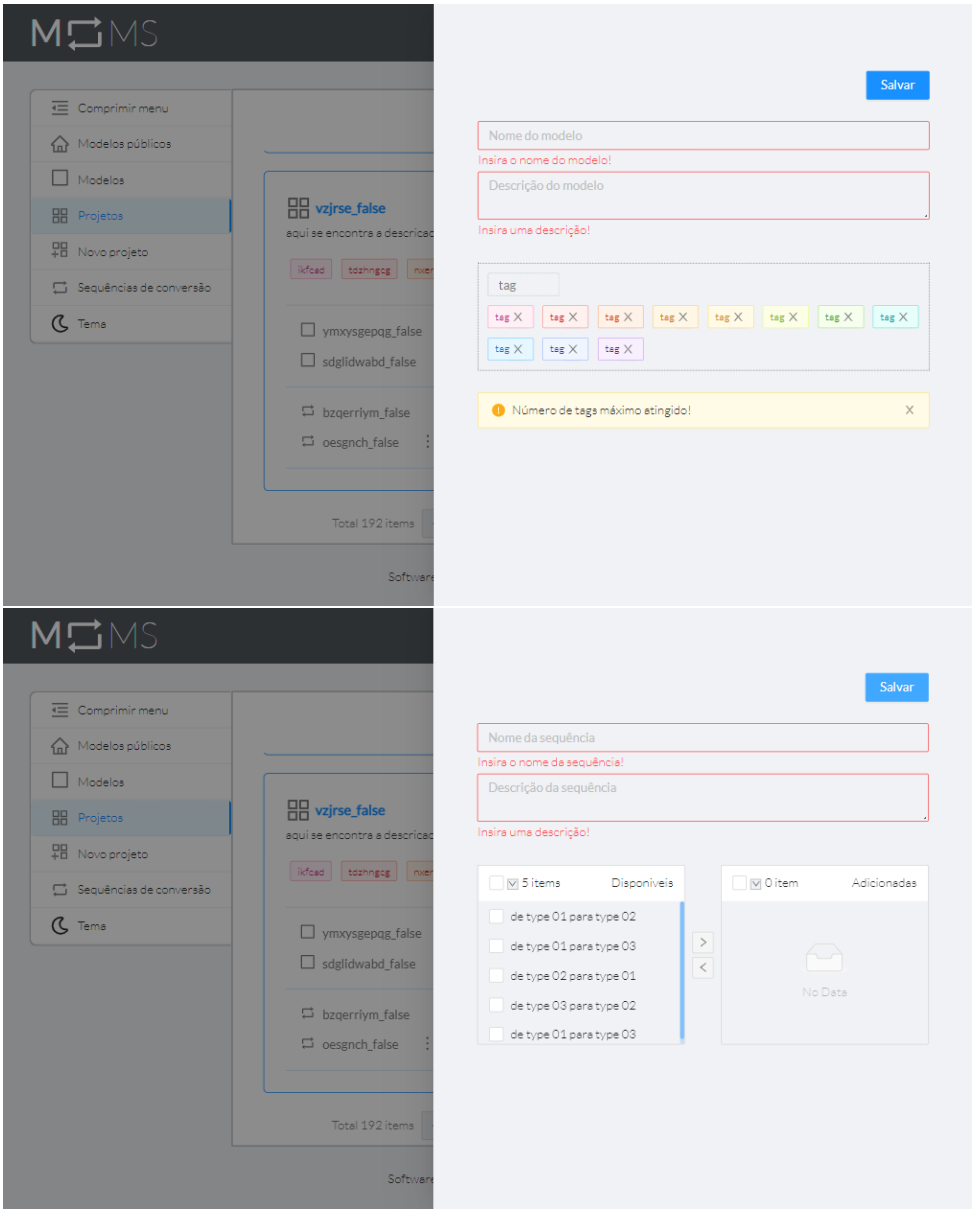
Fonte: De autoria própria

Obviamente, se o intuito do usuário é copiar uma sequência de conversão ou um projeto público, essa ação se torna inviável. Esse tipo de situação já foi previsto no momento em que os requisitos de software foram levantados e discutidos, com isso, a plataforma é estruturada, de modo que, a inclusão de novas funcionalidades

se torne algo simples. Esse recurso de acesso a sequências e projetos públicos serão adicionados na medida que a plataforma for atualizada em versões futuras.

Na Fig. 39 é ilustrado o caso em que o usuário insere uma novo modelo ou sequência a um projeto. No caso da criação um modelo é importante destacar que a plataforma não possui o recurso de carregar seu arquivo para a nuvem, sendo possível apenas, criar um modelo com nome, descrição e tags. O recurso não é disponibilizado em primeiro momento pelo fato da aplicação ser alimentada por uma API 'fake' que não possui o recurso implementado, sendo que, essa funcionalidade será disponibilizada quando uma API definitiva alimentar a plataforma.

Figura 39 – Implementação para o caso de adicionar um modelo ou sequência de conversão a um projeto



Fonte: De autoria própria

4.2.3.5 Grupos

Como já destacado, a plataforma apresenta uma certa complexidade, com isso, alguns recursos previstos no protótipo não foram implementados como, por exemplo, a opção da plataforma suportar grupos. Os grupos são muito importantes, pois só por meio deles é possível compartilhar algum projeto, modelo ou sequência privados com outros usuários. A ideia é que a plataforma passe a ter esses recursos na medida que suas versões sejam atualizadas. Outro ponto importante está no fato de que a plataforma suporte futuramente a inserção de grupos e projetos já existentes a um grupo, visto que, atualmente o protótipo aqui apresentado prevê apenas a criação de um novo projeto ou grupo no grupo em questão.

4.3 LIMITAÇÕES E DISCUSSÕES

As diferentes fases que compõe o desenvolvimento de um software, não são etapas triviais, pelo contrário, no momento do brainstorming, levantamento de requisitos, projeto e desenvolvimento, é essencial manter todos os membros da equipe na mesma página. Anderson et al. (2010) destaca que esse é um ponto critico no incio de qualquer projeto, pois, é fundamental transmitir para as partes interessadas o que esta sendo proposto e qual o objetivo final desejado.

Neste trabalho, a etapa de desenvolvimento, projeto e levantamento de requisitos voltado a interface e experiência do usuário é desempenhado por uma única pessoa. Essas múltiplas tarefas, acarretam alguns problemas, sendo uma delas, a impossibilidade de realizar um estudo mais detalhado em cada fase de projeto. Feedbacks e trocas de ideias com diferentes membros, neste trabalho são impossibilitas, pelo simples fato de tarefas serem concentradas em uma pessoa.

4.4 CONSIDERAÇÕES FINAIS

O trabalho aqui proposto, serve de base para futuros estudos e evoluções da plataforma que eventualmente surgirão. A integração do front-end com uma API e banco de dados, como também, a possibilidade de criar grupos e associá-los a projetos, no intuito de tornar possível o compartilhamento de projetos privados, são algumas das melhorias previstas.

A premissa deste trabalho, está em utilizar uma metodologia de desenvolvimento de software, que possibilite gerar uma documentação da interface gráfica do usuário. Isso obviamente, baseado em uma documentação de requisitos de software. Com os resultados obtidos, os quais incluem: o protótipo da interface gráfica do usuário; documento de requisitos de software e a implementação do frontend, se conclui que a proposta deste trabalho foi atendida.

5 CONCLUSÕES

Para o desenvolvimento da aplicação deste trabalho foi utilizado a metodologia BDUF, onde sua principal característica está em desprender um esforço maior na etapa voltada a interface e experiência do usuário. Conforme indicado por Anderson et al. (2010), não há uma metodologia definitiva para os projetos de software, visto que, dependendo da sua natureza, uma determinada metodologia pode ser mais adequada.

Utilizar métodos que hoje estão amplamente difundidos, como os métodos ágeis, pode parecer a primeira vista mais adequado e agregue mais benefícios. No entanto, o trabalho aqui propõe, elaborar uma documentação para a interface do usuário voltado a um sistema de gestão orientado a modelos, onde, a tarefa de UX/UI design e desenvolvedor da aplicação, está concentrada em apenas uma pessoa. Sendo assim, a troca de interações com feedbacks constantes entre os integrantes, não ocorre, visto que, não há um número suficiente e heterogêneo de pessoas. Em virtude desses pontos levantados, a metodologia BDUF se mostrou mais adequada ao projeto aqui proposto.

Para a elaboração deste trabalho, foi realizado a etapa de brainstorming, dado uma premissa inicial que se baseia no desenvolvimento de um sistema de gestão orientado a modelos, em que seja possível converter diferentes tipos de modelos na plataforma utilizando os principais navegadores para web. No brainstorming é levantado questionamentos no intuito de gerar um entendimento em como estruturar essa premissa na forma de um documento formal de requisitos de software.

Na segunda etapa, a proposta é elaborar uma documentação de requisitos de software que sirva como a base no projeto da interface do usuário. Há diversas ferramentas que podem ser utilizadas na validação desses requisitos, sendo que, a utilizada neste trabalho é a modelagem de eventos. A modelagem de eventos é útil não somente para validar requisitos por meio de um estudo da interação do usuário com a aplicação, mas também, em gerar um esboço inicial de quais elementos devem estar presentes na interface do usuário.

A fase seguinte, compreende em um estudo de UI/UX design, colocando à mesa toda documentação gerada, desde a modelagem de eventos até os requisitos de software. Esse material é a base da elaboração dos wireframes e de um estudo mais elaborado de design tomando a forma de protótipos. O protótipo desenvolvido neste trabalho é fundamental não somente para o cliente interagir com a aplicação sem a necessidade de ser implementada em um primeiro momento, mas também, servir como documentação no momento da construção do software.

Por fim, a última etapa compreende na construção do software. Nesta ultima fase, conforme já previsto, alguns ajustes pontuais precisaram ser realizados, conforme a aplicação estava sendo desenvolvida e testada. Alguns deles incluem: mudanças nos ícones, visto que, os ícones propostos no protótipo são pagos. Também são inseridos elementos que deem um feedback ao usuário no momento em que um modelo é criado, ou o projeto, ou uma sequência. Na barra de menus lateral, quando o usuário não está autenticado é inserido a opção de acessar não só os modelos públicos, como também, as sequências de conversão e os projetos.

Conforme já discutido, a proposta deste trabalho foi desenvolver uma interface do usuário, que deixe uma visão clara do que foi proposto na documentação de requisitos de software. Novos recursos, ajustes nas funcionalidades existentes e mudanças no layout da aplicação, podem ser inseridas na medida que a aplicação for atualizada em versões futuras.

A aplicação aqui proposta ainda necessita realizar a integração com a API responsável por fornecer dados ao front-end. Também, não foi inserido o recurso de criar grupos, funcionalidade essa, já prevista no protótipo apresentado neste trabalho. Conforme já destacado, esse recurso permite que projetos privados sejam compartilhados com outros usuários. O recurso de acompanhar o status de conversão do modelo na fila do servidor, conforme ilustrado na Fig. 29, bem como a recuperação de senha, também não foi implementado. Esses pontos aqui descritos, são propostas para trabalhos futuros, que servirão de continuação e evolução da plataforma.

REFERÊNCIAS

AGNI, E. **Don Norman e o termo “UX”**. 2016. Disponível em: https://uxdesign.blog.br/don-norman-e-o-termo-ux-6dffb3f8d218.

ANDERSON, J. et al. **Effective UI: The Art of Building Great User Experience in Software**. O’Reilly Media, 2010. ISBN 9781449388720. Disponível em: https://books.google.com.br/books?id=I7-IP5P-gdMC.

BECK, K. et al. **The 12 Principles behind the Agile Manifesto**. 2001. Disponível em: https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: http://www.agilemanifesto.org/.

BELGAMO, A.; MARTINS, L. E. G. Estudo comparativo sobre as técnicas de elicitação de requisitos do software. In: **XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná**. [S.l.: s.n.], 2000.

BENETTI, R. **O que é SEO e para que serve?** 2022. Disponível em: https://www.organicadigital.com/blog/o-que-e-seo-e-para-que-serve/.

BOLLINGER, T.; GABRINI, P.; MARTIN, L. Software construction. **SWEBOK**, p. 53, 2001.

BRACKETT, J. W. **Software requirements**. [S.l.], 1990.

DEBASTIANI, C. **Definindo Escopo em Projetos de Software**. Novatec Editora, 2015. ISBN 9788575224298. Disponível em: https://books.google.com.br/books?id=BYTDCAAAQBAJ.

DYMITRUK, A. **Event Modeling: What is it?** 2019. Disponível em: https://eventmodeling.org/posts/what-is-event-modeling/.

FACEBOOK. **React: Uma biblioteca JavaScript para criar interfaces de usuário**. 2022. Disponível em: https://pt-br.reactjs.org/.

FRASER, S. D. et al. No silver bullet"reloaded: retrospective on"essence and accidents of software engineering. In: **Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion**. [S.l.: s.n.], 2007. p. 1026–1030.

GACKENHEIMER, C.; PAUL, A. **Introduction to React**. [S.l.]: Springer, 2015. v. 52.

GARRETT, F. **O que é Figma? Quatro perguntas sobre como usar o site**. 2021. Disponível em: https://www.techtudo.com.br/listas/2021/06/o-que-e-figma-quatro-perguntas-sobre-como-usar-o-site.ghtml.

GITHUB. **Where the world builds software**. 2022. Disponível em: https://github.com/.

GOMES, F. Elementos que influenciam a experiência do usuário na utilização de web sites. **Intercom – Sociedade Brasileirade Estudos Interdisciplinares da Comunicação**, 2015.

GUEDES, M. **O que são aplicações SPA?** 2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-sao-aplicacoes-spa.

ISO Central Secretary. **ISO 9241-210:2019(en) Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems**. Geneva, CH, 2019. Disponível em: https://www.iso.org/standard/77520.html.

ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. **ISO/IEC/IEEE 29148:2011(E)**, p. 1–94, 2011.

KULESZA, R. et al. Evolução das arquiteturas de software rumo à web 3.0. **Sociedade Brasileira de Computação**, 2018.

LANE, C.; KRüGER, N. **How to Write a Software Requirements Specification (SRS Document)**. 2021. Disponível em: https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document.

LAU, E. **Great User Experience Demands Event Modeling**. 2020. Disponível em: https://eventmodeling.org/posts/user-experience-event-modeling/.

MACHADO, F. N. R. **Análise e Gestão de Requisitos de Software–Onde nascem os sistemas**. [S.l.]: Saraiva Educação SA, 2018.

MARINHO, T. **Axios - um cliente HTTP Full Stack**. 2020. Disponível em: https://blog.rocketseat.com.br/axios-um-cliente-http-full-stack/.

PATIL, K. A study of web 1.0 to 3.0. **AGPE THE ROYAL GONDWANA RESEARCH JOURNAL OF HISTORY, SCIENCE, ECONOMIC, POLITICAL AND SOCIAL SCIENCE**, v. 2, n. 2, p. 40–45, Oct. 2021. Disponível em: https://103.160.129.43/index.php/agpe/article/view/45.

PERMINOVA, M. **What is Miro?** 2022. Disponível em: https://help.miro.com/hc/en-us/articles/360017730533-What-is-Miro-.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos Ágeis para Desenvolvimento de Software**. Bookman Editora, 2014. ISBN 9788582602089. Disponível em: https://books.google.com.br/books?id=8rQABAAAQBAJ.

REISSWITZ, F. **Análise De Sistemas: Tecnologia Web & Redes**. [S.l.]: Clube dos Autores, 2008. v. 2.

ROCHA, A. **O que é o Next.js?** 2021. Disponível em: https://blog.geekhunter.com.br/o-que-e-next-js/.

SAWYER, P.; KOTONYA, G. Software requirements. **SWEBOK**, p. 9, 2001.

TEIXEIRA, F. **Introdução e Boas Práticas em Ux Design**. 1. ed. [S.l.]: Casa do Código, 2014. 217 p. ISBN 8566250486.

TEIXEIRA, F. **UX para GPs 2 — o que faz um UX designer**. 2015. Disponível em: https://brasil.uxdesign.cc/ux-para-gps-2-o-que-faz-um-ux-designer-edc6dc3cb4c.

TOTVS, E. **Business Case: quando é utilizado e como construir**. 2021. Disponível em: https://www.totvs.com/blog/negocios/business-case/.

APÊNDICE A - DOCUMENTO DE REQUISITOS DE SOFTWARE

SOFTWARE PARA CONVERSÃO DE MODELOS
DEFINIÇÃO DE REQUISITOS

Versão *1.0*

01/08/2022

HISTÓRICO DE VERSÕES

Versão	Implementada por	Data da Revisão	Motivo da Revisão
1.0	Samuel Possamai	19/10/2021	Definição inicial de requisitos
1.0	Danilo Silva	29/11/2021	Padronização inicial de requisitos
1.0	Danilo Silva	27/05/2022	Revisão dos requisitos e inserção da introdução

Sumário

1. INTRODUÇÃO 4

1.1. PROPÓSITO DA DEFINIÇÃO DE REQUISITOS FUNCIONAIS 4

1.2. PROPÓSITO DA DEFINIÇÃO DE REQUISITOS NÃO-FUNCIONAIS 4

2. RESUMO DE REQUISITOS DE PRODUTO 4

3. REQUISITOS FUNCIONAIS 5

3.1. REQUISITOS FUNCIONAIS DE USO 5

3.2. REQUISITOS FUNCIONAIS DE ACESSO 5

3.3. REQUISITOS FUNCIONAIS DE INTERFACE 6

4. REQUISITOS NÃO FUNCIONAIS 6

4.1. REQUISITOS NÃO FUNCIONAIS DE USO 6

1. INTRODUÇÃO

1.1. PROPÓSITO DA DEFINIÇÃO DE REQUISITOS FUNCIONAIS

Requisitos funcionais tem o propósito de listar os fatores básicos de atendimento do escopo do produto e suas ramificações imediatas, buscando visualizar o essencial para o funcionamento legal, seguro, viável, e útil do produto.

1.2. PROPÓSITO DA DEFINIÇÃO DE REQUISITOS NÃO-FUNCIONAIS

Requisitos não-funcionais tem o propósito de: expandir as ramificações dos requisitos funcionais, definindo os fatores constituintes e/ou apoiadores do essencial; definir a moldura geral de fatores secundários que possuem como objetivo expandir a gama de funcionalidades do produto; e construir a base da existência do funcionamento mínimo do produto.

2. RESUMO DE REQUISITOS DE PRODUTO

As características gerais do produto estão baseadas em um software capaz de cadastrar usuários suportando diferentes níveis de acesso, onde a aplicação deve ser capaz de suportar o carregamento, download, exclusão e conversão de diferentes tipos de modelos. Para ser possível a conversão de modelos o usuário deve utilizar sequências de conversão, de modo que, seja indicado o tipo de modelo de entrada e o tipo de modelo de saída que é suportado na plataforma. Modelos e Sequências de conversão são agrupadas dentro de um projeto, sendo que um projeto, modelo ou sequência de conversão pode ser criado, editado, excluído ou copiado. Projetos são públicos ou privados e podem ser agrupados dentro de grupos onde grupos podem ser compartilhados com outros usuários cadastrados na plataforma.

3. REQUISITOS FUNCIONAIS

3.1. REQUISITOS FUNCIONAIS DE USO

- **RFUS-1.1 -** Armazenar modelos no servidor;
- **RFUS-1.2 -** Permitir transformação entre os modelos suportados;
- **RFUS-1.3 -** Modelos podem ser agrupados para diferentes grupos;
- **RFUS-1.4 -** Permitir grupos dentro de grupos;
- **RFUS-1.5 -** O sistema permite registro de "Usuário individual";
- **RFUS-1.6 -** As restrições de armazenamento e transformação são por projeto;
- **RFUS-1.7 -** Um projeto é um conjunto de modelos pertencente a um grupo ou usuário restrito as operações permitidas;
- **RFUS-1.8 -** Um projeto pode ser criado a partir de outro projeto desde que o usuário ou grupo tenha acesso a ele;
- **RFUS-1.9 -** Um projeto pode ser público ou privado, se for privado e pertencer ao um grupo ele poderá ser acessado pelos membros do grupo e dos subgrupos desde que o usuário responsável permita;
- **RFUS-1.10 -** Projetos públicos podem ser consultados por qualquer usuário;
- **RFUS-1.11 -** Um projeto pode ser manipulado pelos usuários que tem permissão;
- **RFUS-1.12 -** Um projeto tem um usuário responsável;
- **RFUS-1.13 -** Permitir visualização de modelos;
- **RFUS-1.14 -** Permitir edição de modelos.

3.2. REQUISITOS FUNCIONAIS DE ACESSO

- **RFAC-2.1 -** Suportar diferentes tipos de usuários:
    - **RFAC-2.1.1 - Usuário não registrado**: pode consultar o repositório de modelos públicos;
    - **RFAC-2.1.2 - Usuário individual:** com as mesmas atribuições do RFAC-2.1.1, porém com acesso ao serviço de armazenamento e transformação;
    - **RFAC-2.1.3 - Usuário membro de grupo**: com as mesmas atribuições do RFAC-2.1.1, porém com acesso ao serviço de armazenamento e transformação que o "usuário responsável por grupo" permitir;
    - **RFAC-2.1.4 - Usuário responsável por grupo**: com as mesmas atribuições do RFAC-2.1.2, porém pode definir permissão para usuários do grupo e gerenciar usuários do grupo e grupos;
    - **RFAC-2.1.5 - Usuário administrador:** com as atribuições dos usuários anteriores, porém pode gerenciar todos os usuários e gerenciar o uso do sistema.
- **RFAC-2.2 - Usuário membro de grupo** poderá escolher uma ou mais sequência de regras de transformações que o responsável pelo grupo permitiu;
- **RFAC-2.3 -** Gerenciar usuário consiste em poder criar/alterar/excluir/bloquear e desbloquear usuários;
- **RFAC-2.4 -** Gerenciar uso do sistema consiste em verificar a utilização dos serviços de transformação e disponibilizar novos serviços de transformação, cancelar serviços de transformação e acompanhar como os usuários usam o sistema;

- **RFAC-2.5 -** Mais de um usuário pode manipular o projeto, desde que tenha permissão;
- **RFAC-2.6 -** Um usuário pode convidar outro usuário para participar do projeto;
- **RFAC-2.7 -** O usuário responsável pelo grupo tem acesso irrestrito a todos os projetos do grupo;
- **RFAC-2.8 -** O usuário responsável por um projeto:
    - **RFAC-2.8.1 -** pode definir a permissão de leitura ou leitura/escrita para um usuário convidado;
    - **RFAC-2.8.2 -** pode remover a permissão de um usuário convidado;
    - **RFAC-2.8.3 -** pode transferir um projeto para outro usuário com concordância do usuário destino;
    - **RFAC-2.8.4 -** pode excluir o projeto;
    - **RFAC-2.8.5 -** pode gerenciar um projeto, como nome do projeto e serviços que o projeto vai usar;
    - **RFAC-2.8.6 -** vai poder escolher uma ou uma sequência de regras de transformações a ser aplicados nos modelos dos projetos.
- **RFAC-2.9 -** Número de membros em um grupo pode ser alterado pelo administrador.

3.3. REQUISITOS FUNCIONAIS DE INTERFACE

- **RFIT-3.1 –** Suporte para outros idiomas;
- **RFIT-3.2 –** Suporte para tema no modo claro e escuro;
- **RFIT-3.2 –** Aplicação responsiva em Desktop e Tablet;
- **RFIT-3.3 –** Painel de gestão com estatistas para uso do administrador.

4. REQUISITOS NÃO FUNCIONAIS

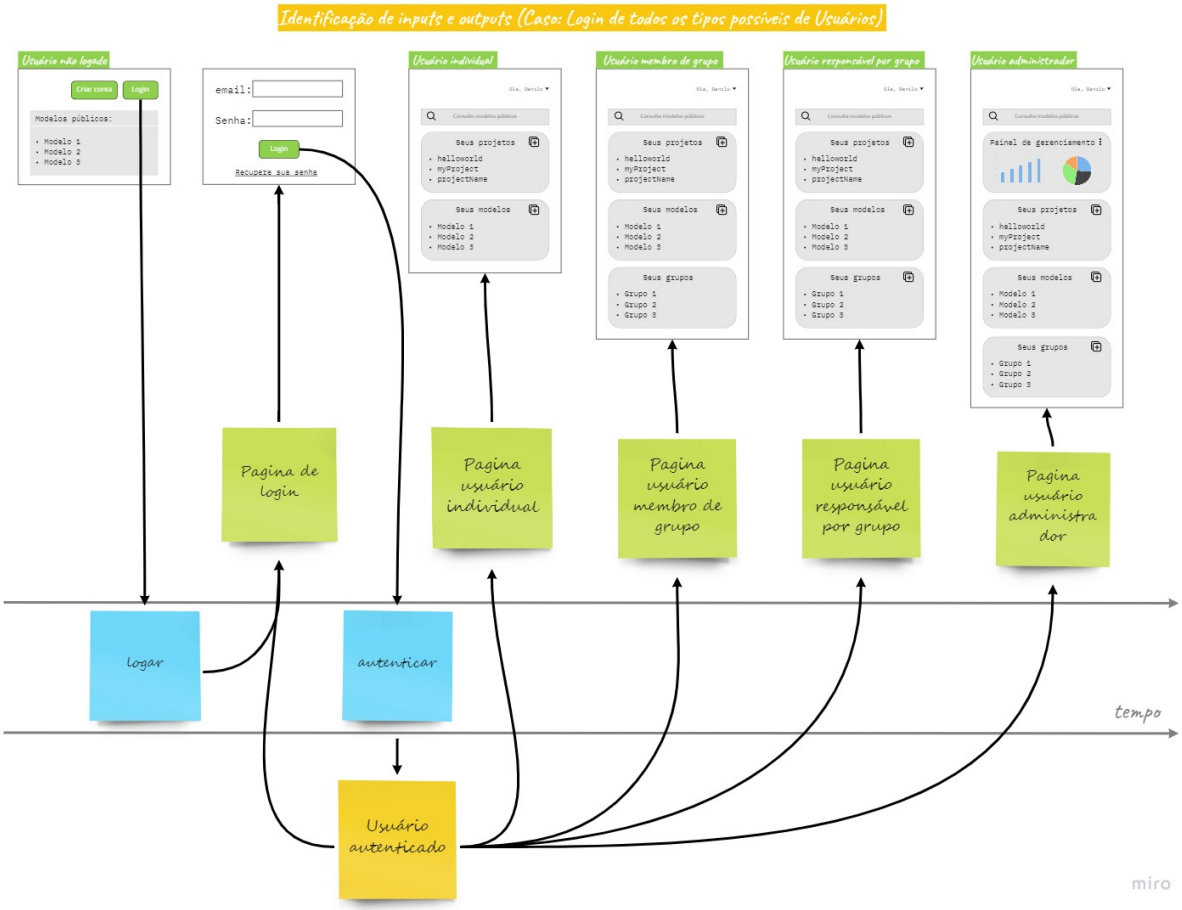
4.1. REQUISITOS NÃO FUNCIONAIS DE USO

- **RNFUS-1.1 -** Manter a privacidade entre os dados dos usuários;
- **RNFUS-1.2 -** Funções devem ser utilizáveis através do site e através de uma API Rest;
- **RNFUS-1.3 -** A disponibilização de novos serviços deve ser via interface de administração do sistema;
- **RNFUS-1.4 -** Limite de modelos no armazenamento de usuário pode ser imposto por um administrador;
- **RNFUS-1.5 -** O usuário responsável pelo grupo pode aplicar permissões a membros, grupo de membros ou a todos os membros;
- **RNFUS-1.6 -** O sistema tem uma cota padrão por usuário, gerenciado pelo administrador de 5MB;
- **RNFUS-1.7 -** Um grupo pode ter no máximo 250 membros;
- **RNFUS-1.8 -** Um grupo pode ter no máximo de 50 subgrupos.
- **RNFUS-1.9 -** Evitar perdas na transformação de modelos quando possível;
- **RNFUS-1.10 -** Suportar três dos principais formatos de modelos;

- **RNFUS-1.11 -** Compatibilidade com as últimas versões dos browsers "Google Chrome" e "Mozilla Firefox";
- **RNFUS-1.12 -** Deve ser usado TLS.

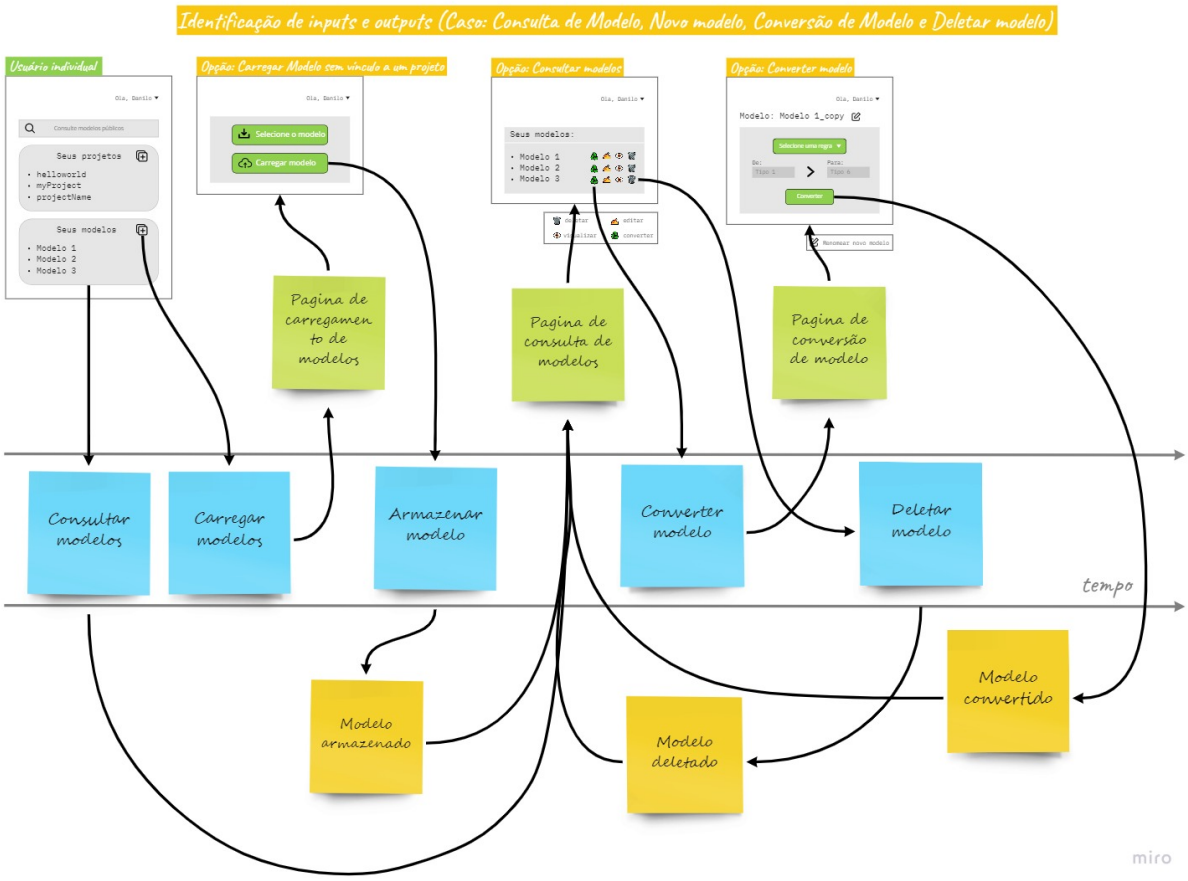
APÊNDICE B - MODELAGEM DE EVENTOS

Figura 40 – Modelagem de eventos para login de diferentes níveis de usuário



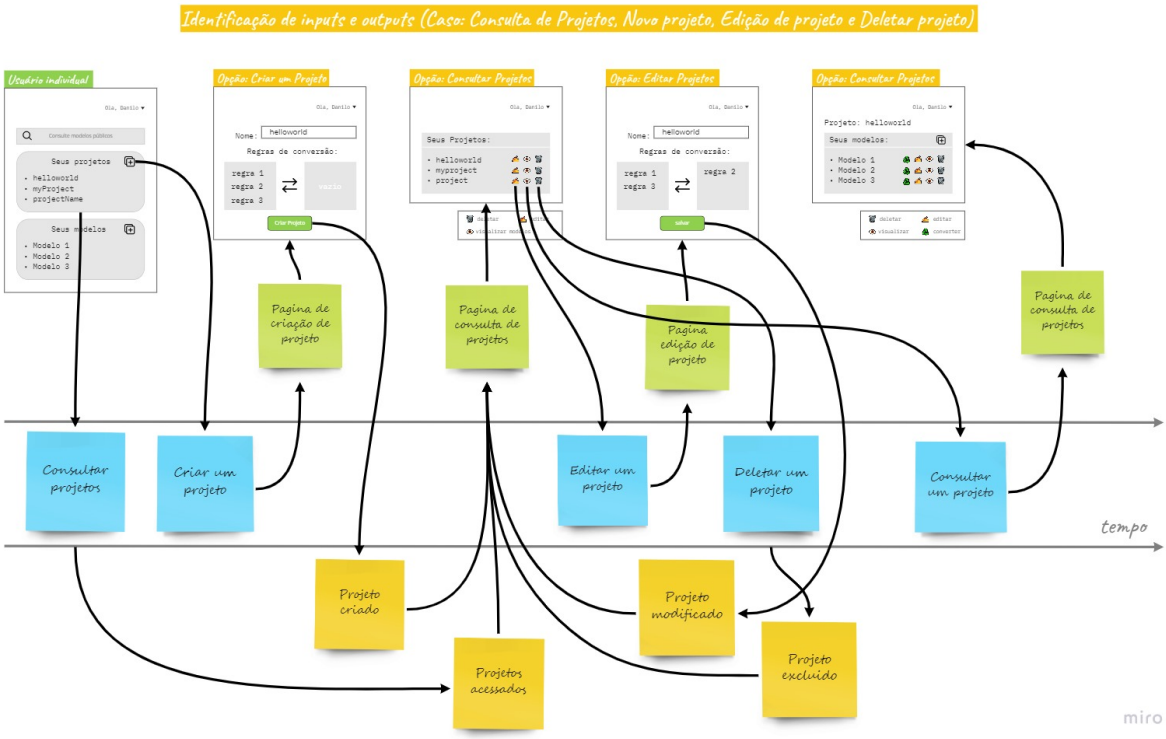
Fonte: De autoria própria

Figura 41 – Modelagem de eventos para operações básicas com um modelo



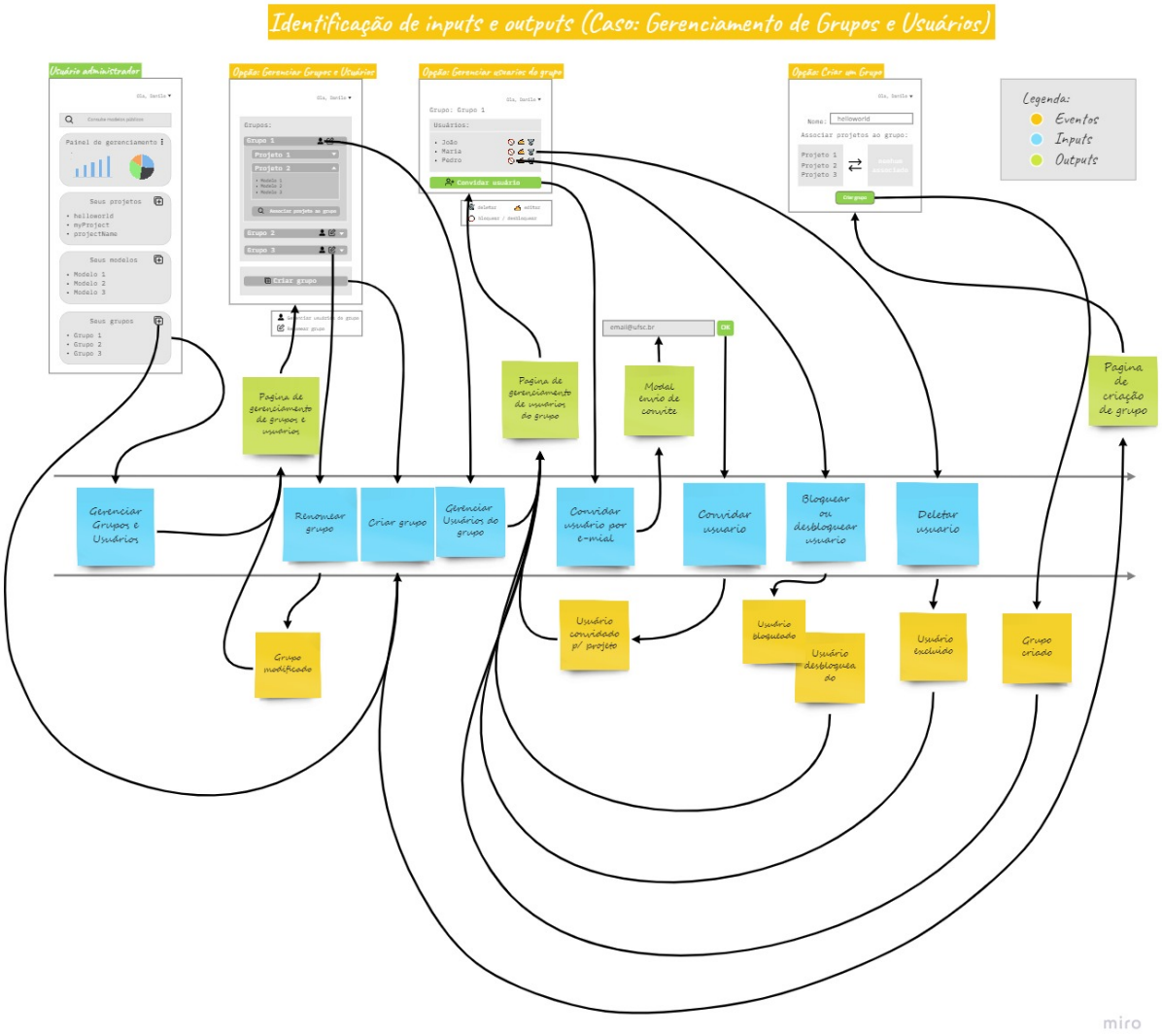
Fonte: De autoria própria

Figura 42 – Modelagem de eventos para operações básicas com um projeto



Fonte: De autoria própria

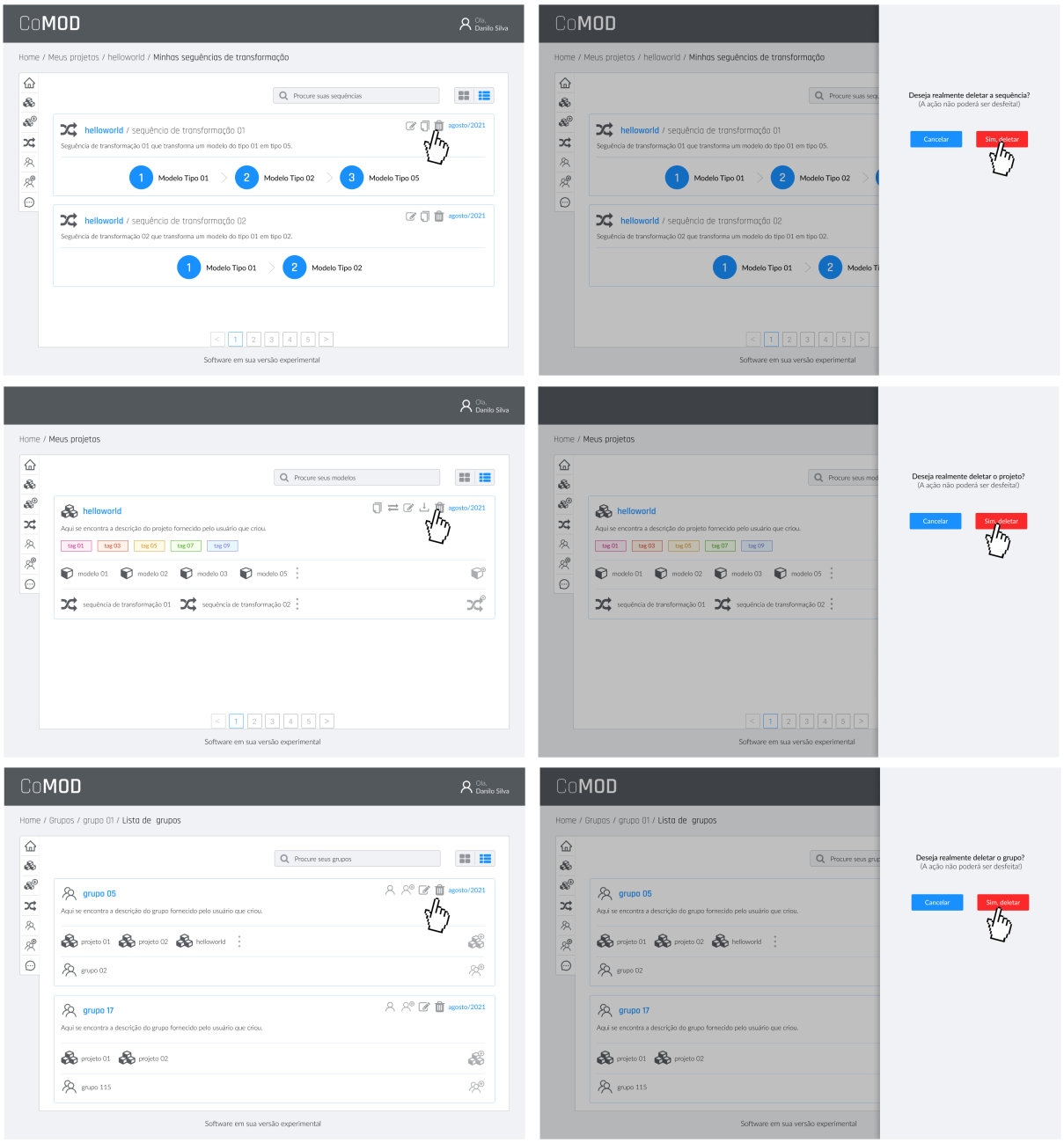
Figura 43 – Modelagem de eventos para gerenciamento de grupos e usuários



Fonte: De autoria própria

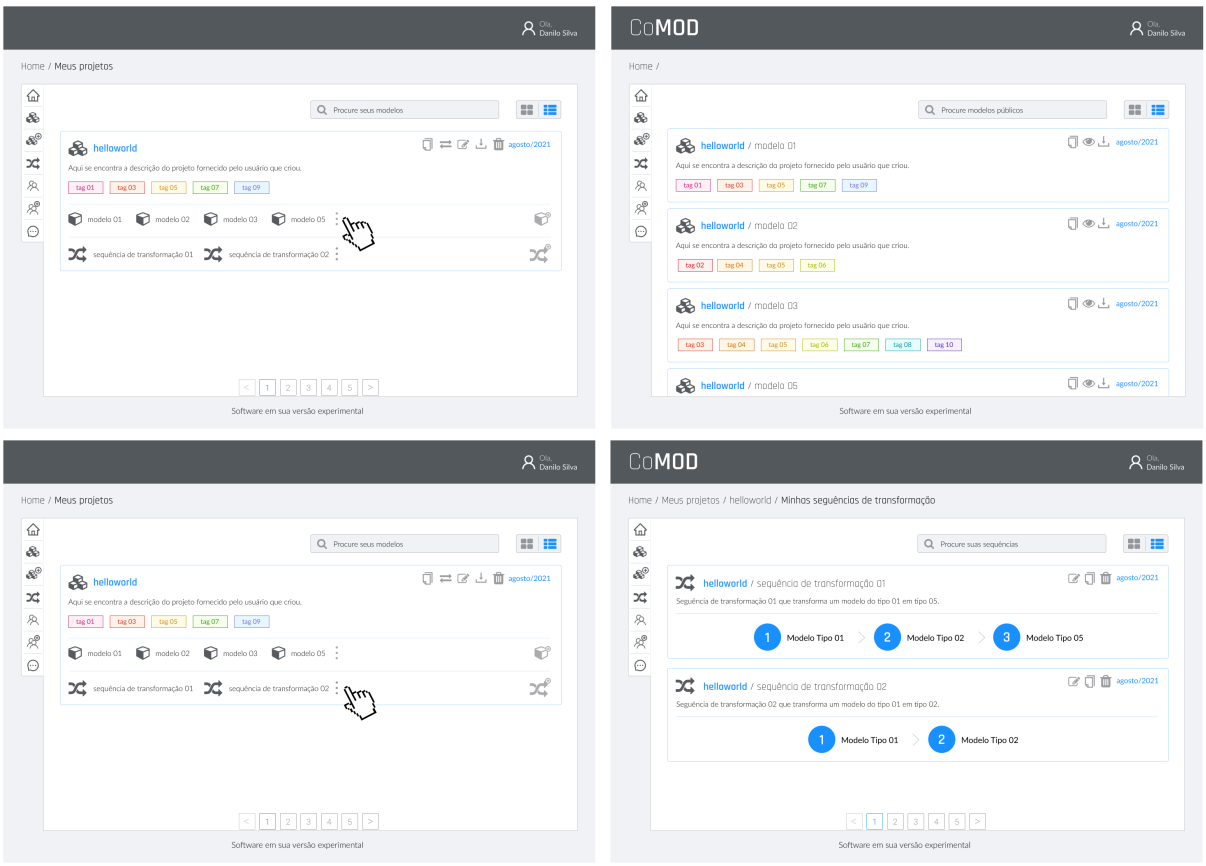
APÊNDICE C - PROTÓTIPOS E IMPLEMENTAÇÃO

Figura 44 – Protótipo para o caso de exclusão de sequência, projeto ou grupo



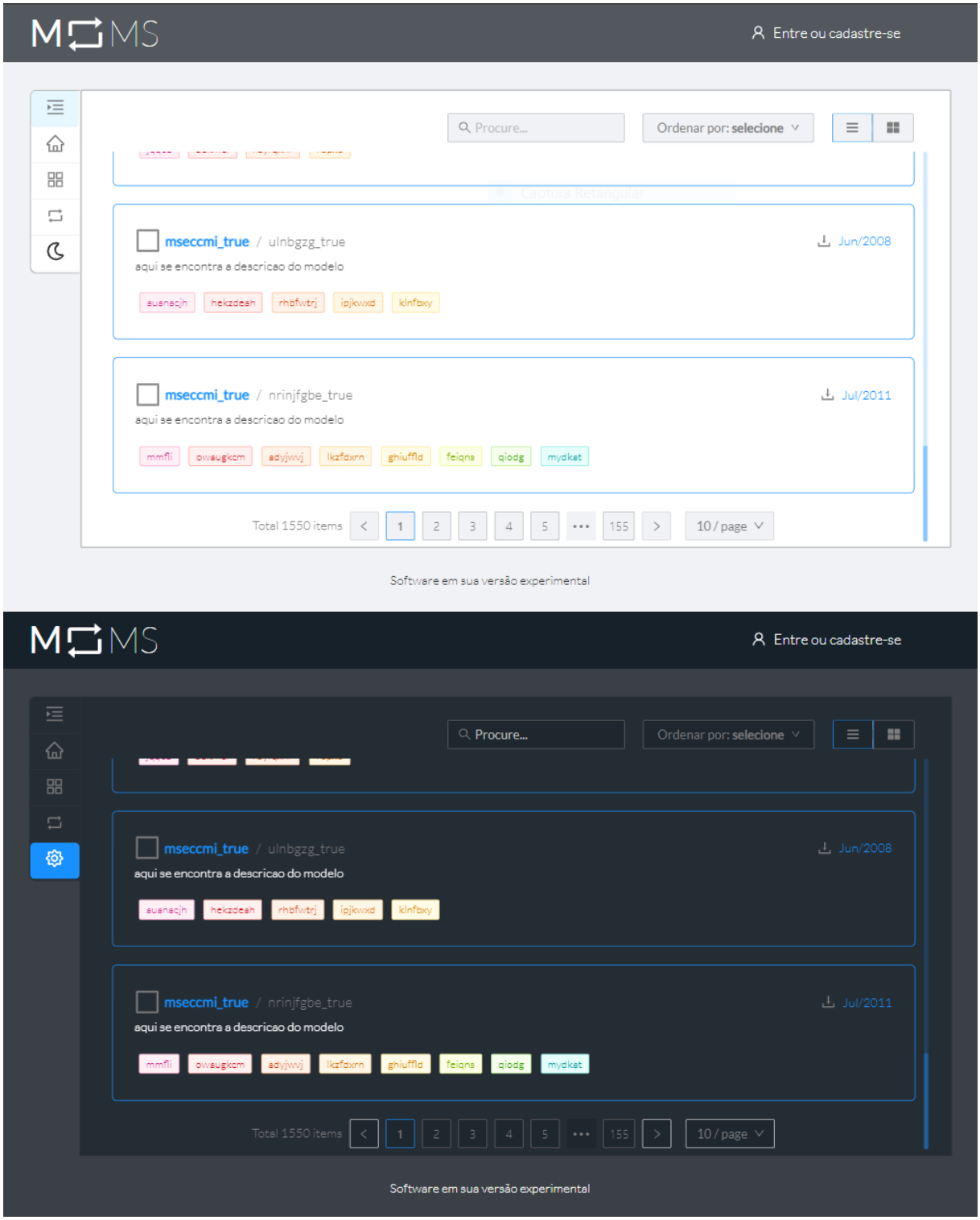
Fonte: De autoria própria

Figura 45 – Protótipo para o caso de consultar os modelos e sequências de um projeto



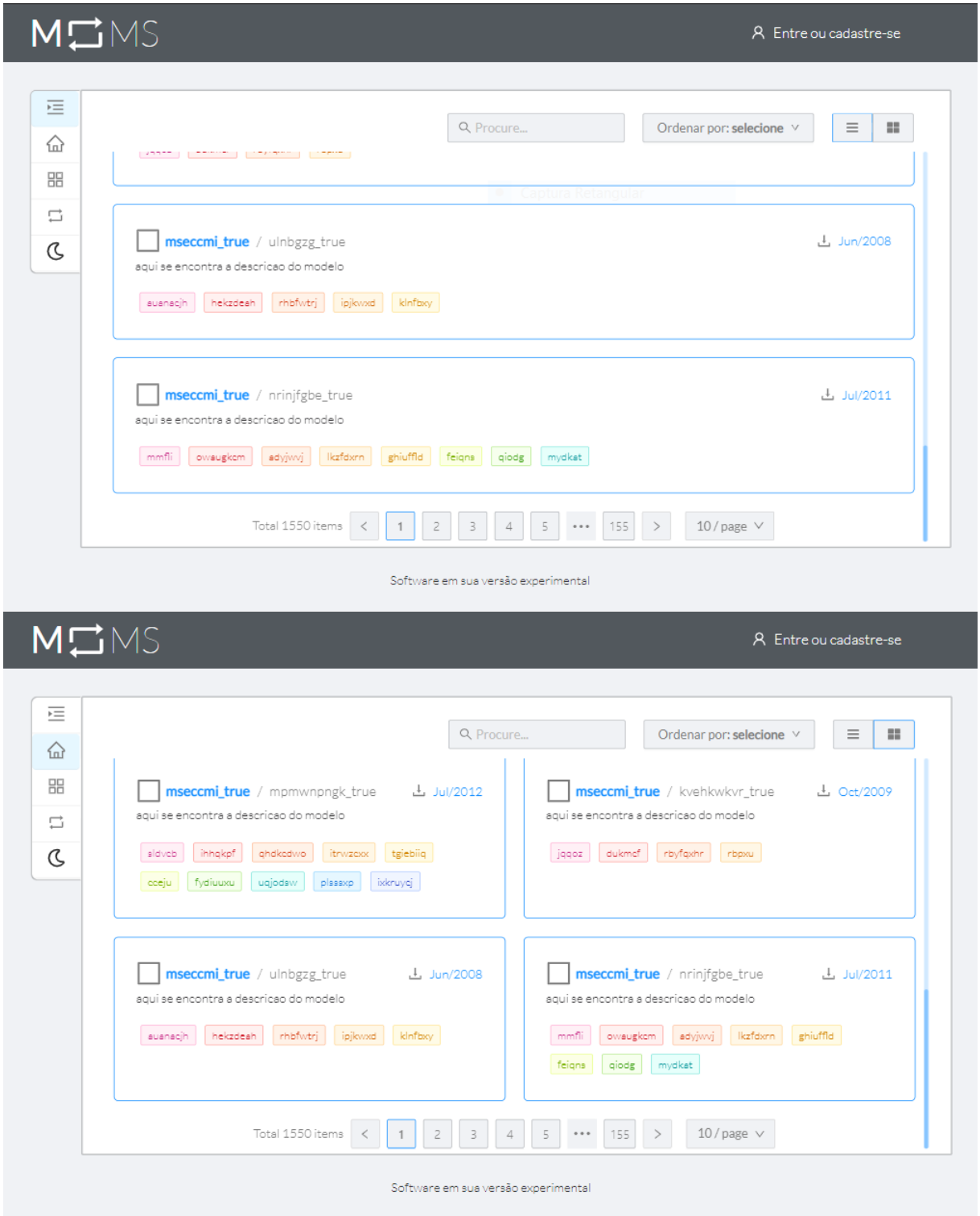
Fonte: De autoria própria

Figura 46 – Implementação para o caso de mudança de tema



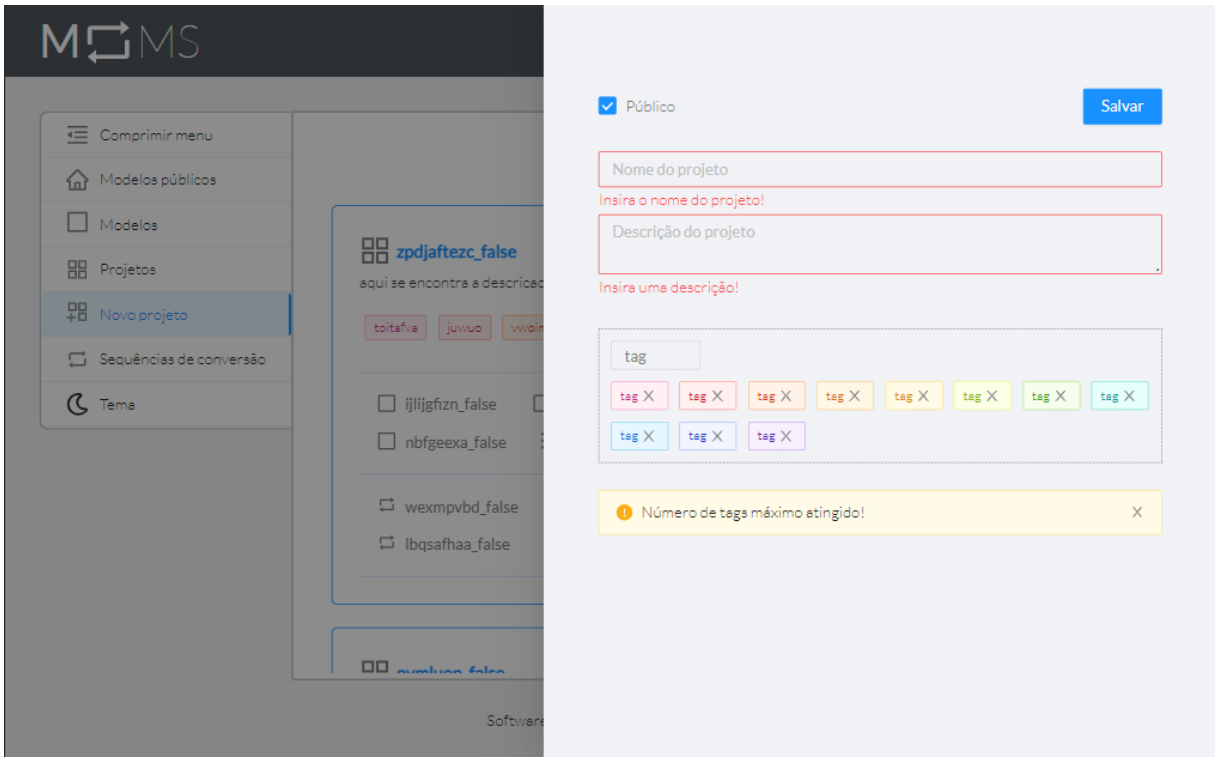
Fonte: De autoria própria

Figura 47 – Implementação para o caso de mudança de coluna simples para dupla



Fonte: De autoria própria

Figura 48 – Implementação para o caso de um novo projeto



Fonte: De autoria própria