

CENTRO UNIVERSITÁRIO FEI

GUILHERME ROCHA VIEIRA E THOMAS ANDERSON FERRARI

ENTITY EDITOR - PROJETO DE ENGENHARIA DE SOFTWARE: Site para a criação
e edição de entidades

São Bernardo do Campo

2020

SUMÁRIO

1	Introdução	4
2	Desenvolvimento do Software	5
2.1	Canvas	5
2.2	Scrum	6
2.3	Aplicação do Scrum	8
3	Planejamento do negócio	10
3.1	Levantamento de Requisitos	10
3.2	Cronograma	12
3.3	Análise de Risco	14
3.3.1	Análise da Matriz SWOT	16
4	Modelagem da Aplicação	17
4.1	Diagrama de Fluxo de Dados	17
4.1.1	DFD de nível 0	18
4.1.2	DFD de nível 1	18
4.1.3	DFD de nível 2	18
4.2	Modelo Entidade Relacionamento	21
4.3	Diagrama de Caso de Uso	22
5	Ampliação e Alteração do Software	25
5.1	Relato do teste com os clientes	25
5.2	Análise de Pontos de Funções	26
5.2.1	Fatores de Complexidade	27
5.2.2	Fatores de Ajuste	28
5.2.3	Cálculo do Ponto de Função	29
5.3	Atualização dos Cronogramas	30
5.4	Atualização dos Diagramas	32
6	Finalização	34
6.1	Mudanças na Aplicação	34
6.2	Checklist da Qualidade Ergonômica do Software	35
6.3	Teste de Caixa Branca com Usuário	37
6.3.1	Caminhos das funções para a Criação de Entidades	37
6.3.2	Caminhos das funções para a Listagem e Exclusão de Entidades	39

6.3.3	Caminhos das funções para a Atualização de Entidades	44
6.3.4	Teste com os Clientes e Resultados dos Testes	49
6.4	Experiência da Equipe Durante o Desenvolvimento	50
	REFERÊNCIAS	52

1 Introdução

No ambiente de programação, mais especificamente na aplicação do modelo de orientação à objetos, é comum a prática de transformar objetos físicos em objetos virtuais, um exemplo seria transformar as características ou atributos de uma caneta, como marca, cor e tamanho, para uma classe em uma linguagem de programação, na qual esse processo gera como resultado uma entidade, ou objeto, que agora pode ser manipulado de acordo com os critérios estabelecidos pela aplicação.

Esse processo de virtualização de objetos materiais é essencial para estabelecer uma ponte entre o mundo físico e virtual e tornar possível a interação entre ambos, como o que ocorre na transação realizada em bancos, na qual você pode consultar seu saldo virtualizado e sacar dinheiro para o meio físico, ou então utilizar esse mesmo dinheiro de forma virtual, através de cartões de crédito ou débito.

E é seguindo essa linha de raciocínio que o nosso projeto será realizado, em que receberemos a proposta de desenvolver um site para a criação e edição de entidades virtuais, nomeado *Entity Editor*, que será utilizado pelos integrantes do SWAMP (projeto de irrigação inteligente para diminuir o consumo de água na agricultura), pois eles precisam virtualizar as fazendas, assim como os equipamentos e conceitos subjetivos que fazem parte delas, para que essas entidades possam ser armazenadas dentro de um banco de dados e, posteriormente, possam ser manipuladas pelas aplicações que eles desenvolveram.

Atualmente, os nossos clientes utilizam o *Postman*, que é uma ferramenta bastante utilizada para testar APIs WEB com o envio de requisições HTTP, para a criação de entidades, no entanto, existe uma grande dificuldade durante esse processo de gerar uma nova entidade, pois eles precisam escrever todas as características da entidade seguindo um determinado padrão estabelecido no SWAMP, além de precisar redigir todo o texto no formato *JSON* (Notação de Objetos em Javascript ou *Javascript Object Notation*, em inglês), se tornando uma atividade bastante maçante, principalmente na criação de múltiplas entidades, ou então no relacionamento entre elas, que é um dos conceitos mais importantes nesse exercício de virtualização.

Portanto, nosso trabalho é desenvolver uma aplicação que seja capaz de fornecer as mesmas ferramentas que o *Postman*, porém de uma forma simplificada e amigável ao usuário, que automaticamente implemente os padrões utilizados no projeto SWAMP, além de facilitar o processo de edição, criação ou exclusão das entidades, e, principalmente, prover um fácil relacionamento entre elas, seguindo as regras estabelecidas pelos clientes.

2 Desenvolvimento do Software

Nesta seção será apresentado o processo de desenvolvimento de software escolhido, assim como os conceitos fundamentais envolvidos em cada etapa.

2.1 Canvas

O Canvas, ou Quadro, como também é conhecido, foi utilizado para a criação do modelo de negócios, na qual esse modelo é essencial para verificar a veracidade do nosso propósito inicial e se ele realmente poderá ser aplicado dentro de um ambiente real, para posteriormente ser utilizado como base para o plano de negócios (SEBRAE, 2013).

Logo, ele é uma ferramenta que contém 9 blocos, que são divididos entre 4 questões que precisam ser respondidas:

- a) **Vou fazer o que?**
 - Proposta de Valor: Se trata da necessidade que os clientes sentem e que precisa ser preenchida com seu produto, isto é, o valor que você pretende oferecer a eles.
- b) **Para quem** vou fazer?
 - Segmentos de clientes: São os clientes que você busca atender, seja eles de uma comunidade, localidade ou nicho específico;
 - Canais: São os meios responsáveis por levar o seu produto até o cliente;
 - Relacionamento com clientes: Que tipo de relação com o cliente, seja através de gestos ou serviços, fará com que ele escolha seu negócio no lugar de outro.
- c) **Como** vou fazer?
 - Recursos Chave: Os recursos essenciais para a realização da proposta de valor;
 - Atividades Chave: As ações primordiais para a prática do negócio.
 - Parcerias Chave: Fornecedores, parceiros ou terceiros responsáveis por te auxiliar através de um determinado serviço necessário para a aplicação do seu negócio.
- d) **Quanto?**
 - Estrutura de Custos: O que será gasto para poder realizar a nossa proposta de valor;

- Fontes de Renda: Como será feito o pagamento dos clientes pelo meu produto ou serviço.

Assim, por meio do Canvas foi possível clarear os nossos pensamentos em torno do projeto ao obter uma visão geral sobre como nosso negócio funciona e qual direção precisamos seguir a partir de agora para cumprir aquilo que foi prometido aos nossos cliente com o tempo que temos à nossa disposição. O canvas que nós produzimos sobre a nossa aplicação pode ser observado na figura 1.

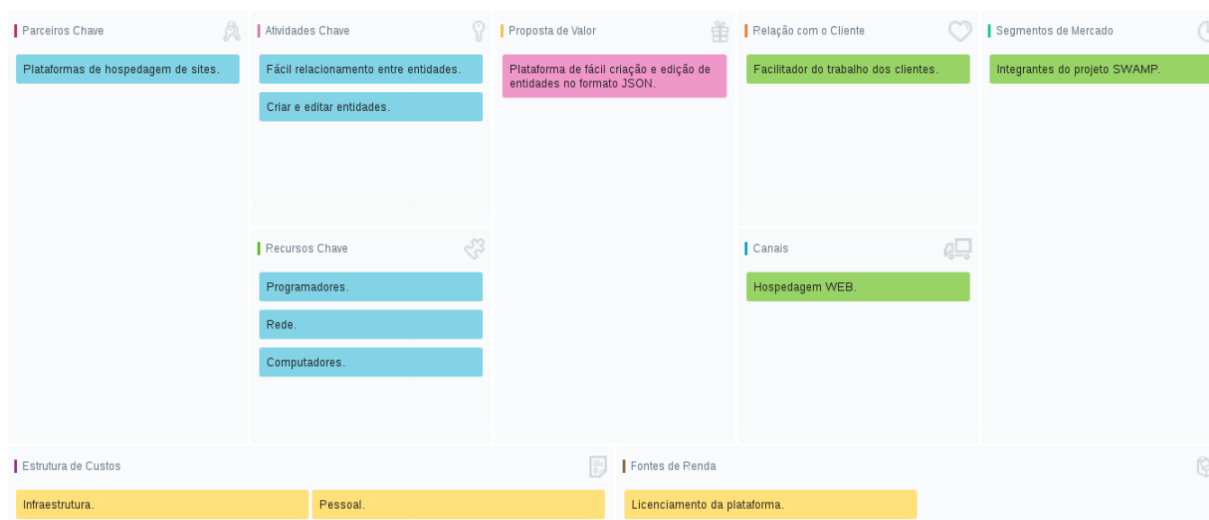


Figura 1 – Canvas.

2.2 Scrum

Após a utilização do Canvas para obter uma visão mais concreta do modelo de negócios, era de suma importância iniciar a construção do projeto, no entanto, antes de prosseguir com essa etapa, precisávamos organizar as funções que cada um teria durante o desenvolvimento do projeto, incluindo cada uma das atividades necessárias para alcançar o produto final da aplicação e como as tarefas seriam divididas entre os desenvolvedores.

Com isso em mente, adotamos a utilização do Scrum para a realização deste projeto. O scrum é um framework para gerenciamento de projetos complexos, que nasceu em meados dos anos 90 por Jeff Sutherland e Ken Schwaber. Até aquela época, os projetos de desenvolvimento de software utilizavam o método de cascata para poder cumprir seus objetivos, o 6 problema é que nesse método era necessário concluir todos os estágios do projeto, como pode ser visualizado na figura 2, para então ser lançado ao cliente, em que, normalmente, o prazo não era o

suficiente, gastava-se mais do que o previsto, e, pela falta de *feedback* dos clientes, já que o produto só era entregue ao consumidor após seu término, os produtos se mostravam insatisfatórios no final (SUTHERLAND, 2014).

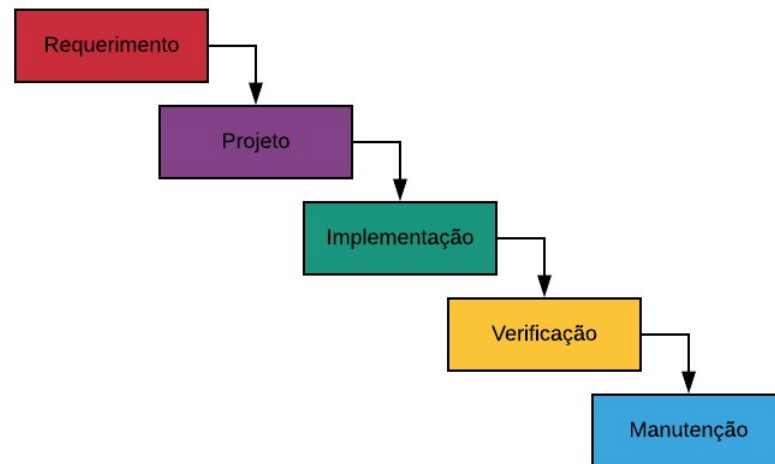


Figura 2 – Modelo em Cascata.

Dessa forma, o Scrum foi criado como uma forma de trocar esse tipo de desenvolvimento que era considerado lento e ineficiente, por um desenvolvimento mais ágil e construtivo, buscando cumprir os seguintes tópicos:

- 1) Os clientes passam a participar do projeto;
- 2) As entregas são frequentes e finalizadas, de forma que é possível entregar parte do produto para o cliente utilizar;
- 3) Existe total transparência entre a equipe e as tarefas que estão sendo efetuadas;
- 4) Regularmente são realizadas reuniões rápidas para discutir a evolução do projeto e da equipe;
- 5) Problemas durante o desenvolvimento não podem ser ignorados;
- 6) Um plano de mitigação de riscos é realizado para reduzir a ocorrência de erros durante o projeto.

Dentro do Scrum, cada integrante da equipe passa a ter um papel que está associado a uma atividade específica dentro do projeto. Primeiramente nós temos o *Project Owner* (P.O., na sigla em inglês) que tem como responsabilidade realizar a reunião de planejamento da *Sprint*, que seriam as atividades realizadas pela equipe dentro de 1 semana até 1 mês, e repassar os objetivos do projeto de forma clara e concisa ao seu pessoal, e também cuidar do *product*

backlog, que se resume na lista de prioridade necessária para criar o produto que a empresa se propôs a fazer, e que pode ser modificada à medida que se conhece mais sobre o produto e as necessidades do cliente, que será utilizado para repassar os serviços que cada membro da equipe terá de finalizar até o término da *Sprint*.

Após, nós temos o *Scrum Master*, que tem como papel a responsabilidade de ajudar a equipe e mantê-los alinhados no cumprimento do projeto. Basicamente o *Scrum Master* é um facilitador/instrutor do projeto, é ele quem impede interferências que possam prejudicar a equipe, é aquele que ajuda cada membro na realização de suas tarefas, ele faz com que todos compreendam e pratiquem os valores do Scrum e colabora para unificar a todos na busca da finalização da aplicação, incluindo o *Product Owner*.

Por fim, nós temos o *Development Team*, ou apenas *Dev. Team*, que tem como responsabilidade exercer e concluir as atividades que se propuseram a fazer, que foram definidas durante a reunião de *Sprint* realizada com o *Product Owner*, com total autonomia e liberdade para gerenciar o cumprimento de tais tarefas da melhor forma que desejarem. Durante a *Sprint* são realizadas reuniões diárias com o *Scrum Master* para responder as seguintes questões:

- a) O que foi feito ontem?
- b) O que será feito hoje?
- c) Existe algum impedimento meu ou da equipe para a realização da *Sprint*?

Na qual essas questões servem para medir o progresso da equipe em torno da *Sprint*, e impedir que quaisquer erros venham a prejudicar a finalização do projeto como um todo.

2.3 Aplicação do Scrum

Em uma primeira instância, ambos os integrantes da equipe atuavam como *Product Owner* para realizar a organização do projeto, incluindo a construção inicial do *product backlog*, e após a conclusão dessa etapa, nós decidimos como seria realizado a divisão dos papéis da equipe, na qual durante cada reunião da *Sprint* seria feito uma troca das funções que cada um seria responsável, alternando entre uma pessoa como *Dev. Team*, *Scrum Master* e *Product Owner* para apenas *Dev. Team*, isto é, a responsabilidade de administrar o projeto será revezado entre os integrantes, enquanto que ambos trabalharão como desenvolvedores.

Depois que definimos o papel de cada um, passamos a efetivamente aplicar o *framework* Scrum no nosso negócio, de acordo com o *product backlog* que foi realizado no início do pro-

jeto, na qual foi desenvolvido utilizando o Trello, que se trata de uma plataforma de gerenciamento de projeto, seguindo uma ordem de prioridade para cada item do *product backlog*, em que foi definido um conjunto de tarefas que foram divididas entre cores que, ao serem finalizadas, representam uma parte do produto que pode ser entregue ao usuário poder utilizar e testar, por conta disso, foram adicionados mais serviços do que o previsto para a produção do projeto, pois mesmo que não sejam concluídos, ainda assim teremos no final um projeto completo para o cliente utilizar, pois os itens mais importantes foram selecionados para serem executados, e somente no caso da produção dos itens restantes serem viáveis é que eles serão cumpridos.

Assim, será essencial que os integrantes da equipe realizem reuniões diárias para verificar o progresso do negócio, além de conferir quaisquer erros ou impedimentos que prejudiquem o nosso progresso e, principalmente, parar em alguns momentos para refletir sobre aquilo que já foi feito, e se essas tarefas concluídas podem ser melhoradas ou não.

3 Planejamento do negócio

Nesta seção será apresentado as atividades realizadas para adquirir as informações necessárias para iniciar o projeto, assim como toda a análise que foi feita para planejar o processo de desenvolvimento da aplicação.

3.1 Levantamento de Requisitos

Para entender melhor a necessidade dos nossos clientes, realizamos uma entrevista com eles para satisfazer algumas dúvidas que tínhamos em relação ao projeto que nos foi requisitado, assim como buscar responder questões que possam nos ajudar a criar uma aplicação que forneça a melhor experiência possível para eles, levando em consideração o tempo que temos para produzi-lo. Antes da entrevista, nós produzimos um questionário com 9 questões, que foi entregue com antecedência a um dos clientes para que ele estivesse preparado ao responder as questões para nós. Abaixo segue as questões que foram realizadas durante a entrevista, assim como a resposta das questões, que foram alteradas do ponto de vista do cliente para um melhor entendimento do leitor:

- a) Qual o mínimo de entidades diferentes necessárias? E qual o máximo? Resposta: No mínimo 4 entidades: sensor, zona de manejo, fazendeiro e atuador, e no máximo até 15.
- b) Qual o tipo de plataforma mais apropriada para que a aplicação seja desenvolvida? Resposta: Site, pois como tudo é feito na nuvem, fica mais fácil de nós acessarmos assim como qualquer outro que faz parte do projeto SWAMP, como também quem for realizar esse trabalho para o fazendeiro, já que não é ele que será responsável por utilizar esse tipo de sistema.
- c) Todas as entidades poderão ter relacionamentos? Resposta: Dentro do sistema do ORION é possível, porém para a lógica do serviço que precisamos, é necessário que exista limitações entre esses relacionamentos, já que, seguindo a lógica que nós utilizamos, um sensor não poderia se relacionar com um atuador por exemplo, então é essencial que essas regras sejam definidas.
- d) Qual a Regra de formatação do arquivo JSON? Resposta: Ele inicia com um id que fornece a identificação da entidade, logo ela não pode ser repetida, e ela também segue o seguinte padrão: "urn:ngsi-ld:<nome da entidade>:<nome da enti-

dade><número correspondente a sua criação>", que tem como exemplo "urn:ngsi-ld:SoilProbe:SoilProbe001", após a identificação, nós temos o *type*, que representa o tipo da entidade, e a partir daí nós temos os atributos da entidade, que são objetos que obrigatoriamente tem como atributo o campo *type*, que se refere ao tipo do valor do atributo, que pode ser do tipo *Text* ou *Number*, e o campo *value*, que se refere ao valor do atributo, e por fim nós temos o atributo de relacionamento, que é responsável por criar a relação entre as entidades, na qual ele também é um objeto, porém o nome dele sempre precisa iniciar com "ref" seguido do tipo da entidade da qual ele está se relacionando, por exemplo, se ele se relacionar a uma entidade do tipo "SoilProbe", esse campo será nomeado "refSoilProbe", e assim como os atributos normais, ele é um objeto que contém os campos *type* e *value*, porém o campo *type* obrigatoriamente precisa ser do tipo *Relationship* que é exclusivo para essa situação, e o campo *value* precisa ser o valor do id da entidade que a entidade atual se relacionará. Inicialmente, não existe a necessidade no momento da criação da entidade de adicionar os atributos dela, basta apenas os campos id e *type*.

- e) Qual a regra de relação entre as entidades? Resposta: Quando uma entidade filha se relaciona com uma entidade pai, é necessário que ambas estejam relacionadas, pois apesar de não ser um requisito do sistema, isso vai facilitar nosso trabalho quando precisarmos realizar alguma *query* no banco.
- f) O que precisa ter no formulário e como deve ser a estrutura dele? Resposta: No formulário é obrigatório que ele tenha os campos essenciais para a criação de uma nova entidades, sendo elas a identificação e o tipo, porém algumas não precisam ser apresentadas ao usuário já que elas seguem padrões que podem ser automatizados, e também é preciso ter a possibilidade de adicionar novos atributos, na qual é possível selecionar qual o tipo desse atributo e poder inserir o valor do atributo.
- g) As entidades que serão criadas têm características diferentes? E se tem, seria interessante um formulário que se adequa ao tipo de entidade? Resposta: Sim, pois mesmo que as entidades sejam da mesma categoria, os atributos que compõem ela podem ser bem diferentes, logo seria interessante que o formulário se adequasse a esse tipo de situação.
- h) As entidades seguem padrões de características, ou seja, se eu tiver uma entidade de um probe de solo, todas as entidades desse mesmo tipo sempre terão a mesma

quantidade de características? Resposta: Não, pois mesmo entidades que sejam da mesma categoria e do mesmo tipo, elas possivelmente terão atributos diferentes para cada uma delas.

- i) Em qual idioma é necessário que o projeto seja desenvolvido? Resposta: Em inglês, pois é necessário que o nome de todas as propriedades das entidades sejam nessa língua, mas no caso da interface não é necessário, porém seria interessante, pois como fazemos parte de um projeto internacional, outros que fazem parte desse projeto poderiam vir a utilizar.

3.2 Cronograma

O cronograma da aplicação foi dividido em 5 etapas, de forma que cada etapa tivesse uma duração de 2 semanas e que cada uma das tarefas que compõem as etapas fossem o suficiente para prover um *feedback* para o usuário, isto é, a cada etapa que é realizada, uma parte funcional do software poderá ser disponibilizada para o usuário utilizar e nos informar sobre sua experiência em relação ao que foi feito até o momento, de forma a seguir com a ideia proposta do SCRUM, na qual cada etapa representa uma *sprint*, e no final de cada *sprint* ter em mão uma parte do produto para o cliente utilizar. Segue abaixo nas figuras 3, 4, 5, 6 e 7 o cronograma de cada uma das etapas.

Cronograma Etapa 1







 Tarefas	 Data de Entrega
 <u>Design do Formulário</u>	Mar 10, 2020 → Mar 24, 2020
 <u>Converter os dados do formulário em um objeto no padrão especificado</u>	Mar 10, 2020 → Mar 24, 2020
 <u>Conectar a aplicação ao ORION</u>	Mar 10, 2020 → Mar 24, 2020
 <u>Processo de Desenvolvimento de Software - Documentação</u>	Mar 10, 2020 → Mar 24, 2020

Figura 3 – Cronograma da Etapa 1.

Cronograma Etapa 2







 Tarefas	 Data de Entrega
 Adicionar no Formulário a opção de adicionar relacionamentos entre as entidades	Mar 24, 2020 → Abr 07, 2020
 Mandar os dados em JSON gerados para o ORION	Mar 24, 2020 → Abr 07, 2020
 Pegar os dados em JSON do ORION	Mar 24, 2020 → Abr 07, 2020
 Planejamento do negócio - Documentação	Mar 24, 2020 → Abr 07, 2020

Figura 4 – Cronograma da Etapa 2.

Cronograma Etapa 3








 Tarefas	 Data de Entrega
 Atualizar as entidades que foram relacionadas com a entidade que foi criada	Abr 07, 2020 → Abr 21, 2020
 Carregar os dados requisitados do ORION para serem editados no formulário	Abr 07, 2020 → Abr 21, 2020
 Efetivamente editar o formulário e atualizar os dados no ORION, incluindo seus relacionamentos	Abr 07, 2020 → Abr 21, 2020
 Teste intermediário com o cliente	Abr 07, 2020 → Abr 21, 2020
 Modelagem da aplicação - Documentação	Abr 07, 2020 → Abr 21, 2020

Figura 5 – Cronograma da Etapa 3.

Cronograma Etapa 4








 Tarefas	 Data de Entrega
 <u>Adicionar a ação de excluir entidades</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Ao excluir uma entidade, suas relação serão automaticamente excluídas</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Adicionar a possibilidade de criar múltiplas entidades no formulário de criação</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Adicionar a API do google earth para a localidade das entidades</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Alteração dos diagramas e cronograma</u>	Abr 21, 2020 → Mai 05, 2020

Figura 6 – Cronograma da Etapa 4.

Cronograma Etapa 5







 Tarefas	 Data de Entrega
 <u>Teste final com os clientes</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Acabamento da aplicação</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Experiência sobre o processo de produção do software - Documentação</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Atualização da documentação</u>	Mai 05, 2020 → Mai 19, 2020

Figura 7 – Cronograma da Etapa 5.

3.3 Análise de Risco

A análise de risco é uma das etapas essenciais durante a realização de um projeto, para poder reduzir os erros e eventuais ameaças que venham a impedir que o projeto seja devidamente concluído. Para esse tipo de análise, foi utilizado a análise SWOT, que é uma técnica de planejamento para identificar as forças, fraquezas, oportunidades e ameaças que uma empresa,

equipe, ou até mesmo indivíduo, pode ter, o que a torna bastante útil, já que nem sempre é fácil perceber essas qualidades e problemas.

Para a realização da análise SWOT, primeiro é construído uma matriz SWOT, na qual ela está dividida em quatro partes, tendo as duas primeiras partes como fatores internos, que podem ser transformadas pelo indivíduo, enquanto que as outras duas são fatores externos, na qual o indivíduo não tem qualquer poder sobre elas, como segue abaixo:

- a) *Strengths* ou Forças: São as qualidades principais que destacam o indivíduo em questão.
- b) *Weaknesses* ou Fraquezas: São as imperfeições que o indivíduo encontra no momento.
- c) *Opportunities* ou Oportunidades: São as ocasiões favoráveis ao redor do indivíduo que possam lhe ajudar a conquistar seu objetivo.
- d) *Threats* ou Ameaças: São as intimidações que envolve o redor do indivíduo e que podem prejudicar a conclusão do seu propósito.

Após a construção da matriz SWOT, é realizado um cruzamento entre as informações adquiridas, com o intuito de definir estratégias de atuação para amenizar ou eliminar os obstáculos que antes existiam. Na figura 8 é possível visualizar a matriz SWOT que foi produzida sobre a nossa equipe e a situação que nos encontramos no momento.

Strengths

- ▶ Experiência e habilidade com a tecnologia empregada no projeto.
- ▶ Bom relacionamento entre os membros da equipe.
- ▶ Ótima relação com os clientes.

Weaknesses

- ▶ Falta de comunicação entre os membros da equipe.

Opportunities

- ▶ Em meio a quarentena, fortalecer a comunicação entre os membros da equipe.

Threats

- ▶ Quarentena devido ao Covid-19

Figura 8 – Matriz SWOT.

3.3.1 Análise da Matriz SWOT

Por meio da observação da matriz SWOT e do cruzamento das suas informações, foi possível obter os seguintes resultados:

- a) Forças e Oportunidades: Como os membros da equipe já tem um bom relacionamento, melhorar a comunicação a distância entre eles ajudará na maturidade da equipe, assim como tornará ela mais dinâmica e adaptável a situações similares onde a comunicação foi prejudicada.
- b) Forças e Ameaças: Mesmo que a quarentena tenha dificultado o diálogo entre os integrantes da equipe e reduzido a produtividade por conta de um acontecimento inusitado e inesperado, pela boa experiência de trabalho que a equipe teve nos projetos anteriores, será mais fácil ultrapassar esses obstáculos e se adaptar a tamanha situação.
- c) Fraquezas e Oportunidades: Apesar do surgimento da dificuldade de comunicação por conta da quarentena, surgiu também a oportunidade de desenvolver as habilidades sociais dos membros da equipe durante esse momento de dificuldade, e resolver de vez esse problema para situações futuras, já que a equipe estava acostumada a interagir e desenvolver o projeto pessoalmente.
- d) Fraquezas e Ameaças: Para diminuir os problemas causados pela quarentena, e que ameaçam a competência e eficácia da equipe, serão adotados meios de comunicação já existentes, como o Whatsapp, assim como ferramentas de organização, como o Trello, e de documentação, como o overleaf e o notion, para melhorar o diálogo da equipe e também para que cada um saiba o que o outro está trabalhando a cada momento.

4 Modelagem da Aplicação

Nesta seção serão apresentados os diagramas essenciais para a modelagem da aplicação, eles são necessários para compreender melhor o escopo do projeto, e representar de forma gráfica tudo o que é importante para a execução do negócio, como as atividades, objetivos, entregas, etc.

4.1 Diagrama de Fluxo de Dados

O Diagrama de Fluxo de Dados, ou DFD, é utilizado para mapear o fluxo de informações dentro da aplicação, representando graficamente as funções ou processos que capturam, manipulam, armazenam e distribuem essas informações entre os componentes que fazem parte do sistema. A representação virtual torna o DFD uma boa ferramenta de comunicação entre o usuário e o designer do sistema, principalmente por conta dos detalhes envolvidos em cada nível do DFD que facilitam sua compreensão.

Alguns conceitos do DFD são explicados abaixo:

- a) Entidade externas: Seria todo indivíduo, hardware ou software externo que fornece ou recebe dados da aplicação. Exemplos: Usuários, sensores, APIs WEB, entre outros. São representados com um retângulo.
- b) Processos: São partes do seu código que transformam os dados em algum tipo de informação que poderá ser repassada a outros processos ou para as entidades externas. Exemplo: Pegar os dados de um usuário e realizar a autenticação dos mesmos. São representados com um círculo.
- c) Fluxo de Dados: Seria os rótulo entre os componentes do DFD, eles representam as entradas e saídas e vem ter um significado relevante. Eles são representados por flechas e não podem ser bidirecionais.
- d) Armazenamento de Dados: Basicamente é onde você guarda ou recebe dados da sua aplicação. Eles são representados por um retângulo sem as bordas da direita e esquerda.

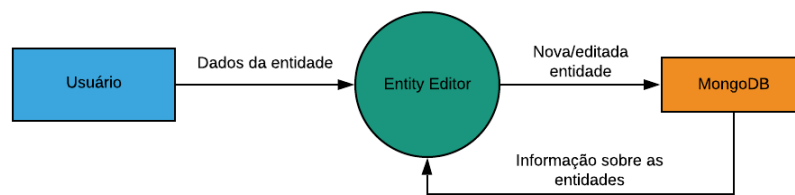


Figura 9 – DFD de nível 0.

4.1.1 DFD de nível 0

O DFD de nível 0, ou Diagrama de Contexto, é o tipo mais básico de DFD, onde todos os processos e atividades estão representados em apenas um processo, isto é, o diagrama de contexto precisa representar todo o fluxo de dados em poucos componentes, na qual é necessário definir nele todas as entidades externas, que serão herdadas pelos DFDs de nível 1 e 2, e no caso das unidades de armazenamento, pelo menos no primeiro nível, elas podem ser representadas como entidades externas. Na figura 9, temos o DFD de nível 0 que representa nossa aplicação:

4.1.2 DFD de nível 1

O DFD de nível 1 seria uma ampliação do DFD de nível 0, ao ponto de você ser capaz de explicar o funcionamento do sistema, porém, sem expor os detalhes principais de cada novo processo que foi adicionado a esse DFD. As entidades externas definidas no DFD de nível 0 são herdadas aqui, porém novas entidades não podem ser adicionadas. Na figura 10 temos o DFD de nível 1 da aplicação.

4.1.3 DFD de nível 2

Por fim, nós temos o DFD de nível 2, que tem como responsabilidade detalhar o fluxo de informação de cada processo definido no DFD de nível 1, adicionando novos componentes que venham a representar o processo anterior de forma detalhada. Aqui é uma regra essencial manter os fluxos de dados que estavam relacionados ao processo que foi expandido, como também é necessário criar um DFD de nível 2 para cada processo determinado no DFD de nível 1. Abaixo nas figuras 11, 12, 13 e 14 temos o DFD de nível 2 de cada processo expandido no DFD de nível 1.

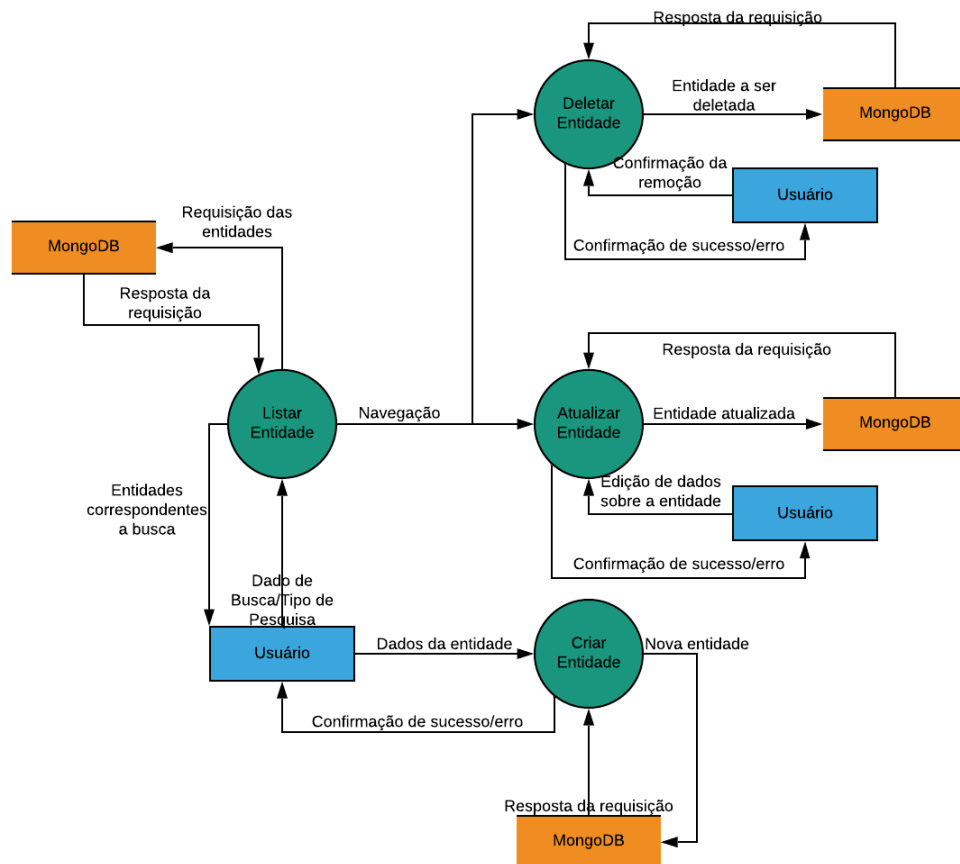


Figura 10 – DFD de nível 1.

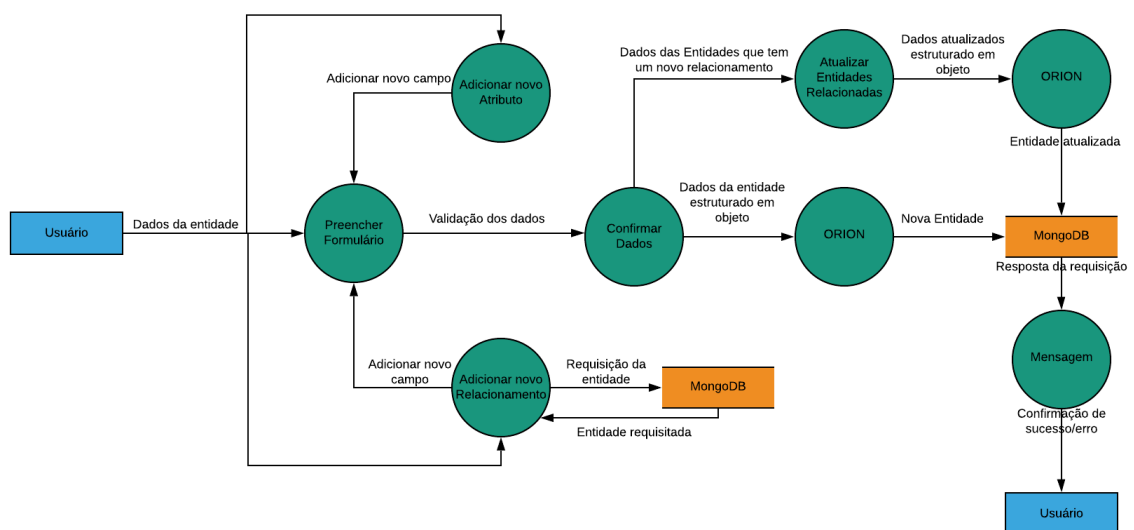


Figura 11 – DFD de nível 2 do processo Criar Entidade.

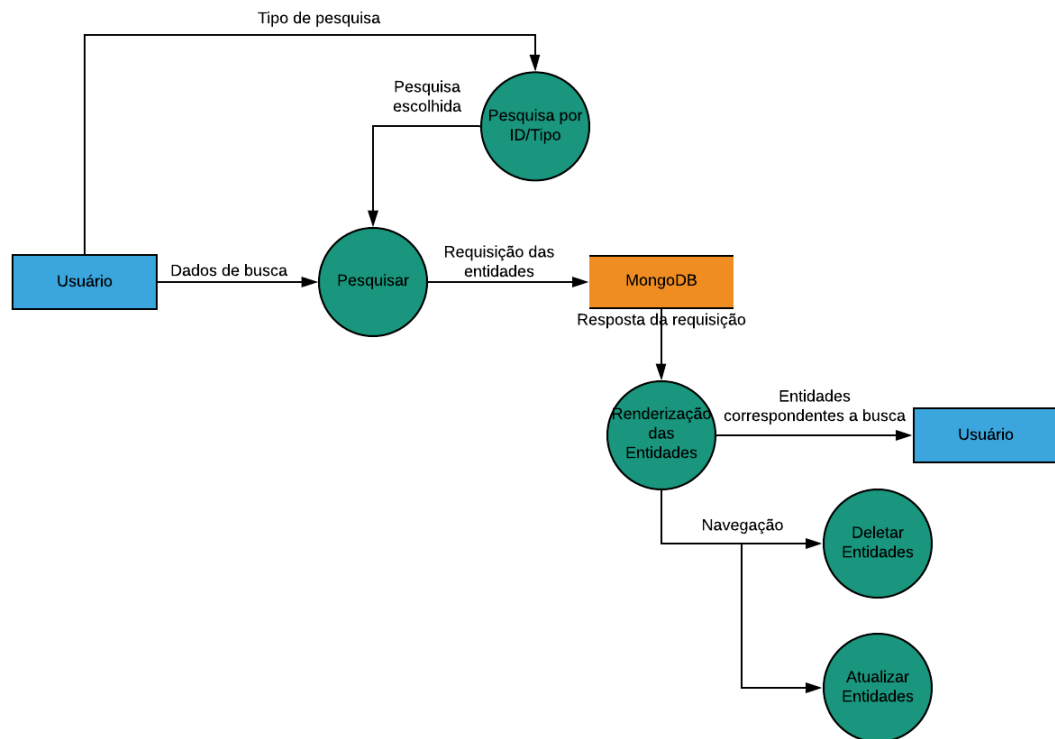


Figura 12 – DFD de nível 2 do processo Listar Entidades.

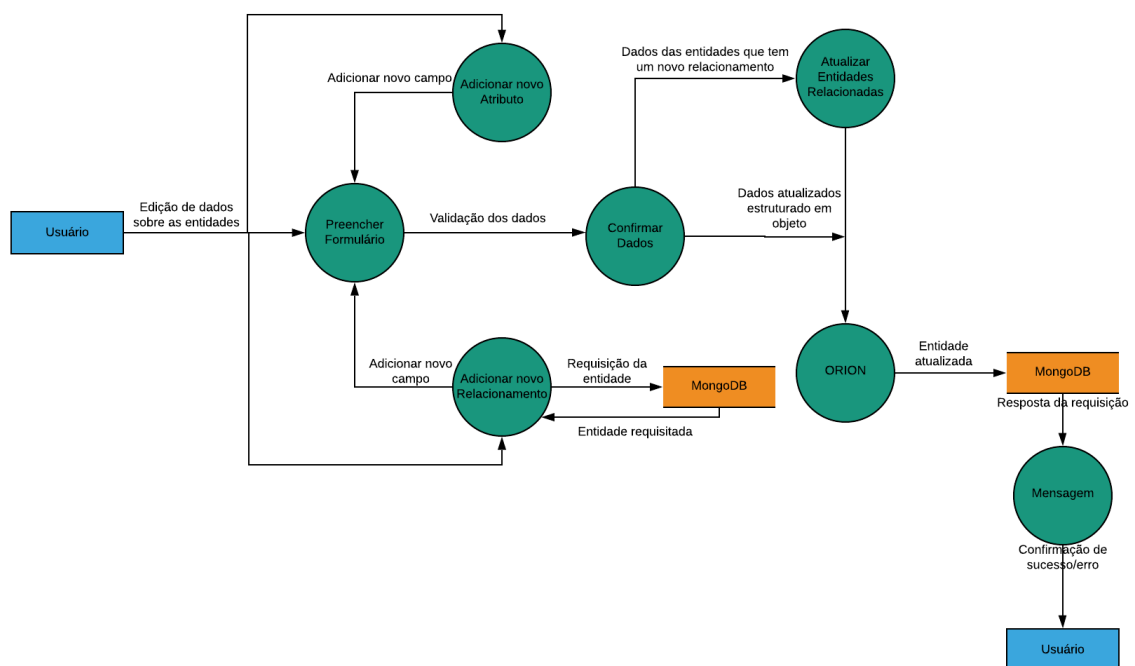


Figura 13 – DFD de nível 2 do processo Atualizar Entidades.

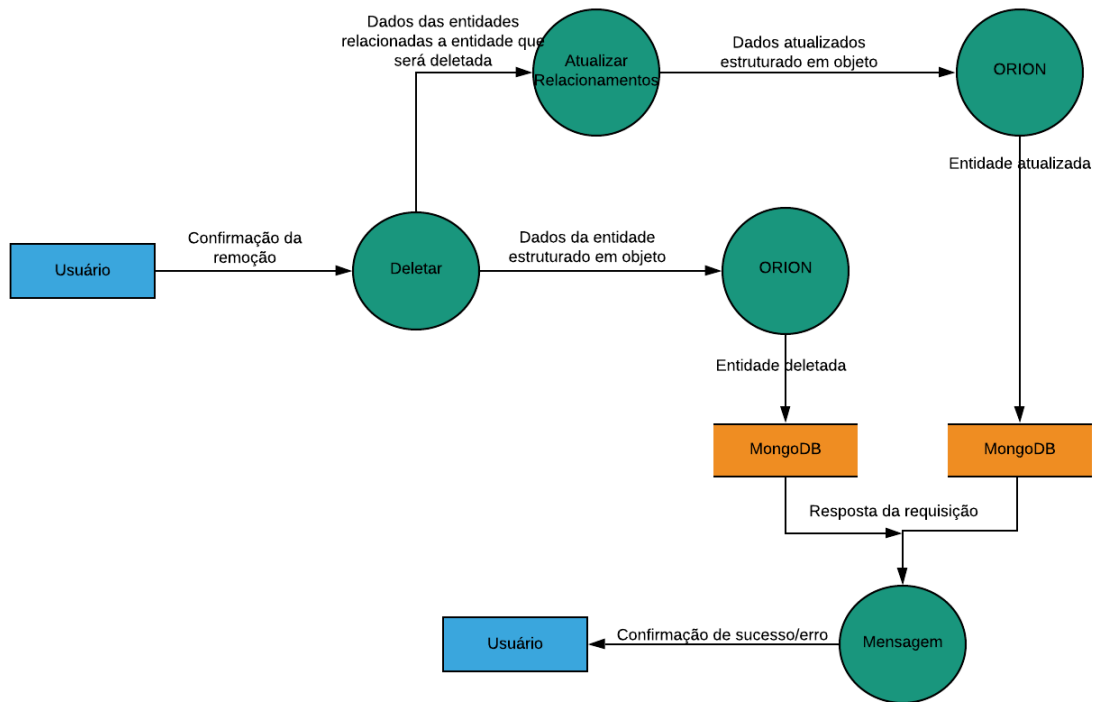


Figura 14 – DFD de nível 2 do processo Deletar Entidades.

4.2 Modelo Entidade Relacionamento

O Modelo Entidade Relacionamento, ou MER, é uma ferramenta gráfica para representar o relacionamento entre as entidades que fazem parte da sua aplicação, mais especificamente os dados que representam dados do mundo real e que serão manipulados e armazenados no seu banco de dados. O MER é útil para entender a regra de negócio, sempre visando a implementação física, para que o modelo seja refinado e possa ser executado em um ambiente realista.

Apesar do nosso projeto utilizar um banco de dados *NoSQL*, ou não relacional, ainda assim é possível demonstrar qual é o tipo de relação entre as entidades, como é mostrado na figura 15.

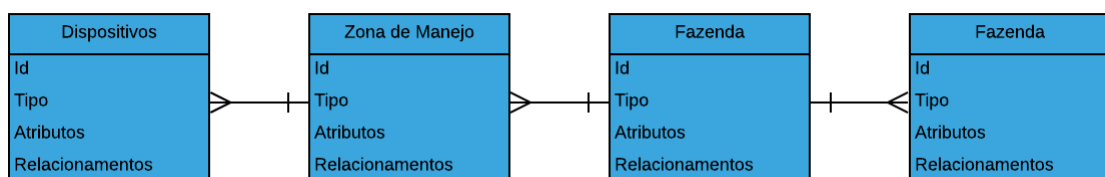


Figura 15 – Modelo Entidade Relacionamento.

No entanto, é importante ressaltar que os dados das entidades que estão sendo representadas precisam passar primeiro pelo ORION, para que depois os dados sejam efetivamente salvos no banco de dados, e como o ORION é um *context broker*, o relacionamento entre as entidades precisa funcionar de uma forma diferente nele, pois é necessário que ambas as entidades guardem o valor do Id da outra entidade que eles estão relacionados dentro do atributo Relacionamentos, o que normalmente não aconteceria no SQL, para que então o ORION possa transformar os dados e salvá-los no banco de dados, pois essa etapa é necessário para que nossos clientes possam utilizar esses dados com outras aplicações que necessitam desse formato.

Como o MER exige um aprofundamento maior sobre a aplicação e o relacionamento entre suas entidades, por meio do Diagrama Entidade Relacionamento, ou DER, que é uma etapa anterior ao MER, foi possível observar algo mais condizente com a nossa aplicação, pois por se tratar de um diagrama mais abstrato ele proporciona mais liberdade na relação entre as entidades, mas que ainda assim permite que nele resida a regra de negócio do nosso software. É possível observar o DER na figura 16.

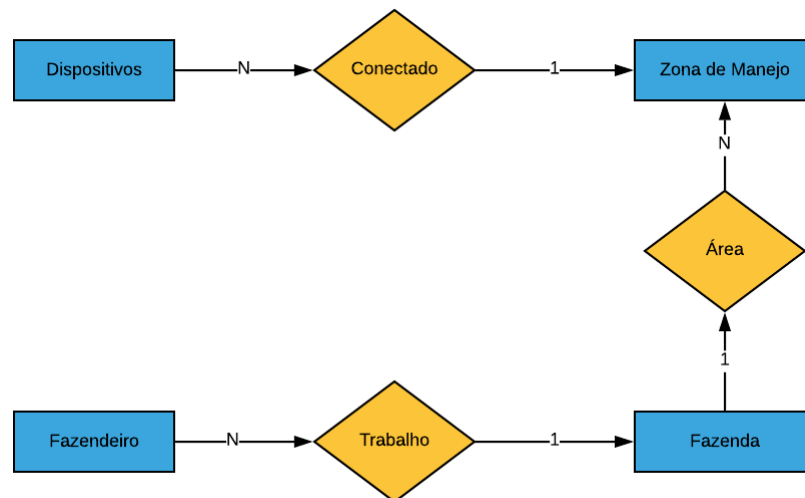


Figura 16 – Diagrama Entidade Relacionamento.

4.3 Diagrama de Caso de Uso

O Diagrama de Caso de Uso apresenta o que o sistema faz do ponto de vista do usuário, isto é, ele mostra as principais funcionalidades da aplicação e como será feita as interações entre ela e o usuário.

Os Diagramas de Casos de Uso são basicamente compostos de quatro componentes:

- a) Cenário: É a sequência de eventos que acontecem quando o usuário interage com o sistema, logo ela engloba as funcionalidades relacionadas ao evento escolhido.
- b) Ator: Seria um tipo de usuário que venha a interagir com a aplicação, como um cliente, gerente, banco ou até mesmo uma sistema, .
- c) Caso de Uso: É uma tarefa ou atividade realizada pelo ator, e que pode estar conectada a outras funcionalidades que venham, também, a interatuar com o usuário.
- d) Comunicação: Simplesmente o que liga um Ator a um Caso de Uso.

A principal utilidade desse diagrama se encontra nas fases iniciais do projeto, quando se inicia a modelagem dele, pois é possível que não exista uma ideia bem definida de como a aplicação será feita para o usuário e como ele será capaz de interagir com ela, logo, criar um Diagrama de Caso de Uso ajuda na estruturação dessas ideias. Na figura 17 é possível observar o Diagrama de Caso de Uso modelado para nossa aplicação, tendo como cenário principal o próprio site, já que é possível englobar todas as atividades dentro dele, e como ator principal nós temos apenas o cliente, que representa os nossos cliente que utilizarão o sistema.

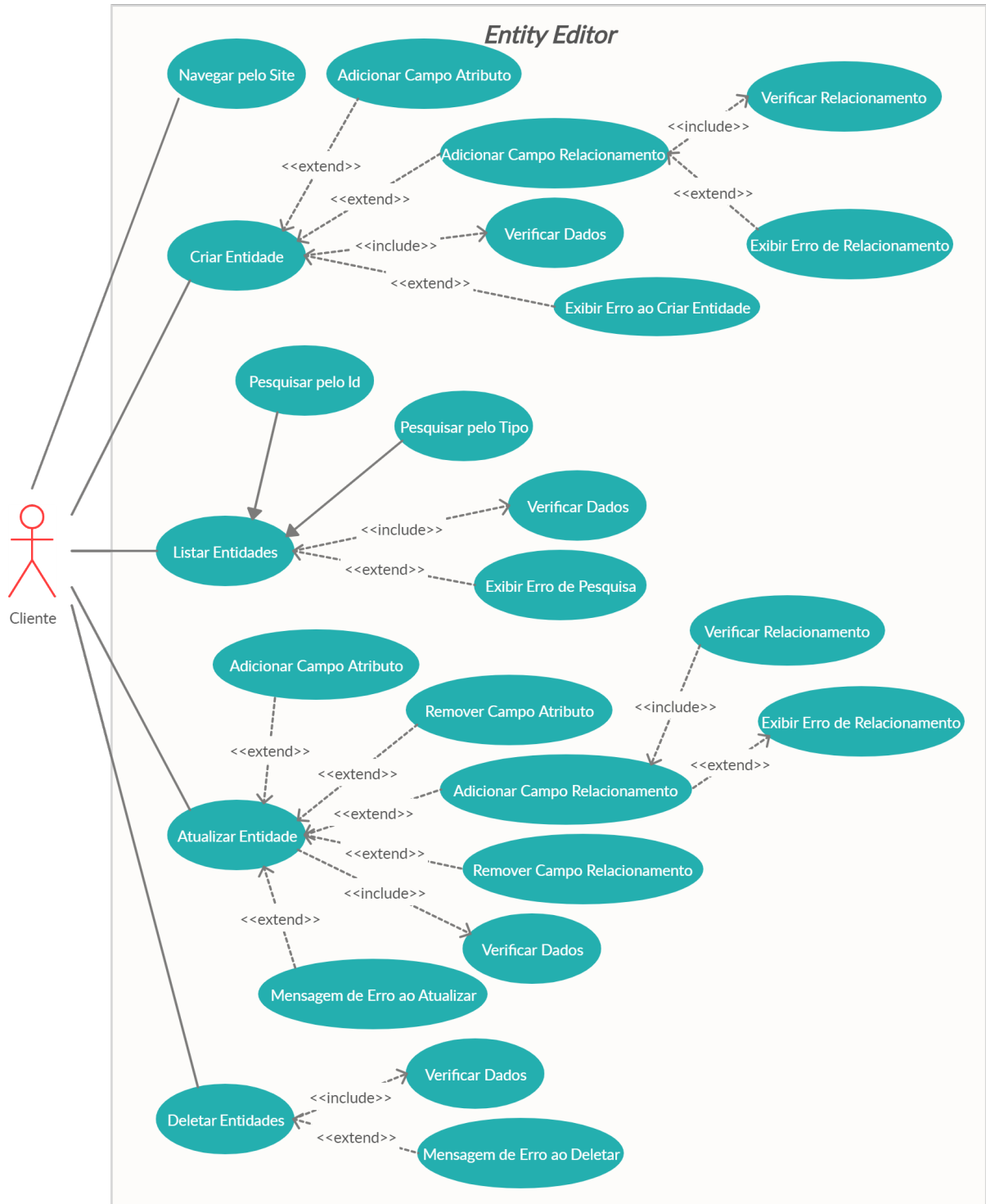


Figura 17 – Diagrama de Caso de Uso.

5 Ampliação e Alteração do Software

Nesta seção serão apresentados o primeiro teste realizado com os clientes, inclusive o *feedback* que nos foi entregue, e como esse momento que paramos para refletir um pouco sobre a nossa aplicação nos forneceu novas informações para melhorar tanto o software quanto a experiência dos usuários, e quais mudanças teremos que fazer para alcançar tal objetivo.

5.1 Relato do teste com os clientes

Para a realização do teste com os usuários, um site intermediário foi hospedado na internet através do site Netlify, para que não houvesse problemas em questão de instalações e dependências, já que os clientes poderiam rodar a aplicação na própria máquina deles, além disso, foi entregue uma versão modificada da aplicação deles, que é utilizada para salvar e armazenar o banco de dados, para impedir possíveis incômodos aos mesmos.

Dessa forma, por conta da situação que enfrentamos no momento, a realização de testes foi conduzida com 2 clientes por meio do aplicativo WhatsApp. Para esse teste intermediário nós deixamos eles utilizarem o site da forma que desejarem, pois assim a possibilidade de encontrarem erros seria bem maior, e apesar de terem encontrado erros, quando foi pedido para eles explicarem o que aconteceu e refazer os mesmos passos, nenhum erro foi encontrado no software, e os próprios clientes reconheceram que os erros foram cometidos por eles. Apesar do teste intermediário ter sido executado dessa forma, no teste final nós iremos conduzir um teste mais bem organizado e concentrado, utilizando técnicas como o teste de caixa branca, para buscarmos erros dentro do software de uma forma mais eficiente.

O *feedback* que recebemos dos clientes foi extremamente positivo, eles gostaram da experiência que a aplicação forneceu a eles e relataram como isso vai facilitar muito o trabalho deles, como também de outras pessoas que participam do projeto SWAMP. Ainda assim, foi possível perceber certos pontos que podemos melhorar na aplicação, assim como sugestões de mudanças que os clientes pediram para nós realizarmos. Segue abaixo uma lista das alterações que teremos que fazer:

- a) Apresentar mais detalhes sobre os tipos de entrada que os usuários precisam inserir nos formulários e nas entradas de pesquisa;
- b) Ser mais claro quanto aos tipos de entidades que podem se relacionar com as entidades que estão sendo criadas ou atualizadas, algo que pode ser resolvido através

de uma combo box que apresenta somente as entidades disponíveis para o relacionamento, evitando erros por parte dos usuários;

- c) Adicionar um botão para listar todas as entidades já criadas pelos usuários, pois existe a possibilidade deles esquecerem quais já foram criadas;
- d) Adicionar um botão na listagem das entidades capaz de gerar um grafo com os relacionamentos entre as entidades;
- e) Modificar a forma que os relacionamentos das entidades são armazenados na própria entidade, de forma que exista apenas 1 atributo referente ao relacionamento de um tipo de entidade, na qual apenas esse atributo irá conter todos os ids das entidades que ele pode se relacionar.

Antes da realização dessas alterações, nós fizemos o cálculo de ponto de função, assim como a atualização do cronograma e dos diagramas, para nos situarmos melhor sobre o que podemos fazer até a entrega final do projeto e quais alterações e modificações são essenciais na aplicação, para que elas sejam priorizadas o máximo possível diante do tempo que ainda nos resta para desenvolvê-las.

5.2 Análise de Pontos de Funções

A análise de pontos de funções é extremamente essencial no início de um projeto, quando os requisitos da aplicação já foram definidos, para definir prazo, equipe e custo do projeto, sem nem mesmo ter qualquer tipo de código, interfaces ou banco de dados implementados, para que a partir desse momento seja realizado uma análise da nossa capacidade, e se o projeto poderá ser concluído em um tempo razoável, e apesar de não estarmos no início do projeto, estamos passando por um processo de mudanças que exigem um prazo, portanto precisamos saber se nossa equipe é capaz de concluir tais alterações na data que nos comprometemos para entregar o projeto final, logo a análise de pontos de função nos ajudará a ter uma visão mais clara sobre nossa competência e capacidade para a realização das tarefas restantes.

Durante o processo de análise de pontos de funções, são realizados 3 passos essenciais: Primeiro precisamos identificar e enumerar as funções da aplicação, depois que as funções foram identificadas, é necessário definir seu nível de complexidade, e por fim ajustar os pontos de função brutos ao nível de complexidade de processamento, que serão explicadas em mais detalhes nas seções abaixo.

5.2.1 Fatores de Complexidade

Os fatores de complexidade se tratam das funções que serão realizadas dentro da aplicação do ponto de vista do usuário, na qual elas são divididas nos seguintes itens:

- a) Número de entradas de usuários: Processamento das informações fornecidas pelo usuário e que buscam alterar os arquivos internos do sistema ou o próprio sistema, como a inserção de dados dentro de um formulário que serão salvos dentro de um banco de dados;
- b) Número de saídas de usuários: Se tratam das requisições de dados de um arquivo de armazenamento, no entanto, o retorno de dados precisa entregar uma informação ao cliente que não esteja contida no banco, como a totalidade de cadastros armazenados no sistema ou representação dos dados em forma gráfica;
- c) Consultas: Diferente das saídas de usuários, as consultas apenas retornam informações do banco de dados por meio de um relatório que não tenha qualquer tipo de alteração dos dados para o usuário;
- d) Arquivos: Basicamente são os arquivos internos ou externos utilizados pela aplicação, como um banco de dados ou até mesmo um arquivo de texto que tenha algum tipo de armazenamento de dados;
- e) Interfaces Externas: Se referem às interfaces geradas pela aplicação, e que serão utilizadas para o usuário interagir com o sistema e cumprir os processos especificados durante seu desenvolvimento

Após a contagem das funções (que já foi realizada pelo *feedback* dos clientes), é necessário definir o nível de complexidade de cada uma das funções, contendo os níveis Simples, Médio e Complexo, na qual cada item tem suas especificações definidas pelo Manual de Práticas de Contagem do IFPUG para colocar ou não uma determinada função em cada um dos níveis disponíveis. Logo, na figura 18 nós temos a contagem dos fatores de complexidade, assim como o peso de cada nível e a pontuação total de fatores de complexidade que resultou em 60 pontos.

Fatores de Complexidade							
Items	Simples	Qtd - Simples	Média	Qtd. - Média	Complexa	Qtd. - Complexa	SubTotal
Número de entradas de usuários	3	1	4	0	6	0	3
Número de saídas de usuários	4	3	5	0	7	0	12
Consultas	3	1	4	0	6	0	3
Arquivos	7	1	10	0	15	0	7
Interfaces Externas	5	7	7	0	10	0	35
Total dos Fatores de Complexidade	-	60	-	0	-	0	60

Figura 18 – Tabela dos Fatores de Complexidade.

5.2.2 Fatores de Ajuste

Os fatores de ajuste são usados para definir o ajuste em relação aos fatores de complexidade, para se obter no final o prazo necessário para a realização das funções que foram definidas. Dentro dos fatores de ajuste existem 14 itens que servem para definir a complexidade do seu projeto, de acordo com o grau de interferência que você indica para cada item, lembrando que esse é considerado um processo bem subjetivo, já que nem sempre é possível definir com facilidade o grau de cada item, logo, refletir um pouco sobre esse processo pode ajudar a evitar dor de cabeça mais pra frente, já que ele ajuda a demonstrar o quão fácil ou difícil será o trabalho que a sua equipe terá que desenvolver. Na figura 19 nós temos a tabela com os itens dos fatores de complexidade e o grau de interferência definido para cada um deles assim como o total de grau de interferência que resultou em 22 pontos.

Fatores de Ajuste

Aa Items	≡ Grau de Interferência
1 - Backup	0
2 - Comunicação	1
3 - Processamento Distribuído	0
4 - Desempenho	1
5 - Ambiente Operacional	0
6 - Entradas on-line	4
7 - Telas/Operações Múltiplas on-line	0
8 - Atualização/Recuperação de Arquivo	3
9 - Entrada/Saída/Consulta Complexa	0
10 - Processamento Complexo	3
11 - Código Reutilizável	4
12 - Conversão e Instalação	0
13 - Portabilidade	4
14 - Manutenibilidade	3
Total dos Fatores de Ajuste	22

Figura 19 – Tabela dos Fatores de Ajuste.

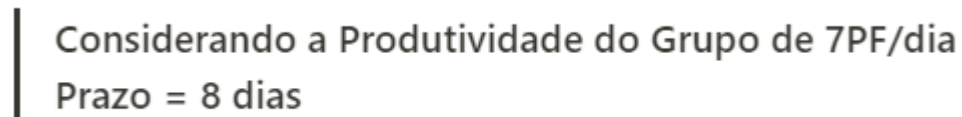
5.2.3 Cálculo do Ponto de Função

Agora que os fatores de complexidade e de ajuste foram definidos, é possível realizar o cálculo do ponto de função já ajustado, e como é possível ver na figura 20, o resultado do ponto de função foi 52,2.

$$\begin{aligned}
 PF &= \text{FatorComplexidade} * (0,65 + 0,01 * \text{FatorAjuste}) \\
 PF &= 60 * (0,65 + 0,01 * 22) \\
 PF &= 52,2
 \end{aligned}$$

Figura 20 – Cálculo do Ponto de Função.

Então, para poder definir o prazo, basta dividir o resultado obtido do cálculo dos pontos de função pelo número de pontos de função que cada membro, ou possíveis membros, da equipe consegue realizar, que no nosso caso escolhemos 7PF para o grupo todo, e como é possível observar na figura 21, nós precisamos de aproximadamente 8 dias para cumprir tais tarefas.



Considerando a Produtividade do Grupo de 7PF/dia
Prazo = 8 dias

Figura 21 – Prazo Definido para o Equipe.

A razão da escolha de 7PF/dia para o grupo todo foi pelo fato de estarmos utilizando um valor mais baixo que o normal, já que normalmente o grupo faria mais de 10PF/dia, para que não existam imprevistos durante o desenvolvimento do projeto, já que além das mudanças, precisamos focar também na realização de testes com usuário, além da escrita da documentação, portanto, um possível ritmo mais devagar na realização das alterações, em conjunto com as outras tarefas que precisam ser cumpridas, provam que nossa equipe poderá entregar o projeto com as novas mudanças na data que foi estipulada.

5.3 Atualização dos Cronogramas

Após a realização do teste intermediário com os clientes, os cronogramas foram atualizados para comportar os novos objetivos da equipe, na qual somente os cronogramas da etapa 4 e 5 foram atualizados, já que o primeiro teste com usuário foi previsto durante o cronograma da etapa 3, assim, alguns processos estabelecidos durante a etapa 4 foram retiradas, pois se tratavam de processos não essenciais para o projeto e que seriam desenvolvidos apenas na ausência de novas, como também importantes, funcionalidades e alterações que seriam identificadas após os testes com os clientes. Assim, na figura 22 e 23 nós temos os cronogramas atualizados das etapas 4 e 5 respectivamente.

Cronograma Etapa 4

+ Add a view

🔍 Search





 Tarefas	 Data de Entrega
 <u>Adicionar a ação de excluir entidades</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Ao excluir uma entidade, suas relação serão automaticamente excluídas</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Restrições para cada tipo de relacionamento entre as entidades</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Aba Home e About</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Melhora do design, responsividade e interação com usuário</u>	Abr 21, 2020 → Mai 05, 2020
 <u>Alteração dos diagramas e cronograma</u>	Abr 21, 2020 → Mai 05, 2020

Figura 22 – Cronograma Atualizado da Etapa 4.

Cronograma Etapa 5









 Tarefas	 Data de Entrega
 <u>Adição das alterações na aplicação exigidas pelo usuário</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Testes de Caixa Branca</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Teste final com os clientes</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Acabamento da aplicação</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Experiência sobre o processo de produção do software - Documentação</u>	Mai 05, 2020 → Mai 19, 2020
 <u>Atualização da documentação</u>	Mai 05, 2020 → Mai 19, 2020

Figura 23 – Cronograma Atualizado da Etapa 5.

5.4 Atualização dos Diagramas

Dentre os diagramas apresentados na etapa de modelagem, somente os diagramas de DFD nível 2 do processo de Listar Entidade e o Diagrama de Caso de Uso sofreram alterações, pois apesar de algumas alterações acarretarem na modificação do código em diversas partes do sistema, somente em relação ao processo de Listagem existirá mudanças que sejam interativas e significativas ao usuário. Assim, nas figuras 24 e 25 nós temos os diagramas que foram atualizados conforme os pedidos dos clientes e nossas observações.

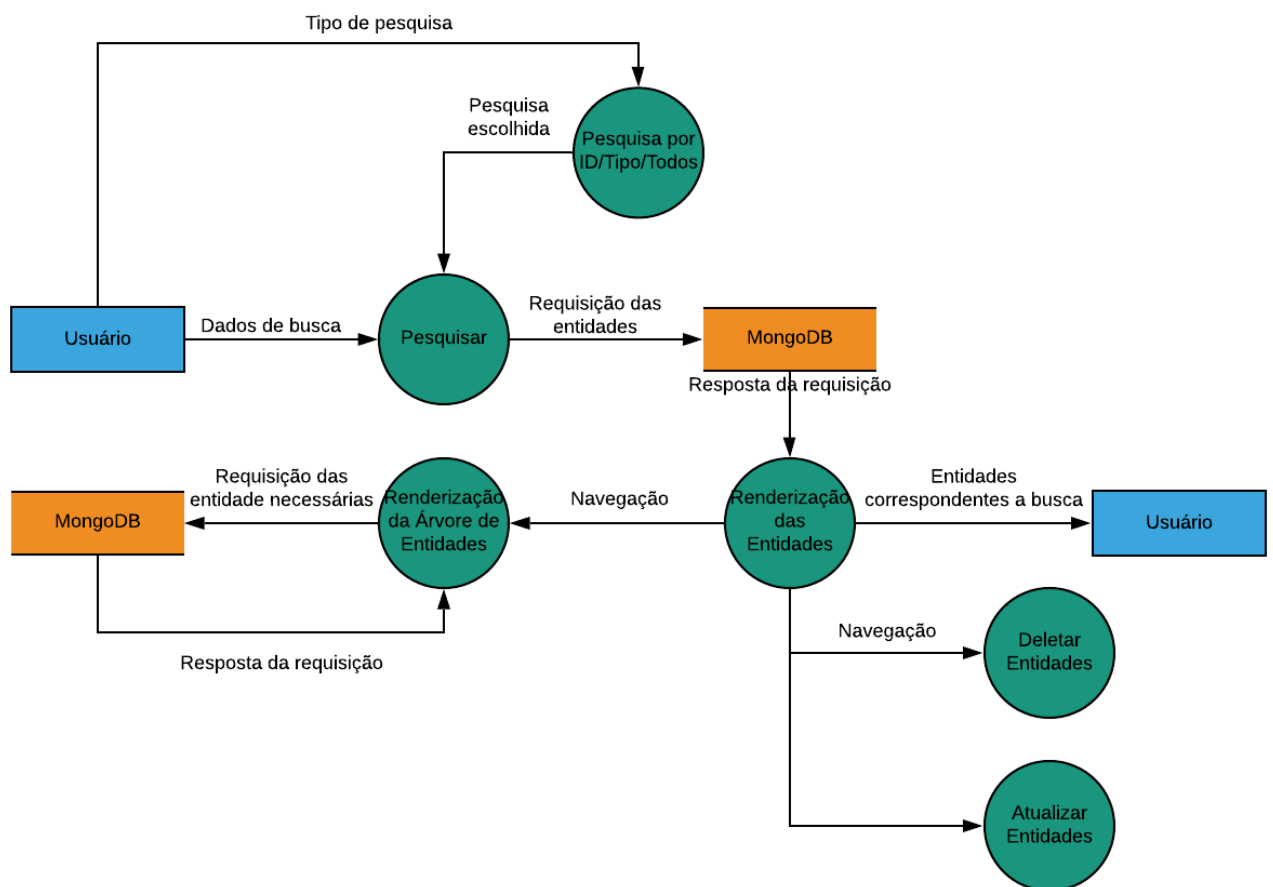


Figura 24 – DFD de nível 2 do processo Listar Entidades Atualizado.

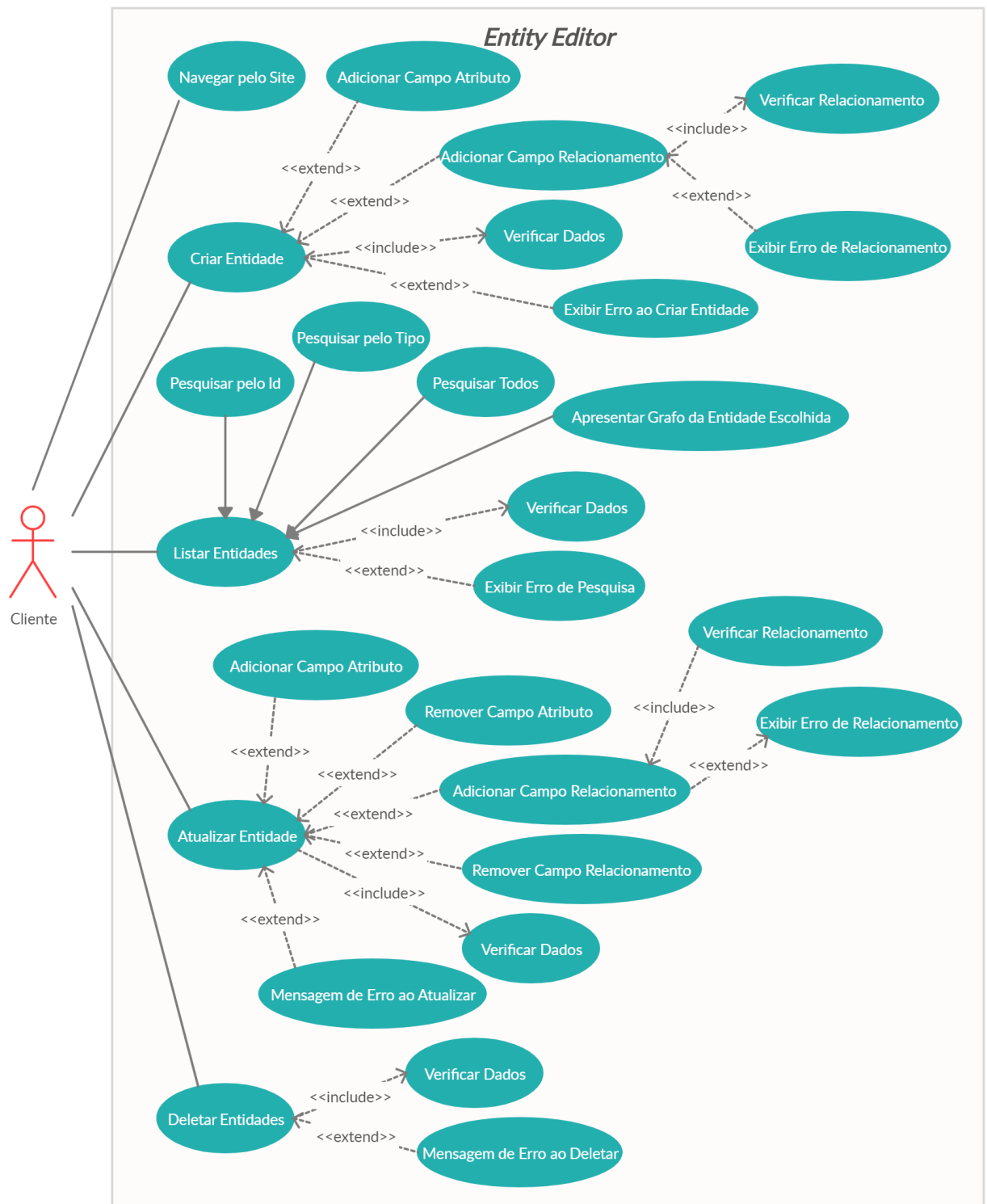


Figura 25 – Diagrama de Caso de Uso Atualizado.

6 Finalização

Nesta seção serão apresentados as mudanças realizadas no software de acordo com os pedidos e observações feitos pelos clientes, como também pela equipe do projeto. Além disso, será mostrado como foram feitos os testes de caixa branca com o usuário e quais resultados foram obtidos a partir desses testes, se foram encontradas falhas no software ou não, e de que forma nossa equipe se posicionou diante desses possíveis problemas. E por fim, será apresentado um breve texto que descreve a experiência da nossa equipe durante o desenvolvimento desse projeto.

6.1 Mudanças na Aplicação

Antes da realização do teste intermediário com o cliente, a aplicação já estava praticamente pronta para ser entregue, tirando o fato de que ainda precisávamos realizar o teste de caixa branca e solucionar alguns pequenos erros, no entanto, após a realização do teste, os clientes realizaram uma série de pedidos, na qual levamos cada um deles em consideração, separando-os em níveis de importância, onde, durante o tempo disponível, trabalharíamos para implementá-los no software, na qual segue abaixo uma lista das mudanças que foram pedidas:

- a) Modificar a forma como os relacionamentos são armazenados dentro dos objetos;
- b) Adicionar um *drop down list* que mostra apenas as entidades disponíveis que podem se relacionar com a entidade a ser criada;
- c) Especificar como deve ser digitado os relacionamentos no campo de adicionar novo relacionamento para a entidade e ser criada ou atualizada;
- d) Impedir a adição de atributos específicos para as entidades, como *id* e *type*;
- e) Adicionar o valor da entidade quando ela for atualizada, pois o campo ficava vazia quando uma entidade era selecionada para ser atualizada;
- f) Um botão para listar todas as entidades;
- g) Adicionar um botão para apresentar um grafo com os relacionamentos entre as entidades.

Entre todos os itens descritos, o único que não foi implementado foi o item g, pois não encontramos uma biblioteca que valesse a pena investir o tempo que ainda tínhamos, pois desenvolver esse item e inseri-lo na aplicação exigiria mais tempo do que o previsto no cronograma,

e como já tínhamos avisado aos clientes que esse seria um dos itens que levaríamos em consideração, podendo ou não realizá-lo, não correríamos o risco de acabar com as expectativas dos clientes em relação ao projeto, principalmente que todos os outros pedidos foram cumpridos, além das melhorias que fizemos nas já implementadas.

Diante de todas as mudanças, aquela que nos deu mais trabalho para implementar foi o item a, justamente aquele considerado o mais importante dentre as mudanças, pois grande parte da nossa aplicação trabalha em torno dos relacionamentos das entidades, pois é um conceito mais importantes para os nossos clientes, logo, inserir uma mudança no relacionamento desencadeou uma série de mudanças em grande parte do software, porém, durante o processo de desenvolvimento, percebemos que esse estilo de armazenar os relacionamentos facilitou o nosso trabalho, na qual precisávamos guardar todos os *ids* dos relacionamentos de uma entidade do mesmo tipo dentro de um vetor, no lugar de criar um atributo de relacionamento para cada uma dessas entidades, e não apenas isso, a implementação dos outros itens também se mostraram bastante relevantes, como o item b, na qual a sua implementação nos ajudou a impedir diversos erros que os usuários poderiam ocasionar, pois em vez de digitar a entidade que você quer relacionar com a entidade a ser criada, com o *drop down list* você apenas lista as entidades disponíveis para o usuário, onde mesmo que tenham mais de 100 itens, é possível digitar o nome da entidade e vai aparecer somente aquelas que condizem com o que foi digitado, algo que tornou o nosso trabalho, assim como o dos clientes, mais fácil.

Essa situação nos revelou a importância de realizar testes em com os clientes, como também manter um contato frequente com os mesmos, pois foram esses momentos que nos ajudaram a verificar erros com mais facilidade e trazer mudanças significativas para o nosso projeto, algo que não seria tão fácil apenas por meio da nossa equipe.

6.2 Checklist da Qualidade Ergonômica do Software

É importante ressaltar que, além das mudanças descritas, também foram realizadas inspeções no software para verificar a sua qualidade ergonômica, isto é, de que forma o usuário interage com a nossa aplicação e como isso impacta sua experiência na utilização do nosso software, por meio de uma série de critérios que são escolhidos pela equipe, mas que precisam estar de acordo com a entrega de valor requisitada pelo usuário, para que então sejam implementadas dentro do software. Para a criação do checklist nós utilizamos a plataforma ErgoList, da qual escolhemos 3 critérios dentre os 18 apresentados na plataforma, sendo eles *presteza*, *feedback*

e experiência do usuário, pois um dos objetivos do nosso projeto é tornar o ambiente de criação e edição de entidades o mais completo e interativo possível com os usuários, de forma que eles tenham uma experiência agradável durante a utilização do software e que facilite a utilização do mesmo, algo que não existia nas aplicações que eles utilizavam, o que tornava essa parte do trabalho deles maçante e tediosa.

Assim, como cada critério tinham diversas questões a serem respondidas, para então serem implementadas ou não no software, dependendo se já tinham sido cumpridas ou não, nós escolhemos apenas 3 questões para cada critério, sendo eles:

1) Presteza:

- Os títulos de telas, janelas e caixas de diálogo estão no alto, centrados ou justificados à esquerda?
- Caso o dado a entrar possua um formato particular, esse formato encontra-se descrito na tela?
- Listas longas apresentam indicadores de continuação, de quantidade de itens e de páginas?

2) *Feedback*:

- O sistema fornece *feedback* para todas as ações do usuário?
- O sistema fornece *feedback* imediato e contínuo das manipulações diretas?
- Qualquer mudança na situação atual de objetos de controle é apresentada visualmente de modo claro ao usuário?

3) Experiência do Usuário:

- Caso se trate de um sistema de grande público, ele oferece formas variadas de apresentar as mesmas informações aos diferentes tipos de usuário?
- O usuário pode se deslocar de uma parte da estrutura de menu para outra rapidamente?
- O sistema oferece equivalentes de teclado para a seleção e execução das opções de menu, além do dispositivo de apontamento (mouse,...)?

Como o nosso software foi desenvolvido dirigido a experiência e interação do usuário com o sistema, grande parte das questões escolhidas já estavam prontas, e foi necessário apenas alguns ajustes no código para cumprir as demais questões.

6.3 Teste de Caixa Branca com Usuário

O teste de caixa branca é um momento extremamente importante durante o desenvolvimento de um projeto, pois não existe um software perfeito que não tenha erros, logo, um dos papéis dos desenvolvedores é minimizar o máximo possível de erros que possam ser encontrados pelos usuários, principalmente aqueles que envolvem ações críticas e que podem influenciar todo o comportamento do software, ou então dos sistemas conectados ao software, e através dos testes de caixa branca, é possível encontrar erros que normalmente passariam despercebidos pelos desenvolvedores, para então solucioná-los e testar novamente o software até que não sejam encontrados mais erros.

Para o teste de caixa branca, nós criamos um grafo de fluxo, assim como uma matriz de grafo, para representar os caminhos que cada uma das nossas principais funções, pois as restantes eram apenas funções auxiliares, poderiam tomar durante sua execução, para então triarmos caminhos suficientes que pudessem passar por cada nó do grafo de fluxo, isto é, passar por cada processo da nossa função, porém, não é possível testar todos os caminhos possíveis de uma função, assim foi necessário utilizar a complexidade ciclomática para descobrir quantos caminhos linearmente dependentes existem em cada grafo, podendo ser das seguintes formas:

a) $V(G) = \text{vértices} - \text{nós} + 2$

b) $V(G) = \text{nósPredicados} + 1$

E por fim, após a criação dos caminhos, bastou nós realizarmos os casos de teste para o software para testarmos cada um dos caminhos e depois repassar as mesmas instruções para os usuários realizarem os testes. A seguir nós teremos a apresentação dos grafos de teste, matrizes de grafo e caminhos para cada uma das funções principais da nossa aplicação, correspondentes aos processos principais do software.

6.3.1 Caminhos das funções para a Criação de Entidades

Nesta sub-seção serão apresentados grafos, matrizes e caminhos para as funções que compõem os processos de criação de uma nova entidade.

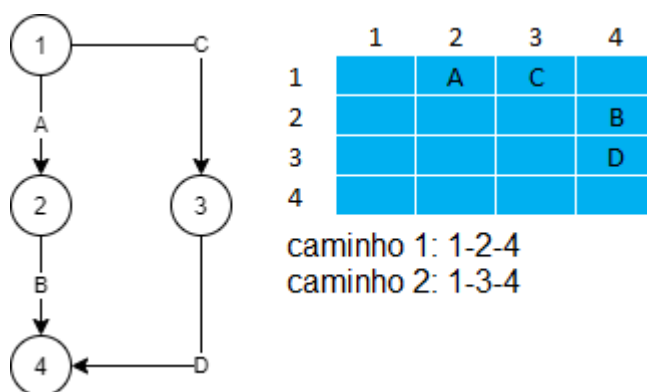


Figura 26 – Grafo, Matriz e Caminhos para a função addNewAttribute.

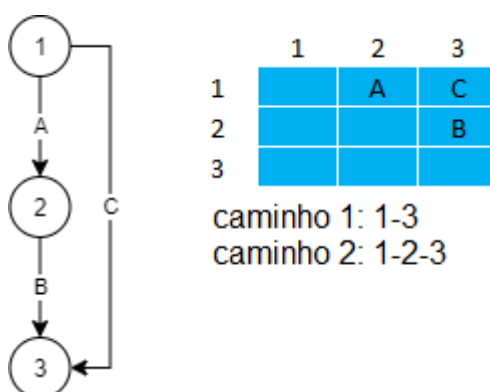


Figura 27 – Grafo, Matriz e Caminhos para a função deleteAttribute.

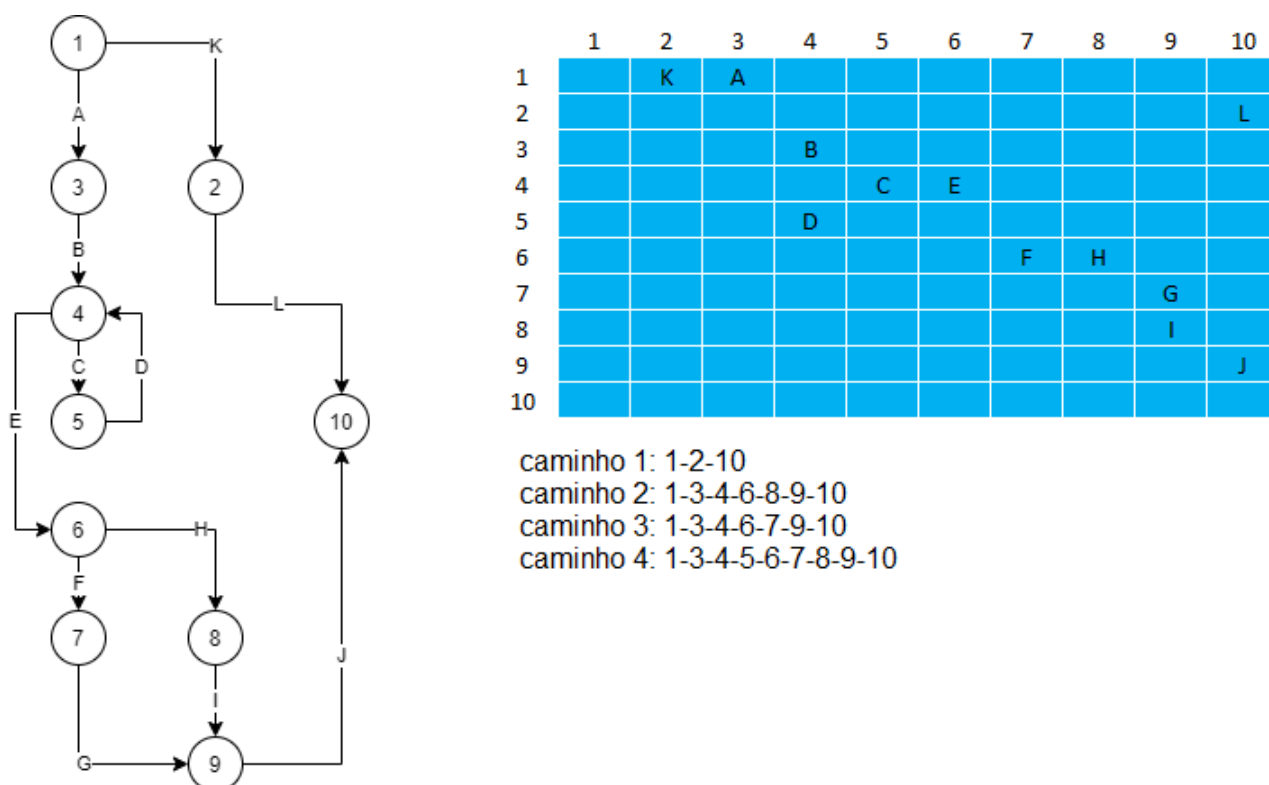


Figura 28 – Grafo, Matriz e Caminhos para a função addNewRelationship.

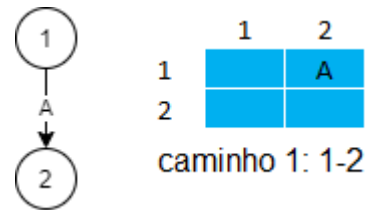


Figura 29 – Grafo, Matriz e Caminhos para a função `deleteRelationship`.

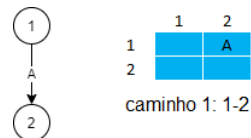


Figura 30 – Grafo, Matriz e Caminhos para a função `handleObjectChange`.

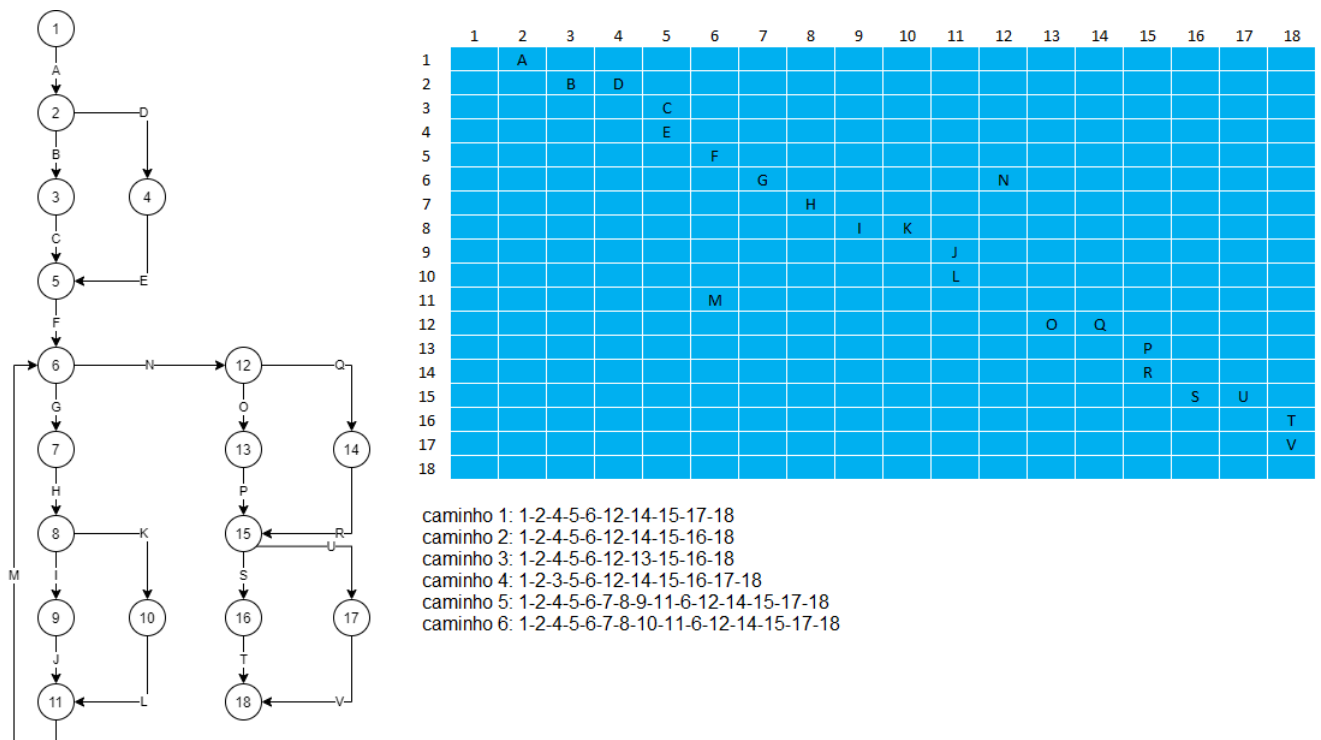


Figura 31 – Grafo, Matriz e Caminhos para a função `handleSubmit`.

6.3.2 Caminhos das funções para a Listagem e Exclusão de Entidades

Nesta sub-seção serão apresentados grafos, matrizes e caminhos para as funções que compõem os processos de listagem e exclusão de entidades.

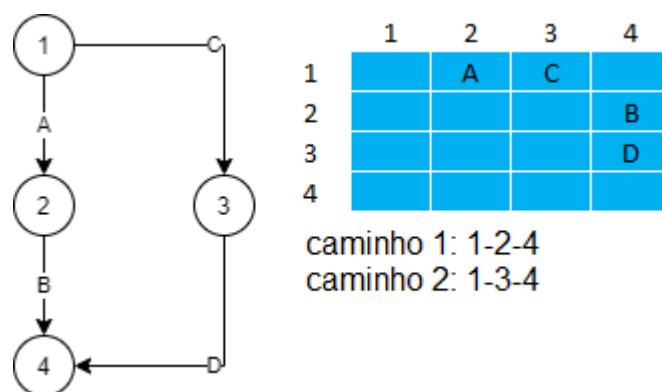


Figura 32 – Grafo, Matriz e Caminhos para a função changeSearch.

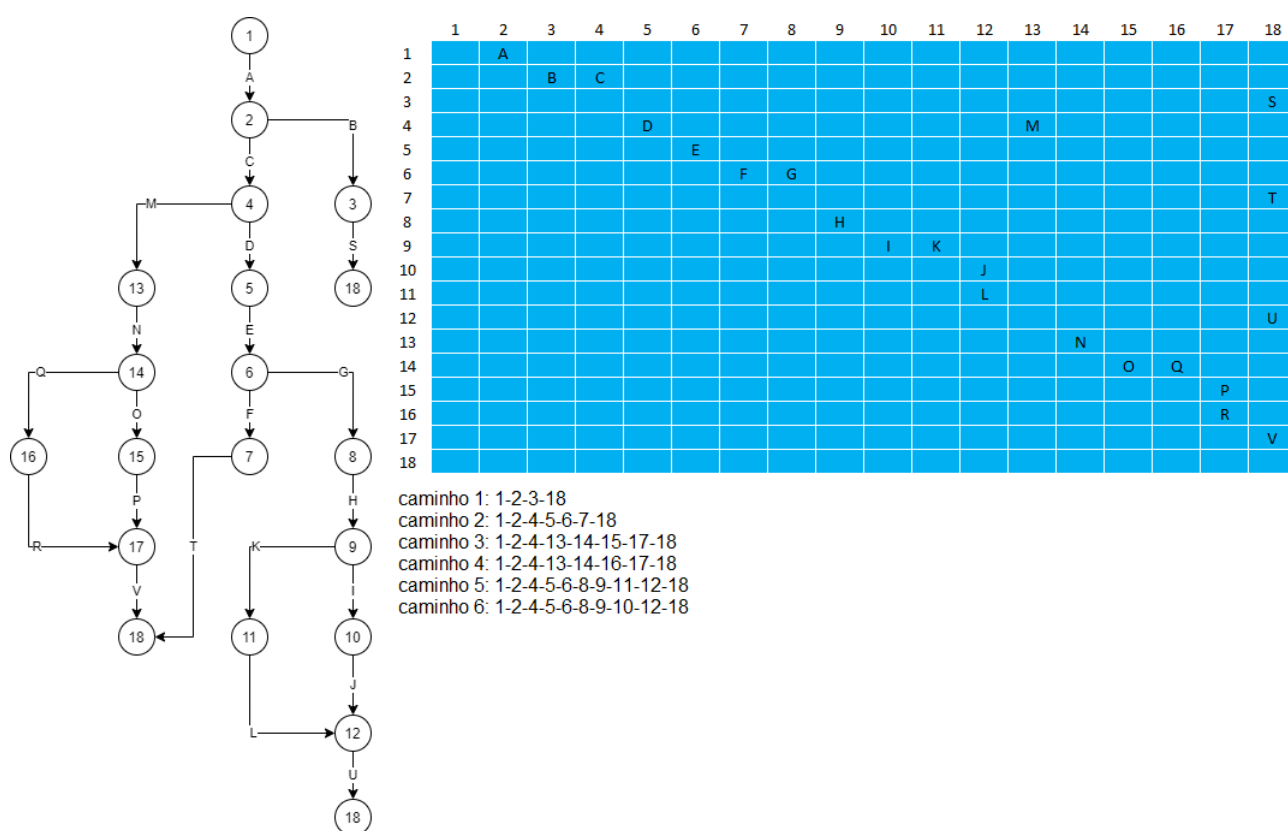


Figura 33 – Grafo, Matriz e Caminhos para a função handleSearch.

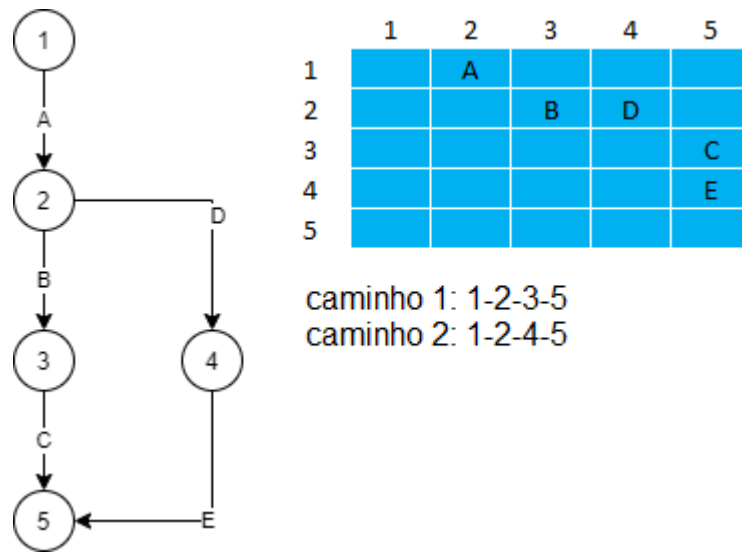


Figura 34 – Grafo, Matriz e Caminhos para a função `changePage`.

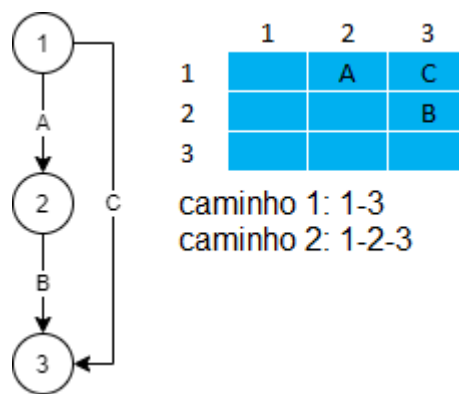


Figura 35 – Grafo, Matriz e Caminhos para a função `goBack`.

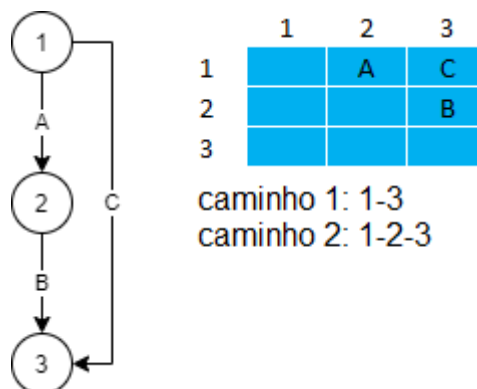


Figura 36 – Grafo, Matriz e Caminhos para a função `goForward`.

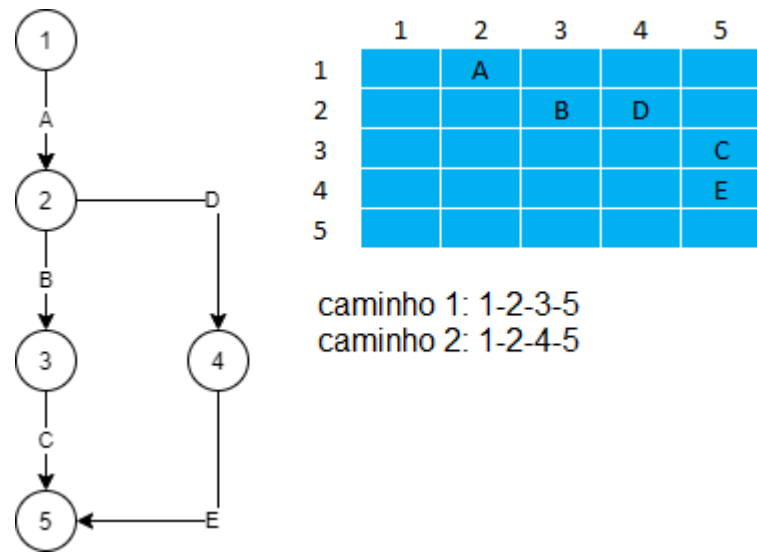


Figura 37 – Grafo, Matriz e Caminhos para a função listAll.

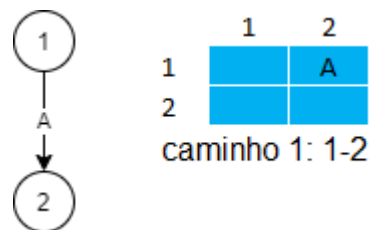


Figura 38 – Grafo, Matriz e Caminhos para a função navigateUpdateScreen.

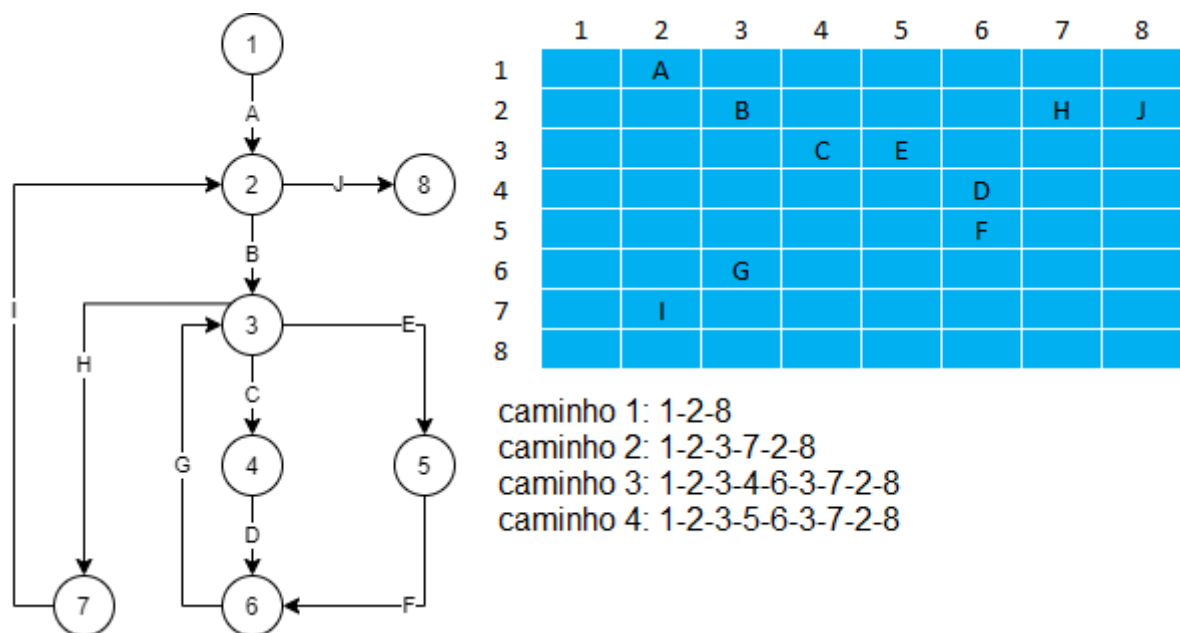


Figura 39 – Grafo, Matriz e Caminhos para a função searchDeletedRelationships.

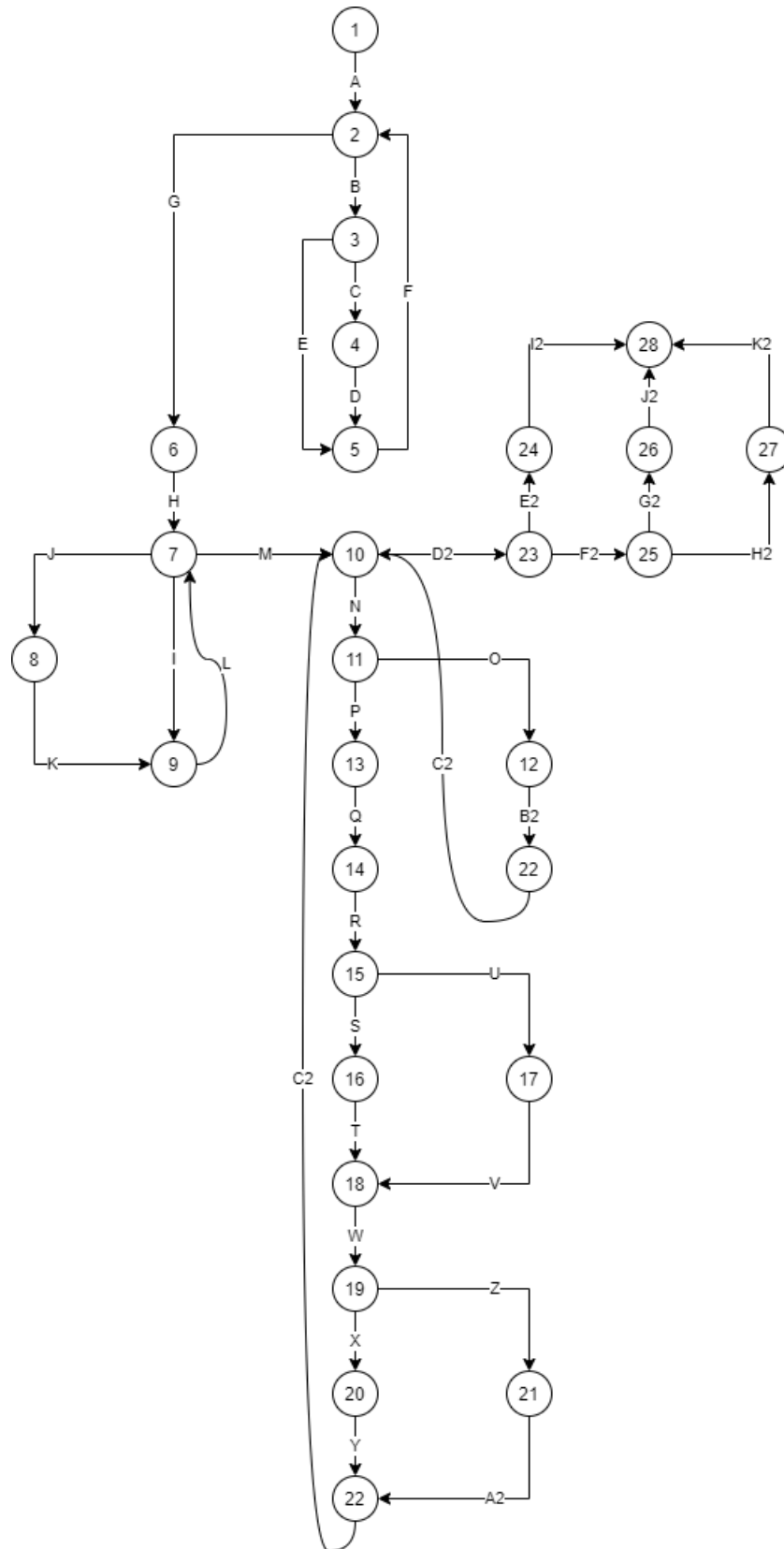


Figura 40 – Grafo para a função handleDelete.

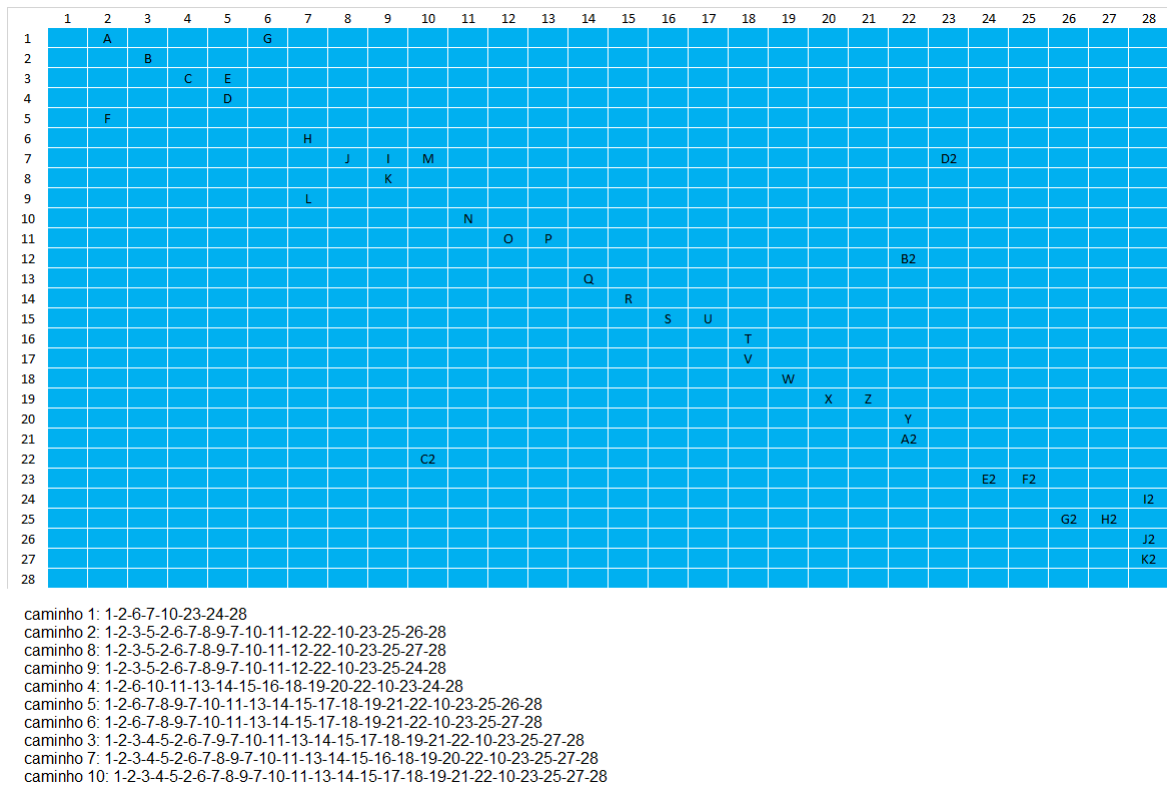


Figura 41 – Matriz e Caminhos para a função handleDelete.

6.3.3 Caminhos das funções para a Atualização de Entidades

Nesta sub-seção serão apresentados grafos, matrizes e caminhos para as funções que compõem os processos Atualização de entidades.

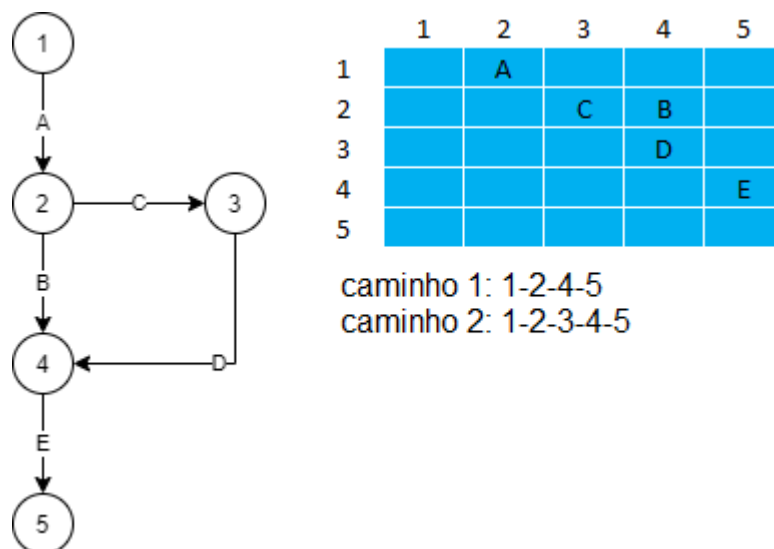


Figura 42 – Grafo, Matriz e Caminhos para a função addNewAttribute.

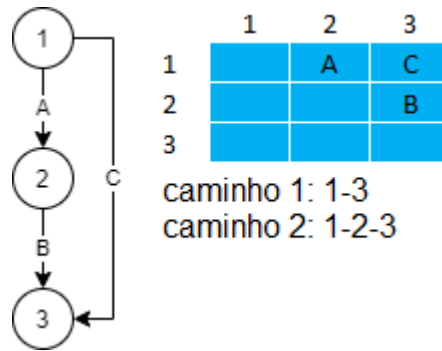


Figura 43 – Grafo, Matriz e Caminhos para a função deleteAttribute.

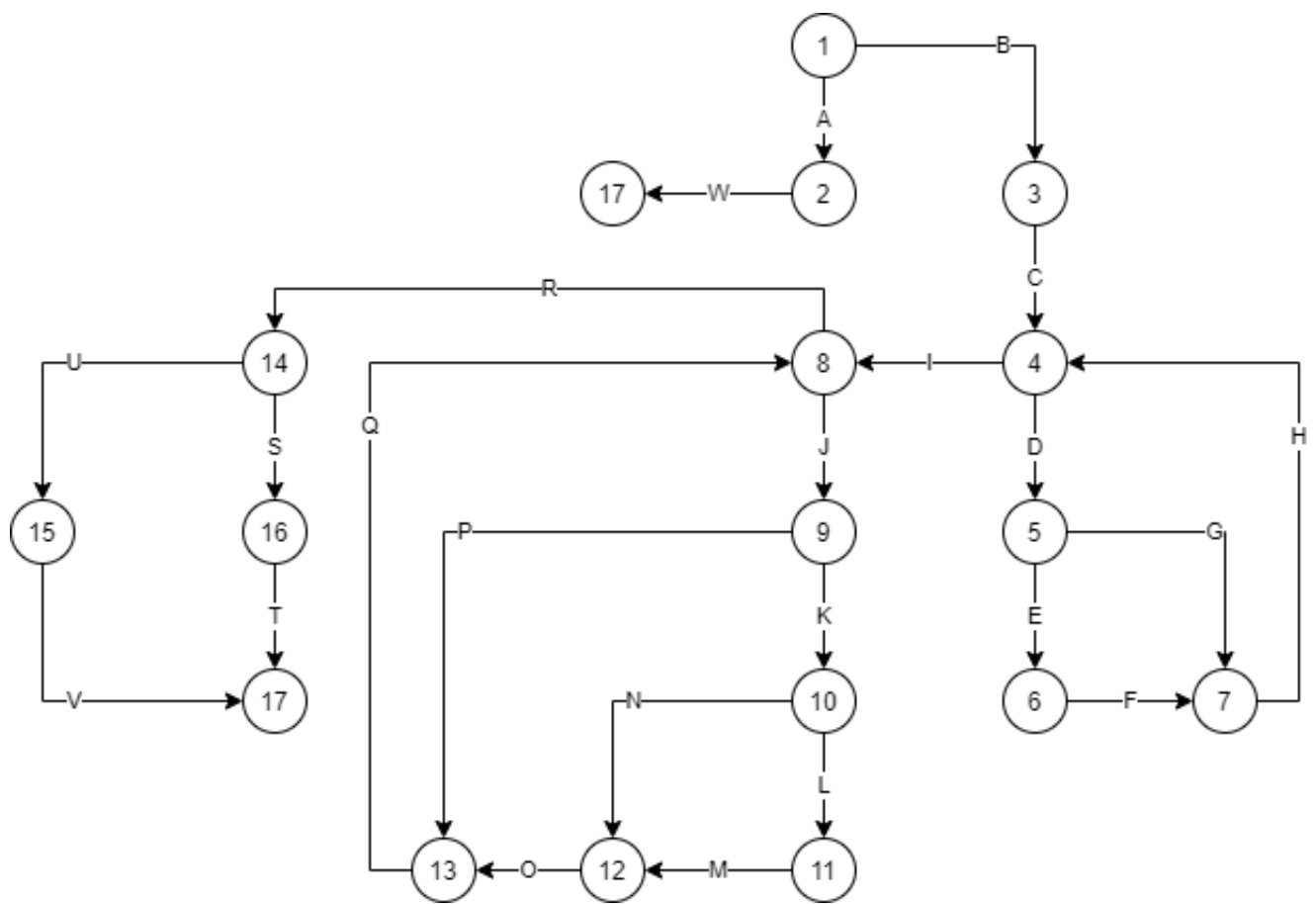


Figura 44 – Grafo para a função addNewRelationship.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1		A	B														
2																	W
3				C													
4					D			I									
5						E	G										
6							F										
7				H													
8									J					R			
9										K			P				
10											L	N					
11												M					
12													O				
13								Q									
14															U	S	
15																	V
16																	T
17																	

caminho 1: 1-2-17
 caminho 2: 1-3-4-8-14-16-17
 caminho 3: 1-3-4-8-14-15-17
 caminho 4: 1-3-4-5-6-7-4-8-14-16-17
 caminho 5: 1-3-4-5-6-7-4-8-14-15-17
 caminho 6: 1-3-4-5-7-4-8-9-13-8-14-16-17
 caminho 7: 1-3-4-5-7-4-8-9-13-8-14-15-17
 caminho 8: 1-3-4-8-9-10-12-13-8-14-16-17
 caminho 9: 1-3-4-5-6-7-4-8-9-10-11-12-13-8-14-16-17
 caminho 10: 1-3-4-5-6-7-4-8-9-10-11-12-13-8-14-15-17

Figura 45 – Matriz e Caminhos para a função addNewRelationship.

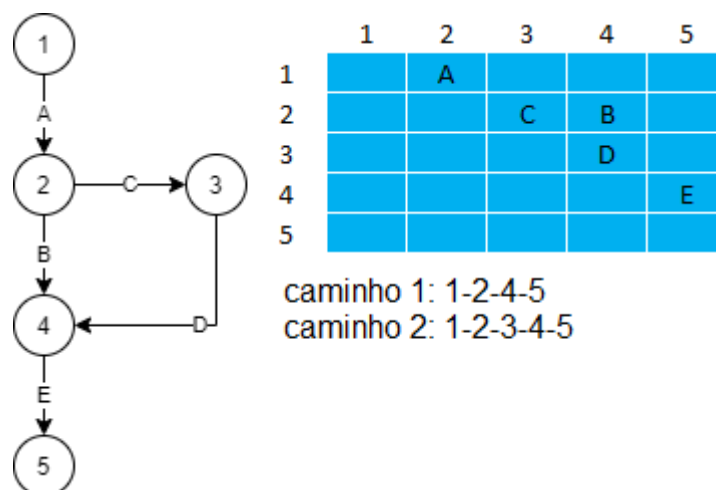


Figura 46 – Grafo, Matriz e Caminhos para a função deleteRelationship.

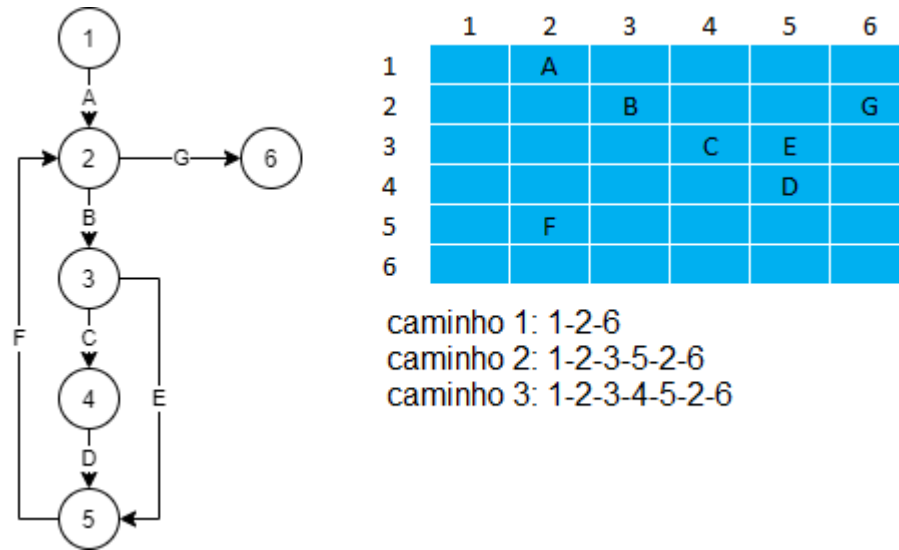


Figura 47 – Grafo, Matriz e Caminhos para a função searchDeletedRelationships.

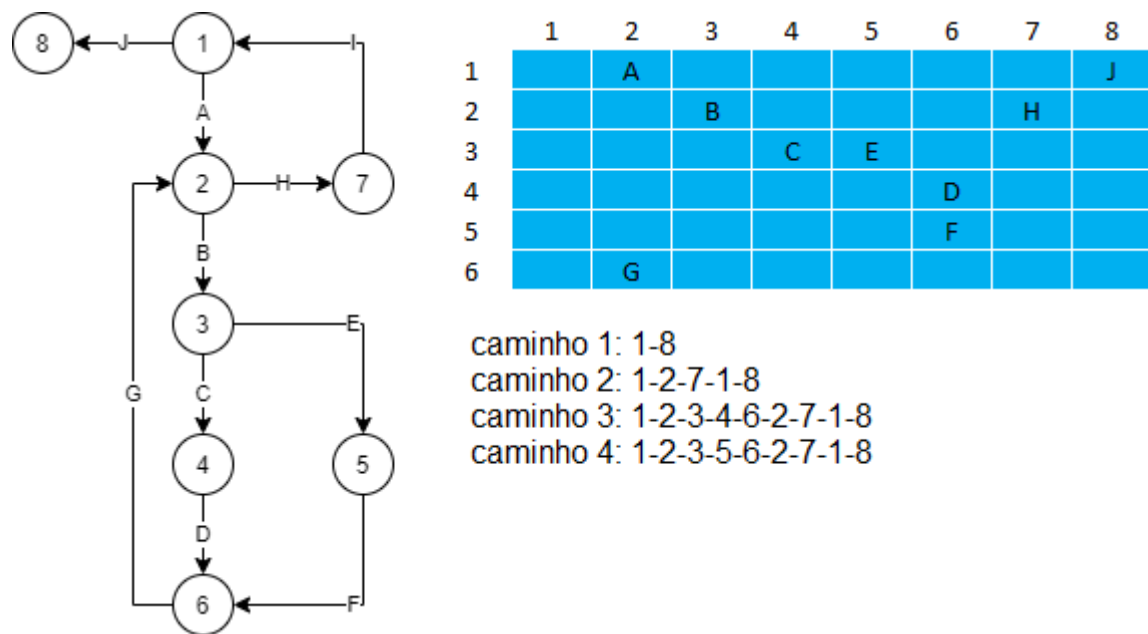


Figura 48 – Grafo, Matriz e Caminhos para a função updateRelatedEntities.

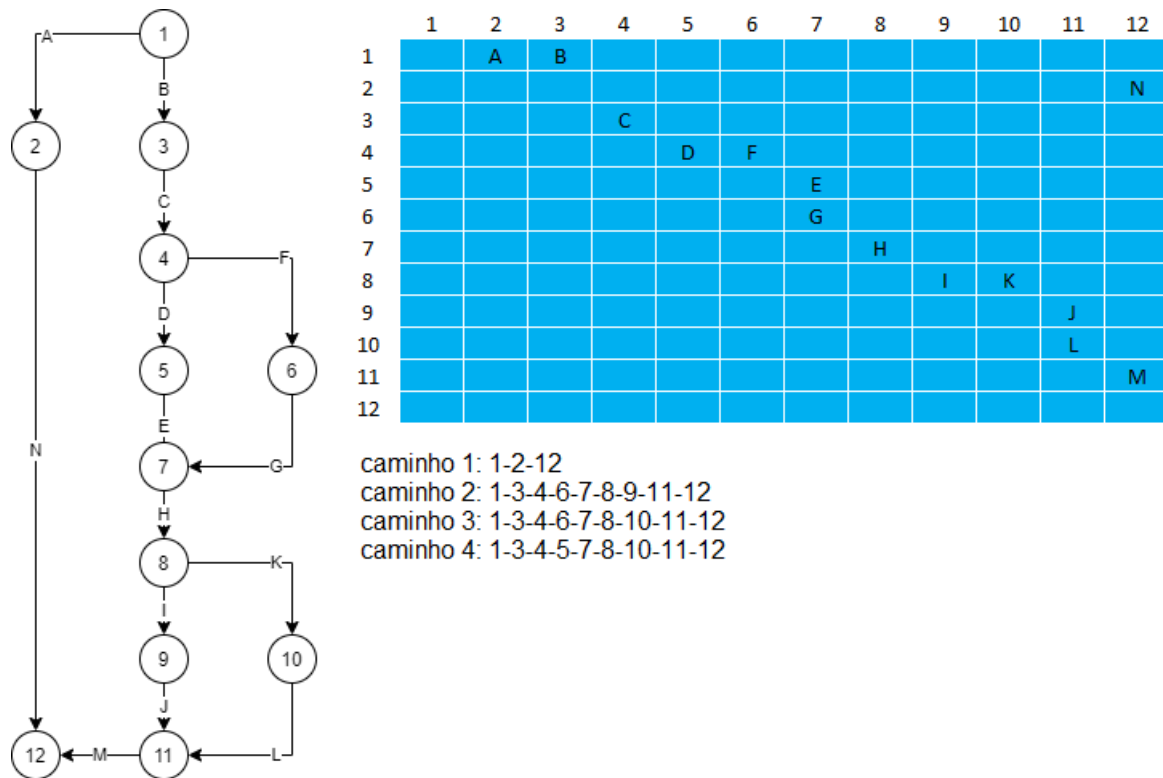


Figura 49 – Grafo, Matriz e Caminhos para a função deletePreviousRelationship.

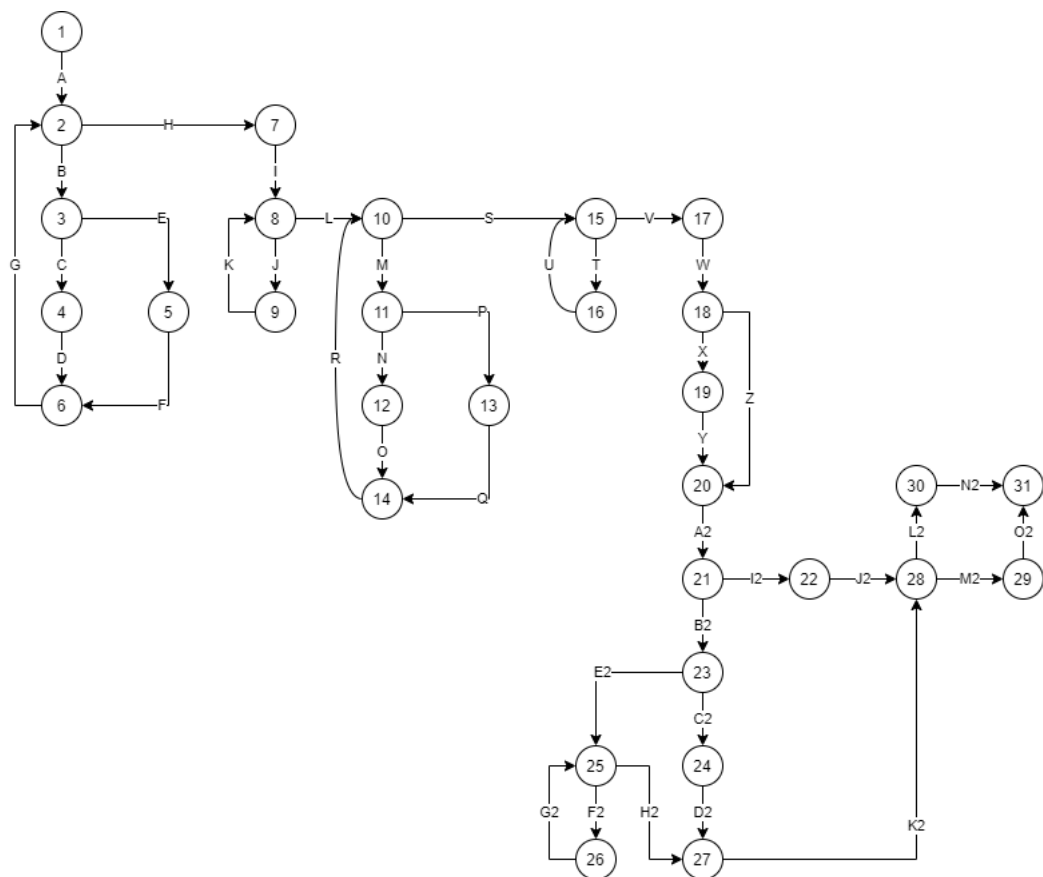


Figura 50 – Grafo para a função handleUpdate.

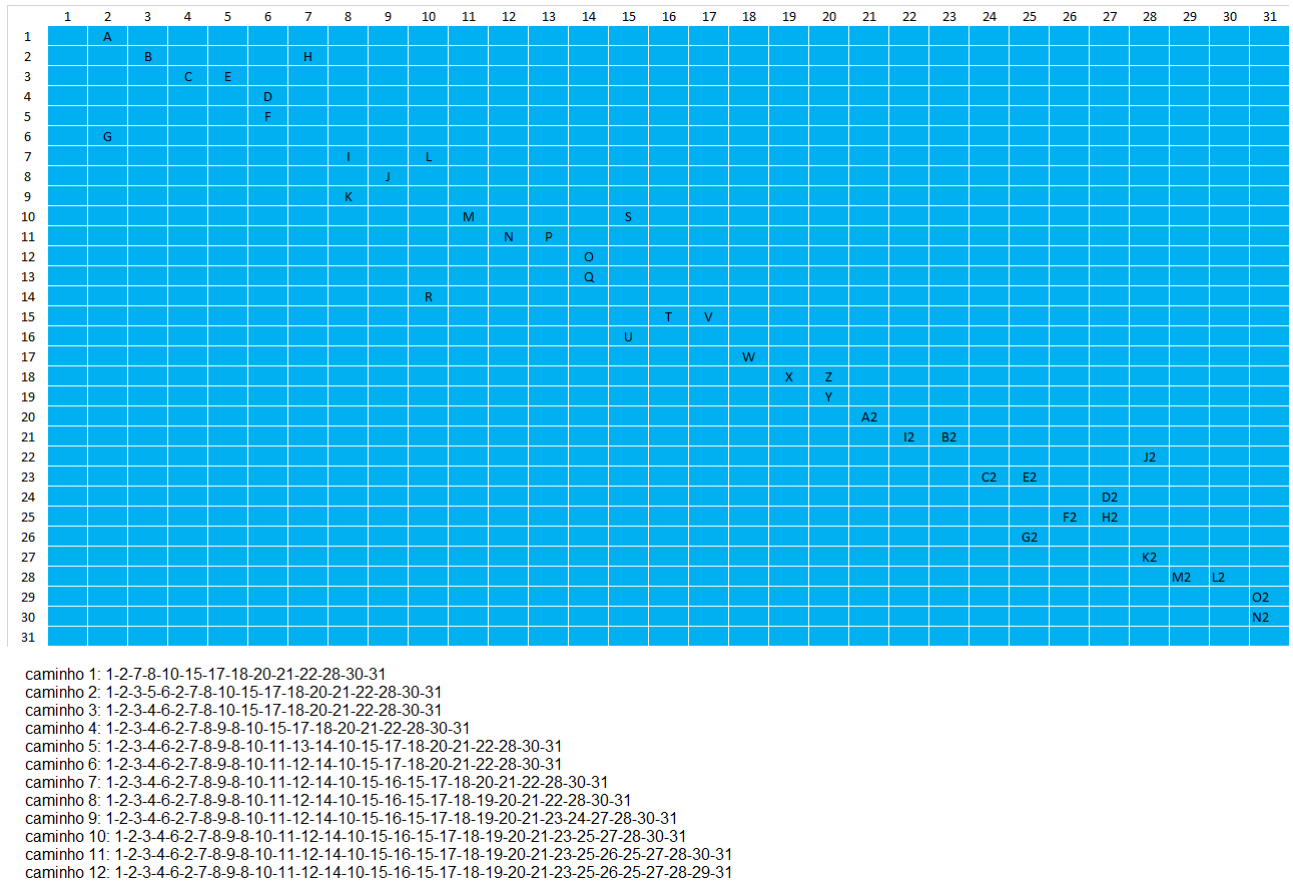


Figura 51 – Matriz e Caminhos para a função handleUpdate.

6.3.4 Teste com os Clientes e Resultados dos Testes

Depois do reconhecimento dos caminhos de cada uma das funções, foi possível a criação dos casos de teste, porém como as funções estão entrelaçadas, na qual para realizar uma é preciso fazer a outra primeiro, nós fizemos um passo-a-passo junto aos clientes para passar por todos testes necessários e que consigam andar pela maioria dos caminhos das funções, pois alguns caminhos específicos exigem situações extremas que não poderiam ser ocasionadas pelo usuário normalmente, mas que foram testadas pela nossa equipe.

Após a realização dos testes, os clientes se mostraram bastante satisfeitos com os resultados obtidos na aplicação, principalmente no processo de atualizar as entidades que se mostrou mais dinâmico e fácil de manusear, porém, eles perceberam que ao criar um novo atributo, se o tipo desse atributo for definido como *Number*, o seu valor não é convertido para esse tipo, pois o context-broker ORION não é capaz de ler o tipo do atributo e converter para o que foi especificado, portanto eles pediram para nós adicionarmos essa conversão no próprio site, antes de

enviar para o ORION, para evitar esse tipo de erro, e como se tratava de uma pequena mudança no software, não houve complicações em adicioná-la.

Apesar disso, os clientes permanecem com bastante interesse na realização do projeto, e se mostraram bastante animados com as próximas atualizações que serão feitas, no caso de prosseguirmos com esse projeto, contudo, para essa fase inicial, as principais funcionalidades como criar e editar entidades estão prontas, e o que nos resta é esperar um futuro próximo na qual tenhamos tempo e recursos para poder investir nessa aplicação e dar continuidade a este projeto.

6.4 Experiência da Equipe Durante o Desenvolvimento

A experiência que tivemos durante o desenvolvimento do projeto foi certamente única, e se provou essencial para o amadurecimento do nosso trabalho em equipe e capacidade de solucionar problemas. Inicialmente a nossa equipe tinha dúvidas em relação ao tema que seria utilizado no nosso projeto, então como nós participávamos do Laboratório de Internet das Coisas da FEI, o nosso grupo pediu ajuda para o pessoal do laboratório, logo, um dos mestrandos pediu para fazermos um software para a criação e edição de entidades, que seria específico para aqueles que trabalham no projeto SWAMP, e isso nos deixou muito animados, pois além de fazer um projeto totalmente inesperado, nós tínhamos a certeza de que ao terminá-lo, teriam pessoas que usariam ele, o que nos incentivou ainda mais a desenvolver o software da melhor forma possível no tempo disponível para a realização do projeto.

Durante o desenvolvimento em si do projeto, pelo menos nas primeiras semanas, a dinâmica da equipe estava muito boa, pois como já tínhamos realizado diversos trabalhos juntos, ficou mais fácil realizar a administração da equipe e dividir os papéis que cada um precisava fazer, e como éramos apenas dois, a melhor ideia que tivemos foi alternar entre os papéis disponíveis no desenvolvimento ágil, para que os dois pudessem obter experiência de cada um deles. No entanto, o que não esperávamos é que uma pandemia chegaria ao nosso país, e certamente para nós, assim como para os outros grupos, isso dificultou muito na comunicação e dinâmica do desenvolvimento, pois quase tudo era feito presencialmente com ambos os integrantes da equipe, e digamos que durante um tempo isso foi um abalo no desenvolvimento do projeto, porém, foi justamente essa situação que nos ajudou a nos tornarmos melhores, pois nós conseguimos contornar essa situação, auxiliando um ao outro no que era preciso e mantendo a comunicação o mais frequente possível acerca do projeto para que ninguém ficasse desanimado

e para que cumpríssemos o que foi prometido aos nossos clientes, utilizando meios como whatsapp, para comunicação, trello, na organização das tarefas, o overleaf, para a documentação, e o próprio github para o versionamento do código.

Apesar da nossa experiência ter sido realmente boa, a única coisa que nós gostaríamos de ter mudado nela seria ter tido um contato mais frequente com os clientes, pois nós tínhamos isso antes da pandemia, mas na época não imaginávamos o quão importante isso era para o desenvolvimento da aplicação, o que acabou resultando em um dos maiores erros dentro do software, que seria a forma como os atributos de relacionamento entre as entidades era armazenado, pois da forma que tínhamos feito, as aplicações que os clientes utilizam não poderiam usá-los daquela forma, e ainda por cima o método que nós utilizávamos não era tão eficiente, em questão de código, se comparada com a nova forma que foi implementada após o teste intermediário com os clientes, e que resultou em uma mudança, praticamente, do software inteiro, já que como foi explicado, o nosso software gira em torno dos relacionamentos, ele é a parte essencial da criação e edição de entidades, e mudar ela resultou na alteração de grande parte da aplicação, na qual se tudo tivesse sido esclarecido antes com os clientes, o projeto estaria adiantado, e não acabaríamos nos sobrecarregando, como foi o que aconteceu nas últimas semanas do projeto, contudo, isso abriu nossos olhos para a importância da comunicação frequente com o cliente, e como eles podem impedir que certos erros sejam cometidos e nos ajudar, mesmo que indiretamente, a facilitar o nosso trabalho.

REFERÊNCIAS

SEBRAE. O Quadro de Modelo de Negócios: um caminho para criar, recriar e inovar em modelos de negócios., p. 44, 2013. Disponível em:

<https://www.sebrae.com.br/Sebrae/Portal%20Sebrae/UFs/ES/Anexos/ES%7B%5C_%7DQUADROMODELODENEGOCIOS%7B%5C_%7D16%7B%5C_%7DPDF.pdf>.

SUTHERLAND, Jeff. **Scrum - A Arte de Fazer o Dobro**. [S.l.: s.n.], 2014. P. 5–6. ISBN 9788544100882.