CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

André Costa Silva

SAG 2.0: Adaptação do Sistema de Automação de Genogramas

Palmas – TO

2021

André Costa Silva

SAG 2.0: Adaptação do Sistema de Automação de Genogramas

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC 2) do curso de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Profª. Me. Cristina D'Ornellas Filipakis

Palmas – TO

2021

André Costa Silva

SAG 2.0: Adaptação do Sistema de Automação de Genogramas

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC 2) do curso de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Profª. Me. Cristina D'Ornellas Filipakis

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Profa. Me. Cristina D'Ornellas Filipakis

Orientadora

Centro Universitário Luterano de Palmas – CEULP

_____

Profa. Esp. Fernanda Pereira Gomes

Centro Universitário Luterano de Palmas – CEULP

_____

Profa. Me. Thaís Moura Monteiro

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2021

AGRADECIMENTOS

Em primeiro lugar quero agradecer a Deus, que me guiou, me capacitou, me deu sabedoria e força para continuar firme, pois sei que Ele esteve sempre comigo durante essa caminhada, e rendo a Ele louvores e gratidão por mais essa conquista.

Agradeço à minha família, aos meus pais Pedro Costa e Rosemir da Silva, por toda educação que me deram, por todo cuidado, por me apoiarem em tudo e por toda ajuda e incentivo.

Agradeço à minha orientadora Cristina Filipakis, por todo suporte, toda ajuda e por todas as orientações, sempre disponível em fazer com que tudo desse certo e que este projeto acontecesse. Obrigado por toda paciência e por compreender as dificuldades que passei durante essa caminhada.

Agradeço a todos os meus professores: Fabiano Fagundes, Jackson Gomes, Madianita Bogo, Parcilene Fernandes, Heloise Acco, Fábio Castro e Fernando Luiz, por proporcionarem o conhecimento e por toda a disponibilidade quando necessário.

Agradeço ao meu amigo Samuel Junior Germano, por todo apoio, amizade e incentivo nessa caminhada, sempre me guiando a auxiliando quando necessário.

RESUMO

SILVA, André Costa. **SAG 2.0: Sistema para Automação de Genogramas. 2020. 53 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia de Software, Centro Universitário Luterano de Palmas, Palmas/TO, 2020.**

O presente trabalho teve por objetivo a implementação de novas representações de gênero e relações familiares, assim como também a aplicação de manutenções adaptativas e corretivas no SAG 1.0; com o intuito de auxiliar psicólogos e terapeutas em sua prática profissional com famílias. Para tanto, foram apresentados alguns conceitos, como o genograma e sua importância para os profissionais que atendem famílias, o processo de manutenção corretiva e adaptativa, a arquitetura selecionada para o desenvolvimento das funcionalidades, sendo a mesma a arquitetura Flux, o processo de desenvolvimento de software, sendo baseado em FDD (*Feature Driven Development*), e o conceito e estrutura dos Testes Funcionais. A metodologia deste trabalho apresenta as tecnologias que foram utilizadas para o desenvolvimento do sistema e os passos seguidos, desde a coleta de requisitos, modelagem até a realização de testes funcionais. Como resultado, foi entregue um software web adaptado com novos recursos visuais, possibilitando a criação de genogramas e um conjunto de novas funcionalidades aos profissionais que farão seu uso.

Palavras-chave: Genograma; Manutenção; Tecnologias.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O genograma, segundo a classificação de Carter e McGoldrick (1989/1995), é a representação gráfica (através de símbolos que incluem basicamente círculos, quadrados e linhas) da composição familiar e dos relacionamentos básicos de uma pessoa em, pelo menos, três gerações. Através dele, pode-se visualizar de uma forma rápida e clara quais membros constituem a família, seja com vínculo sanguíneo ou não. Além disso, permite identificar o estado atual da família, assim como as relações de interação presentes entre as pessoas representadas. Também pode-se acrescentar informações como a idade, ocupação de cada pessoa, situação dos casais (se ocorreu separação, divórcio ou concubinato e há quanto tempo ocorreu).

Além disso, o genograma permite também a representação da ocorrência de adoção, aborto, natimorto ou nascimento de gêmeos. Ainda, é possível identificar as doenças agudas e crônicas dos indivíduos e as pessoas já falecidas, sendo registrado o ano e o motivo de cada morte, a procedência das pessoas, data de migração, a ocorrência de alcoolismo, obesidade, uso de drogas, encarceramento, aposentadoria, entre outros.

Com relação aos padrões de interação familiar, segundo a classificação de Carter e McGoldrick (1989/1995), pode-se registrar, através do genograma, a ocorrência de relacionamentos muito próximos, relacionamentos conflituados, relacionamentos distantes, rompimentos, desavenças ou relacionamentos fusionados e conflituados, entre duas ou mais pessoas. Sua utilização tem se mostrado adequada para a pesquisa com famílias em diferentes fases de transição, em processos psicoterapêuticos, em famílias de crianças acometidas por doenças crônicas, famílias de idosos, com transtorno mental, famílias de adultos com câncer, entre outras (FILIZOLA et al., 2004; NIEWEGLOWSKI, 2004; WENDT, 2006).

O genograma é um instrumento rotineiramente utilizado por profissionais de diversas áreas, como da psicologia, da enfermagem e da assistência social, principalmente para a compreensão de processos familiares, pois se constitui de recursos para avaliar a composição familiar e as interações que ocorrem entre os membros da família e fora dela. Esta técnica diagramática tem sido usada por um amplo número de terapeutas familiares de diferentes orientações, que coincidem em dar importância ao contexto familiar multigeracional. Sendo inserido na conversação terapêutica, transcende suas origens funcionalistas, para transformar-se num recurso de compreensão colaborativa.

Para isso, a utilização de ferramentas para a criação do genograma tem se tornado cada vez mais comum, pois possibilita por meio de forma rápida e simples um conjunto diversificado de informações que vão desde os aspectos genéticos, médicos, sociais, comportamentais até os relacionais e culturais, podendo auxiliar os profissionais que acompanham famílias.

Dito isso, este trabalho partiu do problema de como implementar novas representações de gênero e relações familiares e realizar uma manutenção adaptativa e corretiva no SAG 1.0, que consiste em detectar possíveis erros que não foram identificados na fase de desenvolvimento e também modificar o software já existente tornando-o mais eficiente ou condizente com a atualidade.

Para realização dessa implementação de novas funcionalidades e manutenção corretiva optou-se por utilizar a arquitetura Flux em conjunto com o método de desenvolvimento *Feature Driven Development* (Desenvolvimento Dirigido a Características). A arquitetura Flux consiste numa arquitetura de gerenciamento de dados para desenvolver aplicações web, em que os dados fluem apenas em uma direção. Um controlador geral é responsável por realizar mutações no estado da aplicação e informar à página essa mudança de estado. O FDD tem como objetivo iniciar pequenas iterações que normalmente duram em torno de duas semanas, sendo que ao final acontece a entrega de uma parte do software funcionando (HIGHSMITH, 2002).

Portanto, através deste trabalho pretende-se realizar uma manutenção corretiva e adaptativa com o objetivo de melhorias e adequação do software à proposta de negócio. Assim como a implementação de novas representações de gênero e relações familiares, provendo maior variedade de funcionalidades.

Como objetivo geral, foi implementado novas representações de gênero, de orientação sexual e de relações familiares e aplicar manutenções adaptativas e corretivas no SAG 1.0. Para que o objetivo geral fosse atingido foram modeladas funcionalidades através de diagramas UML e Artefatos, seguido pela implementação das funcionalidades utilizando a arquitetura Flux e o método FDD. No decorrer do desenvolvimento, foram realizados testes funcionais e de usabilidade buscando garantir a qualidade do software.

2 REFERENCIAL TEÓRICO

2.1 ESTUDO DO SAG 1.0: GENOGRAMA E SUAS RELAÇÕES FAMILIARES

Segundo Santiago (2019), o SAG teve por objetivo o desenvolvimento de um sistema para automação de genogramas, no qual tinha por motivação a necessidade de um software para auxiliar os profissionais e estudantes que acompanham famílias, tornando possível mapear a evolução familiar através da criação de genogramas. Dito isso, foram desenvolvidas as seguintes representações gráficas para alcançar tal objetivo, conforme a Figura 1.

O Genograma é a representação gráfica (através de símbolos) utilizado para mapear a estrutura familiar, através do uso de símbolos que são atribuídos a cada membro da família. Eles são baseados na suposição teórica de que o funcionamento dos membros familiares em diferentes níveis, físico, social e emocional, é interdependente, e quando uma parte do sistema familiar muda, todo o resto é afetado (MARCHETTI-MERCER ; CLEAVER, 2000).

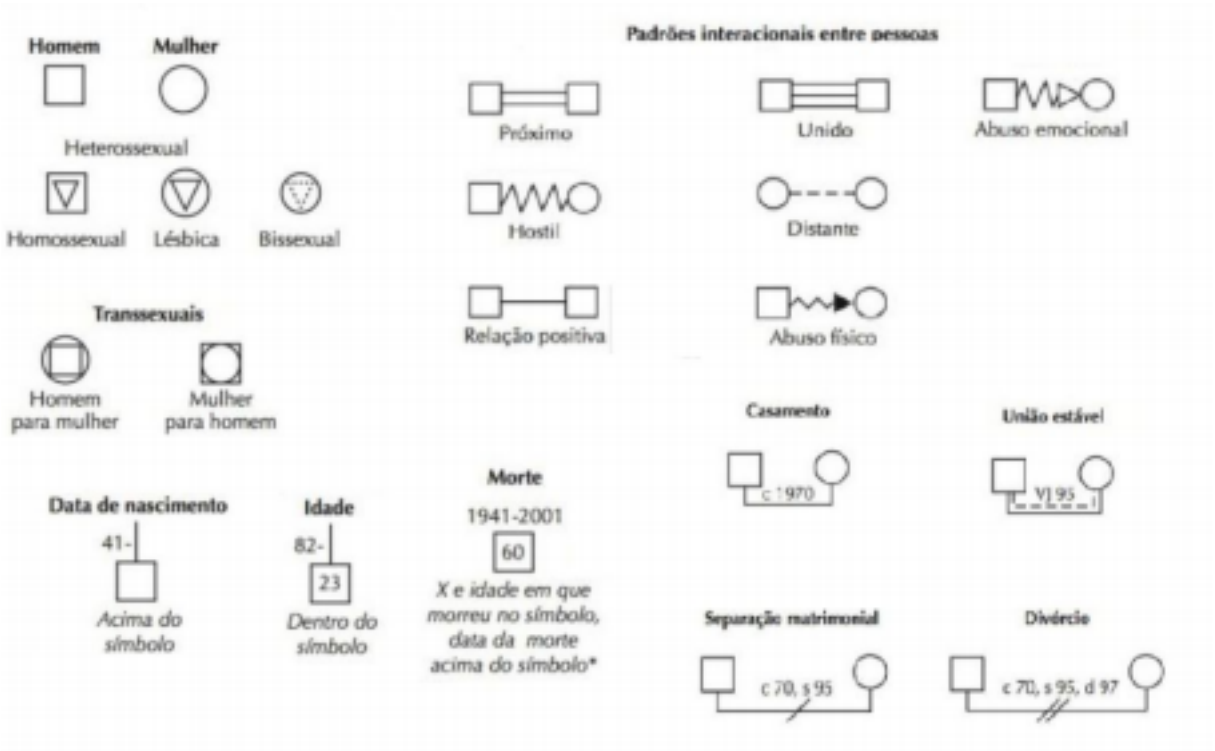

Figura 1: Representações gráficas inclusas no SAG 1.0

O Genograma pode ser considerado como um instrumento que auxilia a família a expressar-se, e que vem somar-se à gama de instrumentos de coleta de dados, como os relatos orais para estudos de caso, histórias de vida e entrevistas reflexivas, assim como afirma Szymanski (2004). Na terapia e no aconselhamento familiar, o Genograma é utilizado como

um instrumento para engajar a família, destravar o sistema, rever dificuldades familiares, verificar a composição familiar, clarificar os padrões relacionais familiares e identificar a família extensa. Frequentemente, sua confecção identifica a razão pela qual a família procura a terapia, ou seja, clarifica a demanda existente por trás da queixa explicitada pela família (MCGOLDRICK ; GERSON, 1985/2005; ROTTER ; BUSH, 2000).

De acordo com Nichols e Schwartz (1998), a principal função do Genograma é organizar os dados referentes à família durante a fase de avaliação e acompanhar os processos de relacionamento e de triângulos relacionais no decorrer da terapia. O modo como o Genograma é feito, dispõe as informações da família graficamente de forma a oferecer uma visão compreensiva dos complexos padrões familiares. Ao mesmo tempo, possibilita a criação de uma série de hipóteses sobre como o problema clínico da família pode conectar-se ao contexto, bem como a evolução de ambos, problema e contexto, ao longo do tempo (MCGOLDRICK; GERSON, 1995).

Segundo Hardy ; Laszloffy (1995), profissionais que estejam realizando a formação em terapia familiar comumente confeccionam o Genograma de suas famílias de origem, com o intuito de averiguar a composição e dinâmica familiar, elucidando seus padrões, regras, valores, crenças e mitos e sua influência na prática profissional. Autores como Carter e McGoldrick (1995) salientam que certos padrões familiares, em uma mesma família, são recorrentes e, por essa razão, é possível fazer determinadas predições sobre os processos futuros que a família vivenciará baseando-se na utilização do Genograma. De acordo com Bowen (1979/1991), deste modo, passado e presente são examinados para se obter possíveis informações sobre o futuro.

Assim como afirma Wendt e Crepaldi (2008), as informações obtidas através do Genograma podem incluir aspectos genéticos, médicos, sociais, comportamentais e culturais da família, sendo evidenciados os seguintes dados: (a) os nomes e idades de todos os membros da família; (b) datas exatas de nascimentos, casamentos, separações, divórcios, mortes, abortos e outros acontecimentos significativos; (c) indicações datadas das atividades, ocupações, doenças, lugares de residência e mudanças no desenvolvimento vital; e (d) as relações entre os membros da família.

Atualmente, o Genograma tem sido difundido como um instrumento científico para coleta de dados, especificamente em pesquisas qualitativas com famílias. Sua utilização tem

se mostrado adequada para a pesquisa com famílias em diferentes fases de transição, em processos psicoterapêuticos, em famílias de crianças acometidas por doenças crônicas, famílias de idosos, com transtorno mental, famílias de adultos com câncer, entre outras (FILIZOLA ET AL., 2004; NIEWEGLOWSKI, 2004; WENDT, 2006).

Neste estudo, o que se pretende é ilustrar a utilização do genograma como um instrumento para as diferentes fases vivenciadas pela família, a fim de explicitar as modificações familiares ao longo do tempo.

2.2 MANUTENÇÃO DE SOFTWARE

O processo de manutenção de software ocorre após a conclusão de todas as etapas do ciclo de vida do software. Ele consiste em manter, conservar, gerenciar e administrar um produto ou serviço de software (BARROS, 2019). Em outras palavras, a modificação de um produto de software após a entrega tem como objetivo corrigir falhas, aprimorar o desempenho ou outras características ou adaptação a um cenário modificado (MAMONE, 1994).

A manutenção de software tem como objetivo modificar um produto existente, tornando-o mais aderente às necessidades em que foi designado. Como por exemplo, adaptar-se às mudanças do ambiente estando de acordo com a atualidade, ou a correção de erros que não foram identificados nas fases do desenvolvimento.

De acordo com LIENTZ, SWANSON, TOMPKINS (1978) a atividade de Manutenção de Software se desdobra em quatro grandes grupos: Manutenção Perfeccionista, Preventiva, Adaptativa e Corretiva. A manutenção perfeccionista tem como objetivo as mudanças que foram originadas através de solicitações externas, melhorias de segurança, performance, dentre outros (MALL, 2018). Refere-se a mudanças que não são diretamente ligadas a defeitos da aplicação, somente do seu amadurecimento.

A manutenção preventiva tem como objetivo o processo de melhoria e otimização de um software já desenvolvido, como também reparo de defeitos. Assim como afirma Bayer (2019), a manutenção preventiva não só consiste em analisar as modificações que podem ser feitas no software para melhorar a eficiência, mas também identificar os problemas recorrentes e tomar medidas para mitigá-los.

Em alguns momentos durante a vida de um software será necessário adaptá-lo ao ambiente em que ele está inserido. Seja pela inserção de uma nova tecnologia, um novo sistema operacional (REZENDE, 2005). Essa mudança é conhecida como manutenção adaptativa, quando às vezes se faz necessário adequar o software à proposta de negócio, tendo como objetivo evitar futuros problemas.

A manutenção corretiva está ligada às modificações necessárias para resolução de problemas que foram encontrados, que causam um impedimento no funcionamento correto do software. Sendo que os mesmos podem ir desde problemas de interpretação à falhas de codificação (GRUBB, 2003). Assim como afirma Barros (2019), este tipo de manutenção se limita a correção de falhas já existentes no sistema que por algum motivo não satisfazem os requisitos necessários, e impedem que o software esteja de acordo com as especificações levantadas no início do projeto. E estes erros vão desde problemas de interpretação às falhas de codificação.

2.3 ARQUITETURA FLUX

A Arquitetura Flux é uma arquitetura de gerenciamento de dados que o Facebook usa para desenvolver aplicações web. Em 2014, o Facebook propôs a arquitetura Flux, com um fluxo de dados unidirecional dos componentes para um gerenciador central de estado da aplicação (FACEBOOK INC., 2014).

De acordo com Boduch (2016), Flux é uma arquitetura de fluxo de dados unidirecional. Isso significa que o Flux impõe a direção dos fluxos de dados e, portanto, elimina a possibilidade dos componentes se atualizarem em uma ordem que quebra o sistema. Não importa os dados que acabaram de entrar no sistema, eles sempre irão fluir pelo sistema na mesma ordem que quaisquer outros dados, conforme ilustrado na Figura 2.

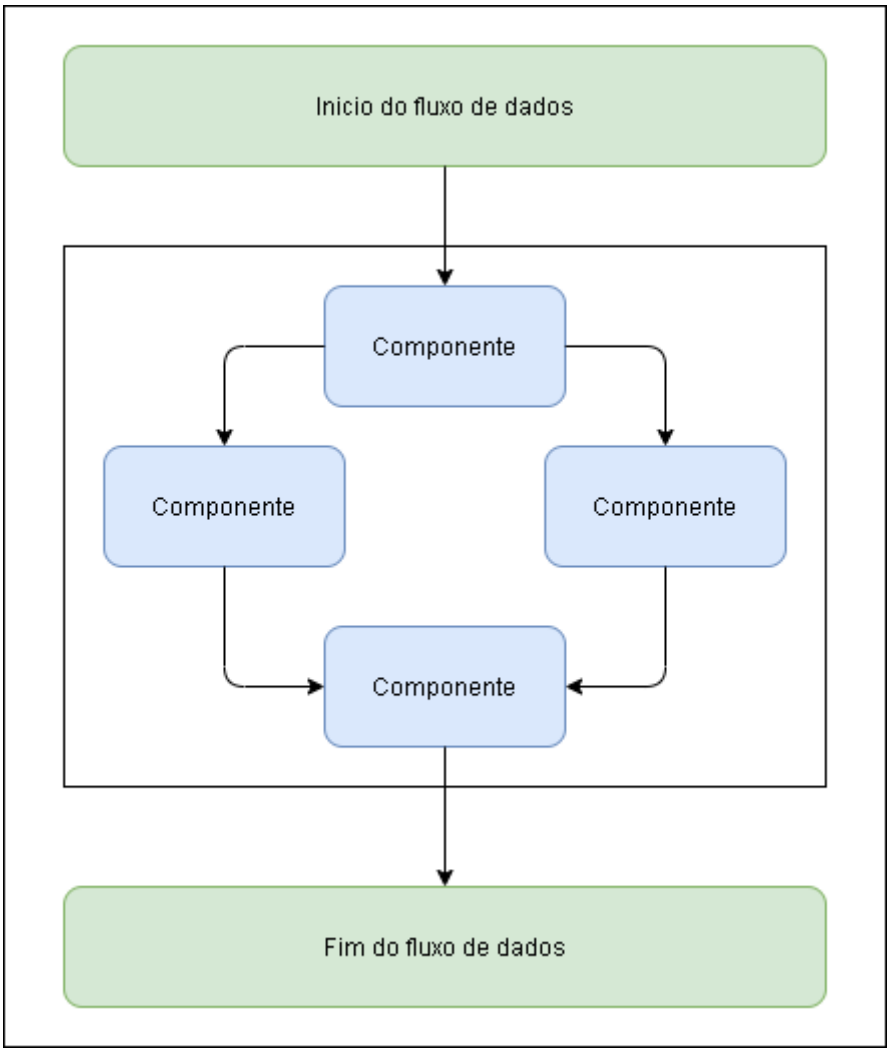

Figura 2: Fluxo (Adaptado de Boduch (2016)).

Para entender Flux de uma maneira mais simples, é necessário entender quais problemas esse padrão resolve e quais problemas o Facebook enfrentou que serviram de motivação para seu desenvolvimento.

No Facebook, quando o usuário recebe uma mensagem de outra pessoa, é incrementado, na página, o número de mensagens não visualizadas. Ao clicar no ícone que representa essa informação, é possível ver a mensagem recebida e interagir com ela. É um sistema simples que espera por um dado e, ao receber esse dado, incrementa um número. Com o aumento da complexidade da aplicação e do número de usuários, esse sistema começou a apresentar falhas. Em muitos casos, a mensagem chegava, o contador era incrementado, mas ao clicar no ícone, nada aparecia, como se o usuário não houvesse recebido mensagem alguma.

O Facebook decidiu, então, criar uma arquitetura diferente, na qual os dados fluem apenas em uma direção. Cada ação do usuário deve ser previsível e deve ser passada a um sistema expedidor que informa essas ações a um controlador geral, responsável por realizar mutações no estado da aplicação e informar à página essa mudança de estado. Eles chamaram essa arquitetura de Flux (GOMES, 2016).

A arquitetura Flux é formada por um conjunto de quatro interfaces distintas que comunicam-se de forma unidirecional, sendo representadas na Figura 3.

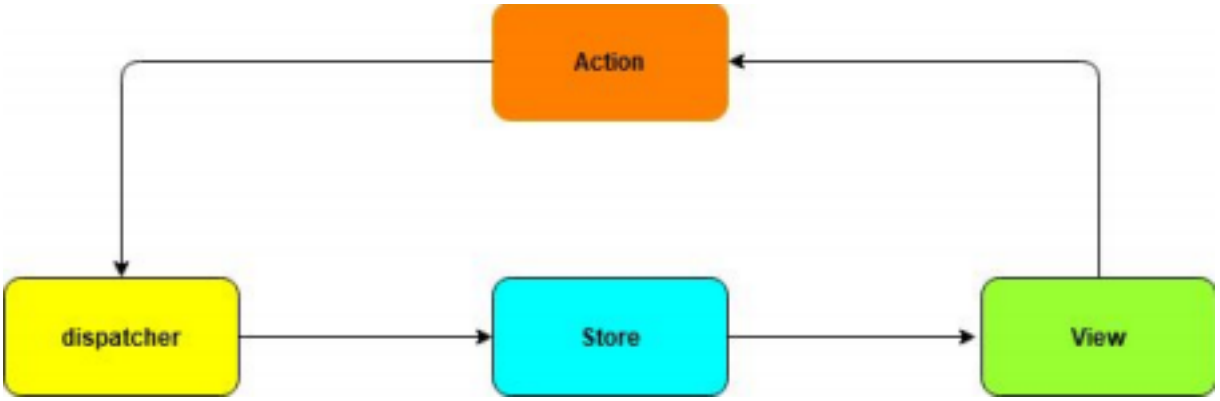

Figura 3: Arquitetura Flux.

O *Dispatcher* é responsável por centralizar o registro de *callbacks* e emissão de eventos. Ele recebe as ações e as encaminha para a *Store*. A *store* é responsável por manter os dados de uma aplicação. A alteração de dados em uma *store* é feita através de *actions* recebidas através do dispatcher. A *view* no Flux tem exatamente a mesma função de exibição de dados que o MVC.

A essa arquitetura, Gamma, Helm, Jonhson e Vlissides (1995, p. 4) declara:

> A arquitetura MVC desacopla as Camadas de Visão e Modelo e estabelece um protocolo entre elas. A Camada de Visão deve ser apresentada como um reflexo do modelo de dados. Sempre que ocorrem mudanças no modelo de dados, as visões que dependem dele são atualizadas. Em resposta a estas mudanças, cada interface tem a oportunidade de atualizar seu estado. Esta abordagem permite conectar várias interfaces ao mesmo modelo de dados. Desta forma é possível criar novas interfaces sem a necessidade de reescrever o modelo.

[1]**Callbacks:** Método de callback é um pedaço de código executável que é passado como parâmetro para algum método, é esperado que o método execute o código do argumento em algum momento.

Por este motivo a *view* pode ser qualquer estrutura, assim como em outras

arquiteturas. A *action* é de fato a API interna da sua aplicação. Elas descrevem a forma que qualquer coisa interage com a sua aplicação.

A primeira fase do Flux, como citado anteriormente, é a *Dispatcher*, que consiste em uma central responsável por registrar *callbacks* e emitir eventos. Por ser uma central haverá apenas um único *Dispatcher* para toda sua aplicação, tendo como objetivo repassar para *store* o que precisa ser feito. A segunda fase é a *Store*, que será responsável pelos dados e lógica da aplicação. É nela que está contida toda a lógica de implementação das *actions*. A *Store* é a única responsável por saber quando e como atualizar a *View*, sendo assim, ela avisará quando concluir a ação. A *View* então consome os dados presentes na Store. A terceira fase é a *View*, que é simplesmente os componentes visuais que são apresentados, ou seja, tudo o que aparece na tela para o usuário. Se tiver algum formulário ou botão, HTML em geral, estamos falando da *View*. A quarta e última fase é a *Actions*, que são as ações que o usuário realiza na *View*. Quando é solicitada alguma ação, elas são representadas por objetos que apenas possuem as novas informações a serem salvas pela aplicação. Ou seja, a *actions* consiste simplesmente nos objetos que usamos para padronizar a comunicação entre *View* e *Dispatcher*.

A arquitetura Flux será utilizada neste trabalho pois possui uma série de propriedades que a torna singular e promove uma série de vantagens quando comparada às demais arquiteturas de software. Essa arquitetura mantém todos os dados centralizados em um único ponto e faz com que o fluxo de dados seja explícito, fácil de desenvolver e dar manutenção (sobretudo pela facilidade de localizar bugs quando eles acontecem).

2.4 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O processo de desenvolvimento de software, segundo Jalote (1997), é composto por uma sequência de passos, ou etapas, cada uma das quais executando um conjunto de atividades bem definidas que conduzem à conclusão do projeto. Embora existam muitos processos de software diferentes, este trabalho tem como foco o FDD que busca o desenvolvimento por funcionalidade, ou seja, por um requisito funcional do sistema.

De acordo com Beck (2001), o método *Feature Driven Development* (Desenvolvimento Dirigido a Características), também conhecido como FDD, surgiu como uma reação aos métodos tradicionais que valorizavam a criação de uma documentação de

software completa. Elas são caracterizadas por serem incrementais com entregas frequentes, curtos ciclos de vida nas fases do processo e que aceitam as mudanças durante o desenvolvimento, com simplicidade no aprendizado e utilização da documentação (ABRAHAMSSON, SALO, et al., 2002).

O FDD tem como objetivo iniciar pequenas iterações que normalmente duram em torno de duas semanas, quando ao final acontece a entrega de uma parte do software funcionando (HIGHSMITH, 2002). Segundo Abrahamsson e Salo (2002), o FDD baseia-se em um conjunto de ações que devem ser utilizadas para assegurar o êxito do método em um projeto de software.

Os princípios desse método ágil foram definidos em um workshop em 2001 (HIGHSMITH ; COCKBURN, 2001), sendo eles:

- Valorizar os indivíduos e interações sobre processos e ferramentas, pois essas interações facilitam o compartilhamento de informações, discussão e adaptação das mudanças no projeto, em vez de depender somente da qualidade do processo e das ferramentas.
- Valorizar a colaboração do cliente sobre o contrato negociado, pois é necessário estabelecer as prioridades com o cliente e alinhar o desenvolvimento do produto final, em vez de seguir rigorosamente o que foi definido no contrato. - Valorizar o software funcionando acima de documentação exaustiva, pois é mais importante entregar um software funcionando do que somente produzir documentações completas do software. Valorizar as respostas às mudanças acima de execução de um plano, pois é necessário adaptar o produto às mudanças das necessidades do cliente do que seguir rigorosamente todas as etapas somente para executar um plano.

O FDD foi criado em 2000 por Jeff Luca, Peter Coad e Stephen Palmer (ABRAHAMSSON, SALO, et al., 2002). Essa metodologia engloba as boas práticas de desenvolvimento de software relacionadas à modelagem de objetos de domínio, desenvolvimento por artefato e gerenciamento de configuração (PALMER ; FELSING, 2002).

A metodologia FDD é formada por cinco processos divididos em duas fases:

concepção e planejamento, construção, sendo representadas na Figura 4.

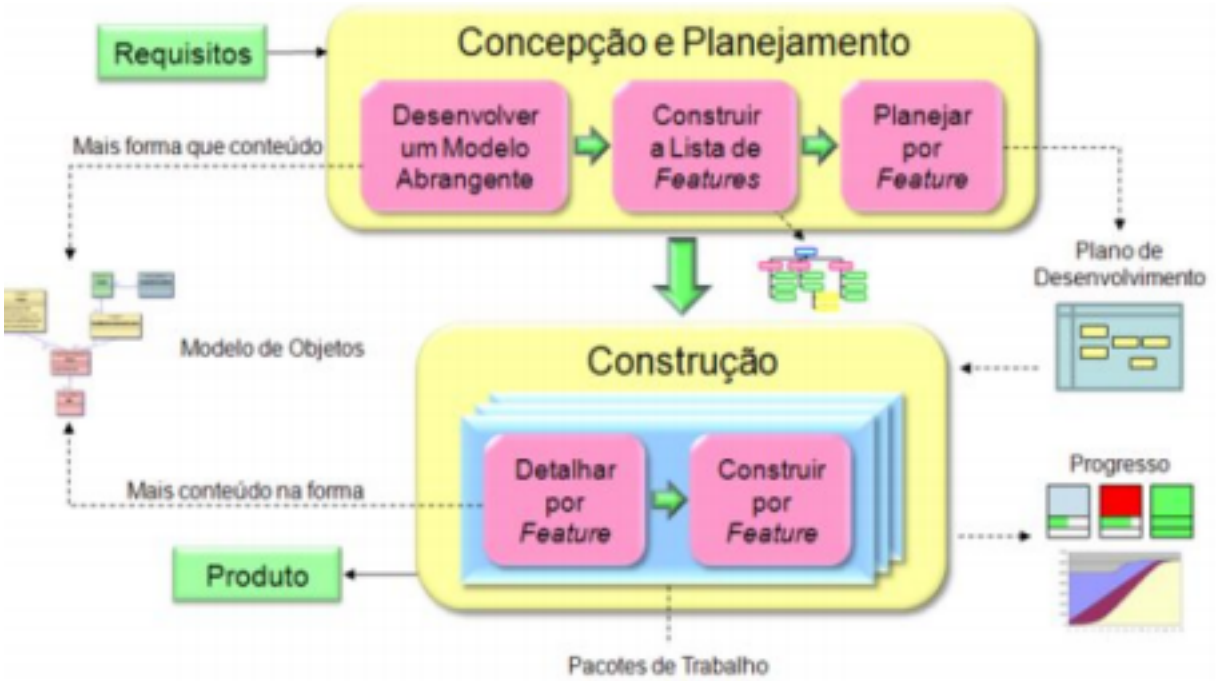

Figura 4: Processos do FDD (Ricardo (2014))

A fase de concepção e planejamento é formada por três processos: desenvolver um modelo abrangente, construir a lista de funcionalidade e planejar por funcionalidade. A fase de Construção é formada por dois processos: Detalhar por Funcionalidade e Construir por funcionalidade (PALMER ; FELSING, 2002).

O primeiro processo da fase de Concepção e Planejamento é desenvolver um modelo abrangente, o qual abrange todo o projeto. Nesse processo é formado o time de modelagem, sendo composto por especialistas de negócio e programadores. Inicialmente, os especialistas de negócio apresentaram para o restante do grupo uma visão do produto e depois realizaram apresentações de pequenas partes do negócio. Por meio do quão complexo é a área de negócio apresentada, o grupo pode solicitar um pequeno período de tempo para estudar a documentação fornecida pelo especialista de negócio. Após as apresentações, o grupo é dividido em subgrupos, que elaboraram uma proposta de modelo não detalhada para aquela parte específica do negócio. Então, as propostas são apresentadas e uma delas, ou uma junção de várias, é escolhida para ser o modelo. O modelo escolhido é incluído no modelo abrangente do produto. Este modelo abrangente é o resultado da junção de todos os modelos escolhidos para cada parte do negócio apresentado. São incluídas então notas e observações nesse modelo, e essas atividades vão se repetindo até que um modelo abrangente que cubra tudo aquilo que foi previsto nas partes de negócio para o produto seja encontrado.

O segundo processo é a construção da lista de funcionalidades (*Features*), o qual também se trata de um processo executado apenas no início do projeto. Esse processo é formado por um time que normalmente é composto pelos programadores-chefes que participaram do processo anterior e é construída uma lista de funcionalidades. Utilizando as partes do produto que foram identificadas no processo anterior, chamadas aqui de áreas de negócio, o grupo deve identificar as atividades dessas áreas e a partir dessas atividades, as funcionalidades que as constituem.

O terceiro processo é o processo abrangente, é executado somente uma vez no início do projeto. Durante esse processo é formado o time de planejamento, no qual esse time faz a sequência do desenvolvimento do projeto se baseando em suas dependências, na equipe de desenvolvimento, na carga horária e na complexidade das funcionalidades. Depois são atribuídas aos programadores-chefes responsabilidades sobre um conjunto de atividades de negócio, que serão responsáveis por todas as funcionalidades que a compõem. Então, são atribuídos "donos", programadores, às classes, que serão responsáveis pela manutenção delas. Assim, as classes são distribuídas com base na experiência de cada programador.

A fase de construção é formada por dois processos, o primeiro é o construir por Funcionalidade, o qual é executado uma vez para cada funcionalidade. Nesse processo formam-se as equipes de funcionalidades, quando o programador-chefe chama os "donos" de cada classe para participar da equipe. Dependendo do quão complexa seja a funcionalidade, o programador chefe pode convocar um especialista de negócio para esclarecer dúvidas a respeito daquela funcionalidade em si. Ainda dependendo do quão claro ficou para o time sua função, pode ser reservado um tempo para ser estudada a documentação de negócio e as anotações relacionadas àquela funcionalidade. É feito então um diagrama de sequência relacionado àquela funcionalidade e é refinado o modelo abrangente, já incluindo métodos e atributos nas classes envolvidas nela. Com essas informações, cada programador cria os prólogos de suas classes, com cabeçalhos de métodos com tipagem de parâmetros, atributos e outros, não fazendo nenhuma implementação ainda. Então, o programador-chefe dessa funcionalidade convida outro membro da equipe para avaliar o desenvolvimento de sua classe durante o processo.

O segundo e último processo é o construir por funcionalidade (Features), este processo também é executado uma vez para cada funcionalidade. Nele são implementadas as classes e métodos de acordo com a visão abrangente e o detalhamento realizado nos

processos anteriores. Então, cada desenvolvedor convida um membro de outra equipe para avaliar aquilo que foi desenvolvido em sua classe durante este processo. E após isso os desenvolvedores ficam responsáveis por testar suas classes e métodos e assegurar que as necessidades do negócio sejam alcançadas. E com as classes testadas e funcionando, pode ser então feito o *build*.

O *build* ou entrega final do produto consistirá na junção de todas as partes produzidas em cada iteração, sendo que cada parte englobará as funcionalidades desenvolvidas durante o processo de construção por funcionalidade, tornando-se então o produto final, o SAG 2.0.

2.5 TESTES FUNCIONAIS

Teste funcional é uma técnica utilizada para se projetarem casos de teste, na qual são fornecidas entradas no sistema e avaliadas as saídas geradas para verificar se estão em conformidade com os objetivos especificados. De acordo com Delamaro (2013), o teste funcional pode detectar todos os defeitos, submetendo o programa ou sistema a todas as possíveis entradas, o que é denominado teste exaustivo. Nessa técnica, os detalhes da implementação não são considerados e o software é avaliado segundo o ponto de vista do usuário.

Segundo Fournier (1994), podem ser definidas duas categorias genéricas de defeitos: erros e ambiguidades. Erros são falhas no software de aplicação que farão o sistema falhar ou gerar resultados inválidos. Daí, é importante identificar o máximo possível de erros durante o ciclo de testes e eliminá-los antes de entregar o sistema a seus clientes. Para conseguir isso, os testes devem ser vistos como um processo destrutivo, que é executado com a intenção de encontrar erros.

De acordo com Pressman (1995), a atividade de teste é um elemento crítico da garantia de qualidade de software e representa a última revisão de especificação, projeto e codificação. A atividade de teste consiste de uma análise dinâmica do produto e é uma atividade relevante para a identificação e eliminação de erros que persistem. Do ponto de vista de qualidade do processo, o teste sistemático é uma atividade fundamental para garantia de qualidade.

Segundo Maldonado (1997), o processo de software possui três metas principais:

a) verificar que o software executa como especificado na documentação do projeto;

b) verificar que o software satisfaz todos os requisitos especificados na Especificação de Requisitos de Software;

c) fornecer um status do progresso do projeto ao gerente do projeto.

Durante esse estágio pode ser utilizado vários métodos e técnicas de teste, sendo alguns como: teste de unidade, teste de integração, teste de sistema e teste de aceitação.

Segundo Mueller (1995), teste é o processo de garantir que um programa se adapta aos requisitos da especificação e funciona em todos os casos em que se espera que funcione. Para isso o teste consiste em:

a) selecionar um conjunto de dados de entrada com os quais se executará o programa;

b) determinar a saída que se espera ser reproduzida;

c) executar o programa;

d) analisar os resultados produzidos pela execução do programa.

Segundo Kolm (2001), um produto de software pode ser testado utilizando-se algumas técnicas, sendo duas as mais usadas: quando se conhece as funções que foram especificadas para o software executar, o teste pode ser conduzido para demonstrar a operacionalidade das funções; e quando se conhece a estrutura interna do software, o teste pode ser conduzido para verificar se todos os componentes internos, quando exercitados, operam de maneira adequada. Essas duas técnicas são conhecidas, respectivamente, como Teste Caixa Preta ou Funcional e Teste Caixa Branca ou Estrutural.

Segundo Pressman (1995), os métodos do Teste Caixa-Preta ou Funcional concentram-se nos requisitos funcionais do software. Ou seja, esse teste possibilita que o engenheiro de software derive conjuntos de condições de entrada que exercitem completamente todos os requisitos funcionais para um programa. O teste de caixa preta procura descobrir erros nas seguintes categorias: (1) funções incorretas ou ausentes; (2) erros de interface; (3) erros nas estruturas de dados ou no acesso a bancos de dados externos; (4) erros de desempenho; e (5) erros de inicialização e término.

Teste Caixa-Branca ou Estrutural segundo Pressman (1995), é um método de projeto

de casos de teste que usa a estrutura de controle do projeto procedimental para derivar casos de teste. Usando métodos de teste de caixa branca, o engenheiro de software pode derivar os casos de teste que (1) garantam que todos os caminhos independentes dentro de um módulo tenham sido exercitados pelo menos uma vez; (2) exercitem todas as decisões lógicas para valores falsos ou verdadeiros; (3) executem todos os laços em suas fronteiras e dentro de seus limites operacionais; e (4) exercitem as estruturas de dados internas para garantir a sua validade.

De acordo com Mueller (1998), o teste funcional também é conhecido como teste caixa preta pelo fato de tratar o software como uma caixa cujo conteúdo é desconhecido e da qual só é possível visualizar o lado externo, ou seja, os dados de entrada fornecidos e as respostas produzidas como saída. Na técnica de teste funcional são verificadas as funções do sistema sem se preocupar com os detalhes de implementação.

O Teste Caixa-Preta ou Funcional será o utilizado neste trabalho, pois possibilitará a verificação das saídas dos dados utilizando entradas de vários tipos, levando a descoberta de erros conforme citado anteriormente e a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar.

3 METODOLOGIA

Para o desenvolvimento deste trabalho, foram utilizados os seguintes materiais e tecnologias:

● GoJS: segundo o site oficial (2020), o GoJS é uma biblioteca JavaScript para criação de diagramas e gráficos interativos. Permite criar todos os tipos de diagramas e gráficos, desde simples fluxogramas e organogramas a diagramas industriais altamente específicos e diagramas médicos, como é o caso dos genogramas. Ele foi utilizado para a montagem e manipulação dos elementos do genograma.

● Vue.js: de acordo com o site oficial (2020), o Vue.js é um *framework* progressivo para a construção de interfaces de usuário, o qual foi projetado para ser adotável incrementalmente, sendo fácil sua adoção e integração com outras bibliotecas ou projetos existentes. Ele foi utilizado para construir a interface gráfica do projeto.

● BootstrapVue: de acordo com o site oficial (2020), o BoostrapVue é uma estrutura progressiva para a construção de interfaces de usuário, no qual foi projetado para ser adotável incrementalmente, sendo fácil sua adoção e integração com outras bibliotecas. Ele foi utilizado para construir a interface gráfica do projeto.

● JavaScript: de acordo com Getting Started (2020), JavaScript controla o comportamento do HTML e do CSS. É uma linguagem de programação caracterizada como dinâmica. É uma das três principais tecnologias da *World Wide Web* (WWW), permitindo páginas web interativas. Muitos sites usam JavaScript, e os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo. Ela foi utilizada no desenvolvimento do front-end, juntamente com o BootstrapVue criando animações e manipulação de eventos.

● Visual Studio Code: segundo o site oficial (2020), o Visual Studio Code é um editor de código-fonte leve, mas poderoso. É uma ferramenta multiplataforma, que está disponível tanto para Windows, quanto para Mac OS e Linux, e ainda possui suporte à sintaxe de diversas linguagens como Python, Ruby, C++ e JavaScript. Ele foi utilizado como ambiente de desenvolvimento do projeto.

Inicialmente foi realizado um estudo sobre representações de gênero e relações familiares. por meio deste estudo realizado, foi possível a realização das atividades relacionadas à manutenção adaptativa e corretiva e à implementação de novas representações de gênero e representações familiares, conforme ilustrado na Figura 5.

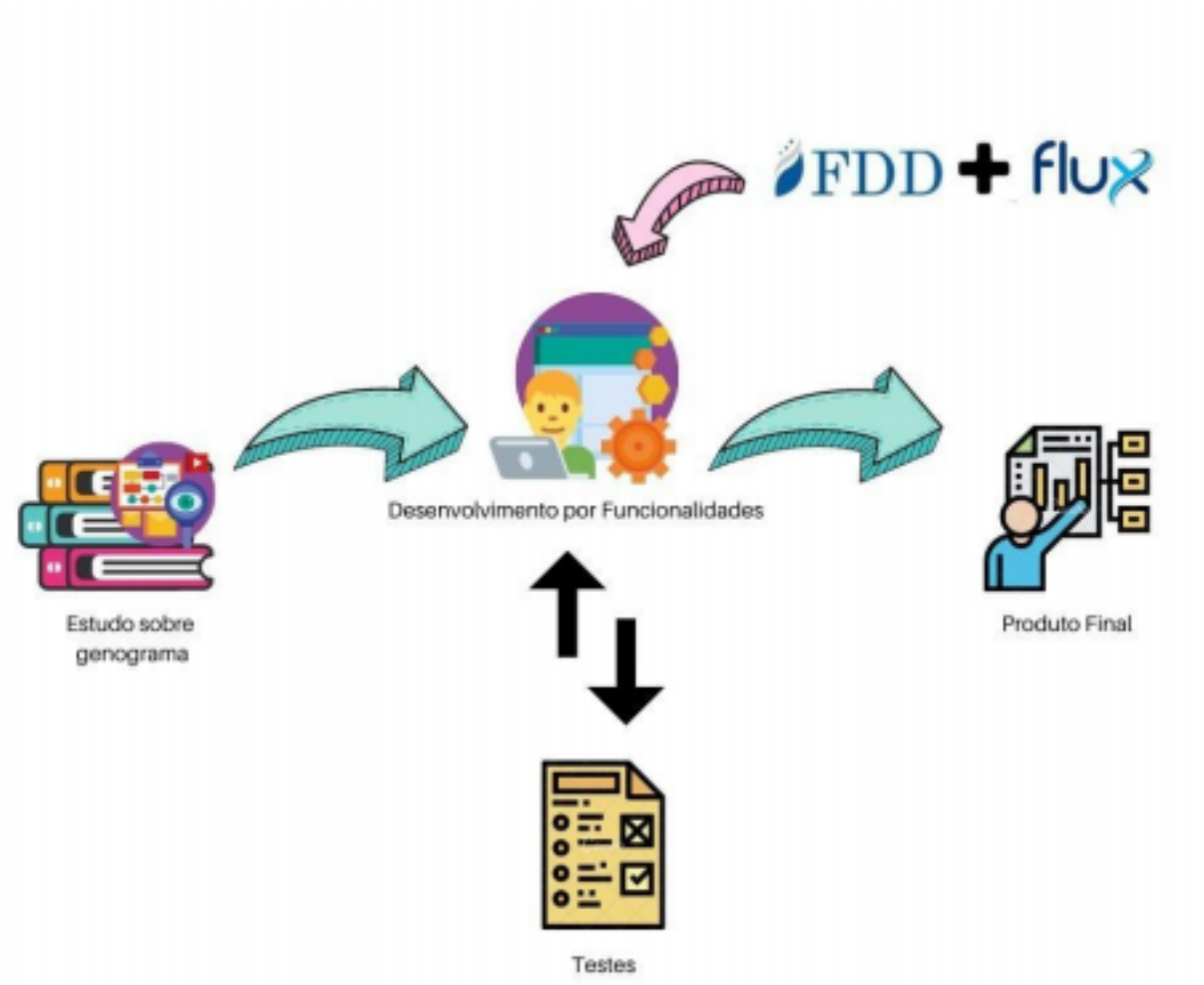

Figura 5: Metodologia .

1. Estudo sobre o genograma

Esta etapa teve por objetivo obter uma visão geral dos conceitos sobre relacionamentos familiares e sobre o genograma, como: origem, símbolos e sua utilidade para os profissionais que trabalham com famílias.

2. Desenvolvimento de novas funcionalidades para criação dos genogramas

Esta etapa abrange o desenvolvimento das novas funcionalidades no SAG 1.0, de forma que foram desenvolvidas as seguintes funcionalidades.

Inserir e excluir os símbolos:

- Símbolo para representação de animal de estimação
- Símbolo para representação de Terapia ou ligação a outras instituições.
- Símbolo para representação de Segredo da Família.
- Símbolo para representação de Imigração.

Inserir e excluir as relações conjugais:

- Ligação entre símbolos para representar Caso de Amor secreto.
- Ligação entre símbolos para representar Relação Sexual / Vivendo Juntos.
- Ligação entre símbolos para representar Reconciliação Conjugal após separação.
- Ligação entre símbolos para representar Reconciliação após divórcio.
- Ligação entre símbolos para representar Divórcio e Recasamento.

Inserir e Excluir as relações entre familiares:

- Ligação entre símbolos para representar Focado em.
- Ligação entre símbolos para representar Próximo-hostil.
- Ligação entre símbolos para representar Focado negativamente.
- Ligação entre símbolos para representar Rompimento*.
- Ligação entre símbolos para representar Rompimento reparado.
- Ligação entre símbolos para representar Cuidador.
- Ligação entre símbolos para representar Conexão espiritual ou afinidade.
- Ligação entre símbolos para representar Relação positiva.
- Ligação entre símbolos para representar Abuso sexual.

As novas funcionalidades foram construídas utilizando as ferramentas e tecnologias definidas no processo de desenvolvimento. Dentro deste processo, foi utilizado o *Visual Studio Code* como ambiente de desenvolvimento, no qual foram implementadas as novas funcionalidades na interface gráfica, utilizando as tecnologias *BootstrapVue*, *JavaScript*, *Vue.js* e o *GoJS*.

3. Realização de teste durante o desenvolvimento

Esta etapa teve por objetivo a realização de testes funcionais durante e após o desenvolvimento das novas funcionalidades no SAG 1.0, buscando garantir que o software esteja apto a realizar as funções na qual foi definida para fazer.

4 RESULTADOS

Nessa seção são apresentados os resultados obtidos com o desenvolvimento deste trabalho, bem como as soluções encontradas para o cumprimento dos seus objetivos. Inicialmente serão apresentados os artefatos gerados na modelagem do sistema.

4.1 ARTEFATOS

No início do desenvolvimento do sistema, após a realização do levantamento de requisitos foram feitas as Telas que compõem o sistema. A Figura 6.2 apresenta a nova Tela Inicial do sistema, após as devidas alterações e adaptações em relação à tela inicial do SAG 1.0, para uma melhor experiência do usuário.

Figura 6.1 - Tela Inicial (Antiga)

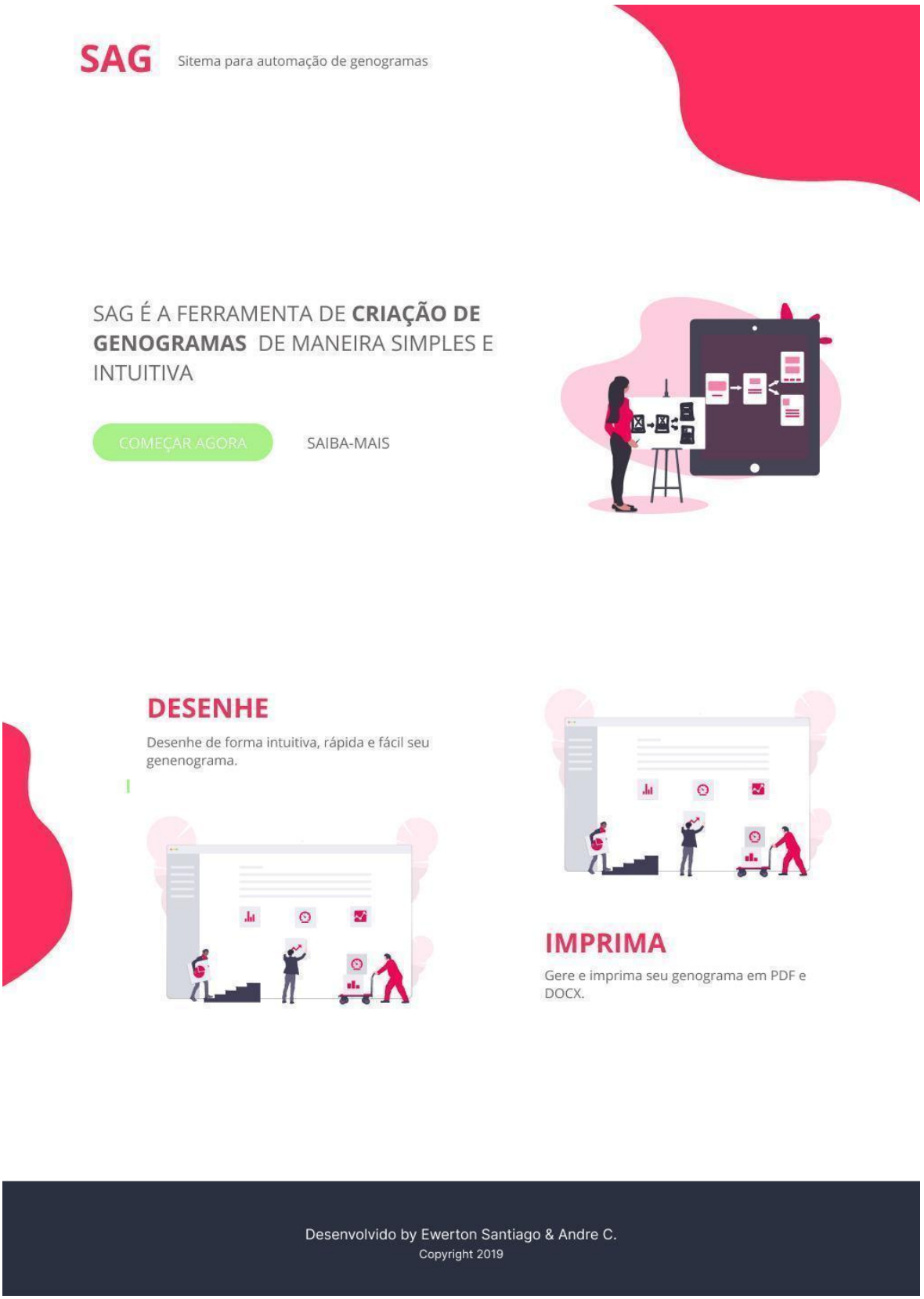

Figura 6.2 - Tela Inicial

Ao analisar a ilustração da Tela Inicial na Figura 6, é possível identificar um *layout* com padrão de cores suaves, assim como um conjunto de informações que trazem de forma objetiva o propósito do sistema aos usuários.

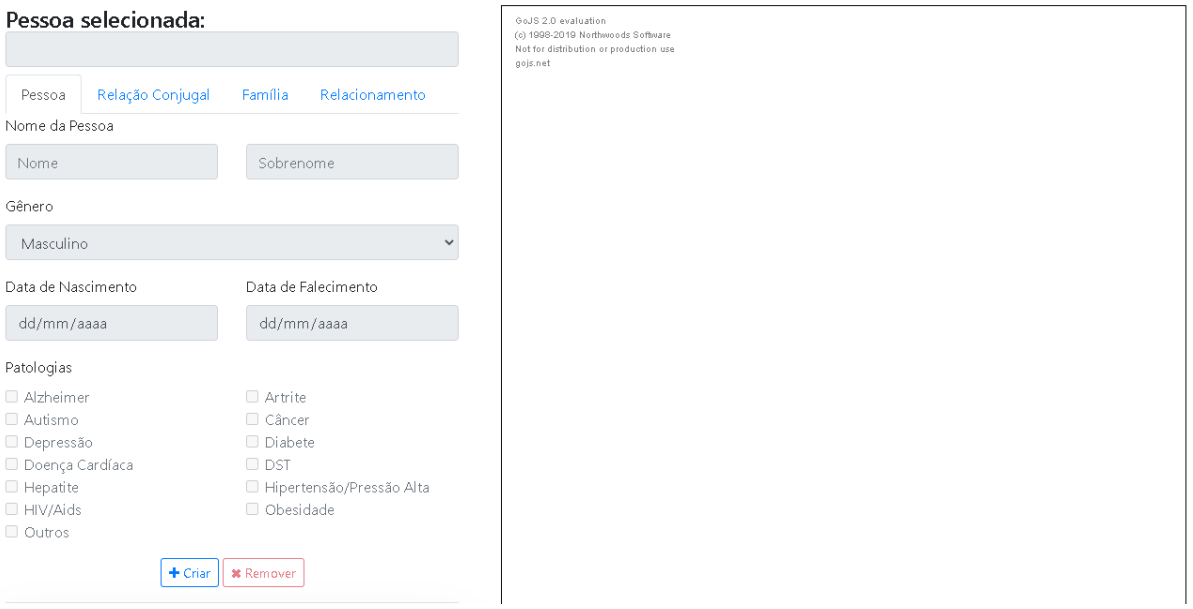

Figura 7.1 - Página de criação do genograma (Antiga)

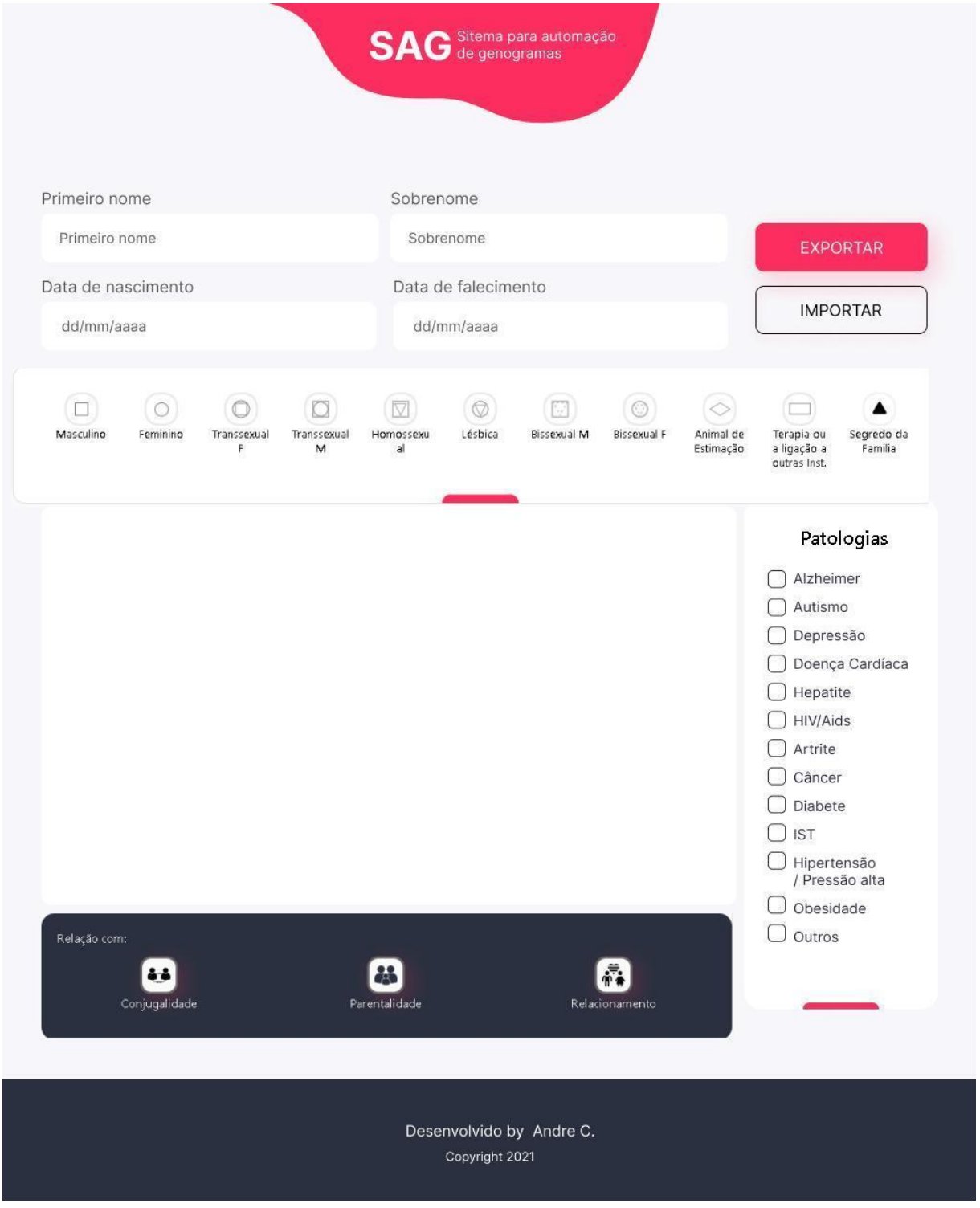

Figura 7.2 - Página de criação do genograma

A Tela principal do sistema é responsável por mostrar o desenho do genograma, porém com alterações que trazem uma melhor experiência ao usuário com relação à usabilidade no sistema web. O novo *layout* da Tela do Genograma conteve um conjunto de

novas funcionalidades, sendo elas já apresentadas na metodologia deste trabalho, como mostra a Tela de criação do genograma na Figura 7.2.

Conforme também ilustrado na Figura 7.2, o usuário tem acesso a interface do sistema por meio do navegador web. Esta interface é composta por páginas desenvolvidas em HTML 5 e estilizadas com CSS e Bootstrap. Essa página comunica-se com o Vue.js, *framework* responsável por criar interfaces web modernas; com o JavaScript, responsável por trabalhar em conjunto com o Vue.js para apresentar interfaces dinâmicas e intuitivas; e com o GoJS, biblioteca que contém as funções responsáveis pela criação do genograma. A biblioteca GoJS é executada completamente no navegador, renderizando em um elemento HTML5 sem nenhum requisito do servidor, seguindo a mesma lógica da versão SAG 1.0. Devido a utilização desta biblioteca, não foi implementado o Back-End, por não possuir lógica de negócio, sendo toda manipulação feita no Front-End.

Para construção da parte inicial da página de criação do genograma, foi inicialmente adaptado o formato existente no SAG 1.0 para que as informações a serem passadas para o objeto ficassem na parte superior. Assim como as funcionalidades “Exportar” e “Importar”. Segue a ilustração na Figura 8.

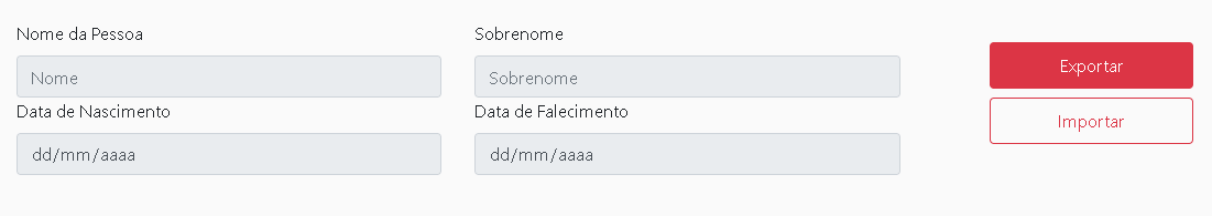

Figura 8 - Adaptação superior da página de criação do genograma.

Para adaptação dessas funcionalidades já existentes no SAG 1.0, foi criado um *Container* e subdividido dentro dele vários grupos de funcionalidades (Linha 32), sendo realizado o controle de cada funcionalidade por meio de *Inputs,* conforme mostra a Figura 9.

```
32    <div class="container">
33        <div id="controles-form">
34            <div class="row m-0 ml-0 mr-0">
35                <div class="col-md-9 col-sm-12 col-lg-9">
36                    <div class="row d-flex" style="margin: 0 auto; justify-content: center; align-items:center;">
37                        <div class="col-md-6 col-sm-12 col-lg-6">
38                            <label>Nome da Pessoa</label>
39                            <div class="input-group date">
40                                <input type="text" class="form-control" name="nome" placeholder="Nome"
41                                    :disabled="person_selected.identificacao == null" v-model="person_selected.nome">
42                            </div>
43                        </div>
```

Figura 9 - Código da adaptação superior da página de criação do genograma.

Então, foi adaptado o formato de seleção do Genograma, sendo que no SAG 1.0 era realizado por meio de Caixas de Select. Porém, para dar mais usabilidade ao Sistema Web optou-se por adaptar esse formato para seleção com botões, por meio de figuras e nomenclaturas referentes a cada gênero, conforme a ilustração da Figura 10.

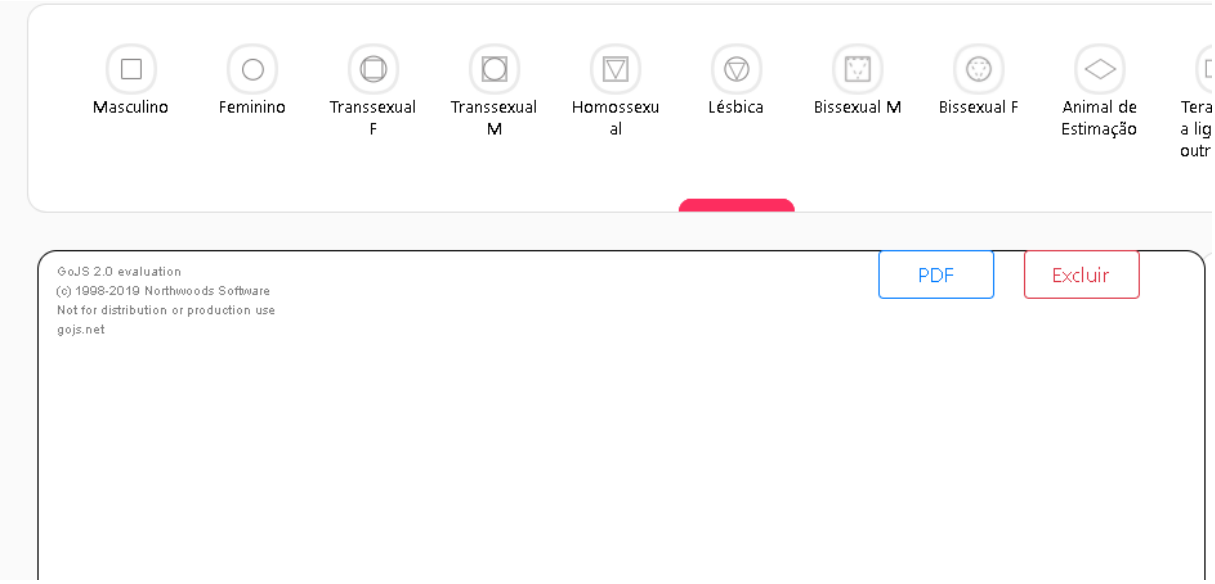

Figura 10 - Adaptação do desenho do genograma.

Para adaptação dessas funcionalidades foi criado uma *Div* e subdividido dentro dela vários grupos de funcionalidades (Linha 99). Sendo realizado o controle de cada funcionalidade por meio de classe do tipo *buttons,* onde se recebe uma *img* contendo as imagens com cada representação de genograma, conforme a ilustração na Figura 11.

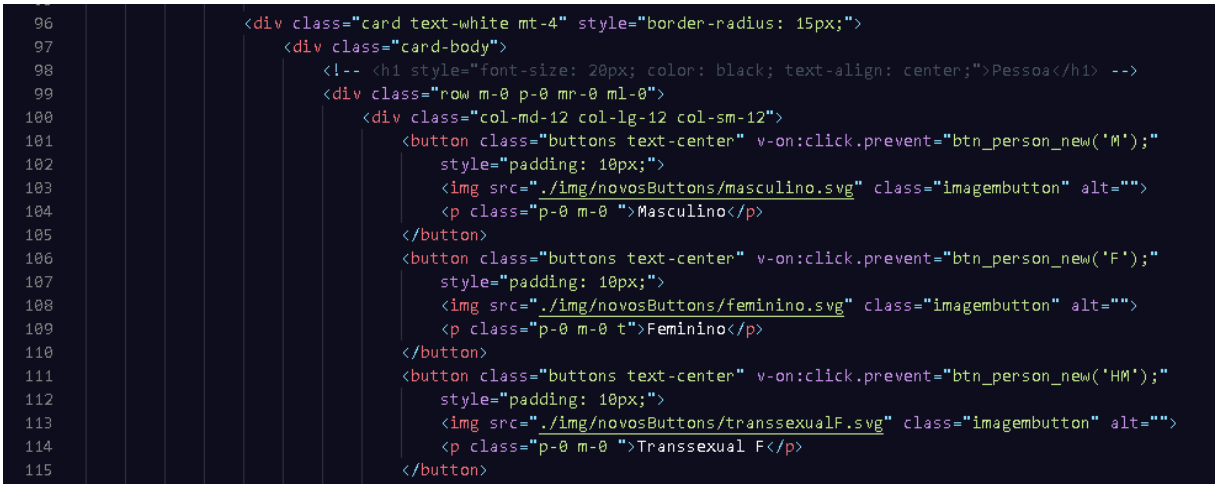

Figura 11 - Código de seleção do genograma.

Também conforme demonstrado na Figura 10 tem-se a área de desenho do genograma. Nela é utilizado um recurso já existente do GoJS, sendo passado por um *Id* em uma *Div* para a chamada desse recurso (Linha 457), conforme a ilustração na Figura 12.

Figura 12 - Código da área de desenho do genograma.

Para a seleção das patologias optou-se por continuar utilizando *checkbox,* já que o mesmo acaba sendo autoexplicativo sobre seu propósito. Porém, para uma maior usabilidade na página Web, o mesmo encontra-se na parte lateral da área de desenho do genograma, conforme a ilustração na Figura 13.

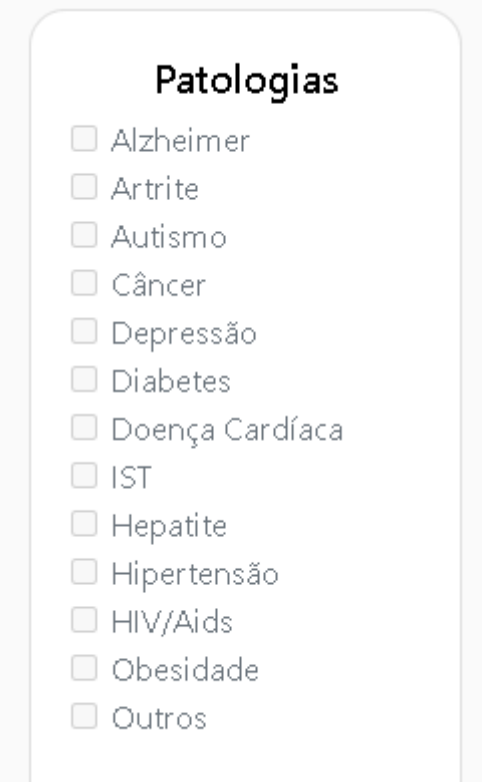

Figura 13 - Adaptação da área de seleção das patologias.

O desenvolvimento da área de seleção de patologias permaneceu com o mesmo formato apresentado no SAG 1.0, onde é realizada a chamada de uma listagem contendo as patologias. Essa execução ocorre por meio de um *v-for* (Linha 233) que traz uma listagem de objetos para a marcação do mesmo por meio de um *checkbox,* sendo então inferida a patologia, conforme ilustrado da Figura 14.

```
233    <div class="form-check" v-for="item in patologias">
234        <input class="form-check-input" type="checkbox" :value="item[0]"
235            :disabled="person_selected.identificacao == null"
236            v-model="person_selected_patologias_sorted">
237        <label class="form-check-label">{{ item[1] }}</label>
238    </div>
```

Figura 14 - Código da área de seleção das patologias.

Para o desenvolvimento no bloco de seleção das relações (relação familiar, relacionamento conjugal e relacionamento entre familiares), optou-se por utilizar o formato de seleção de botões, com figuras e nomenclaturas referentes a cada tipo de relação, conforme ilustrado na Figura 15.

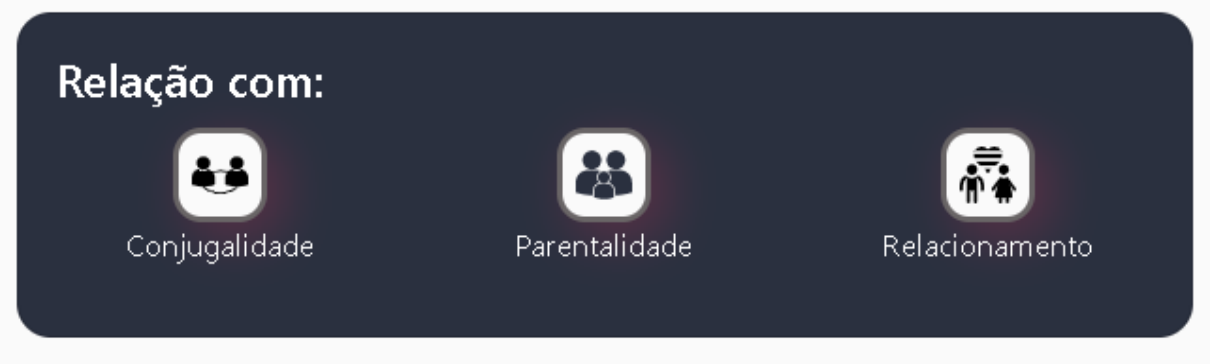

Figura 15 - Adaptação da seleção de relações.

O desenvolvimento da área de seleção de relações foi trabalhado da seguinte forma: foi criado um *Container* e foi subdividido dentro dele vários grupos de funcionalidades. Para criação do relacionamento familiar, basta selecionar uma pessoa na área de apresentação e em seguida clicar em "Relacionamento", logo em seguida serão listados os tipos de relacionamento entre familiares. No primeiro campo são listadas todas as pessoas do genograma, e no segundo campo os tipos de relacionamentos entre familiares, conforme ilustrado na Figura 16.

Figura 16 - Adaptação do *Model* de seleção de relações

Sendo realizado o controle de cada funcionalidade por meio da *class* do tipo *buttons,* ocorre o recebimento de uma *img* contendo as imagens com cada representação de relações. Após a seleção de um tipo de relação, será aberto um *Modal* para especificação desse tipo de relação (Linha 176), conforme ilustrado na Figura 16.

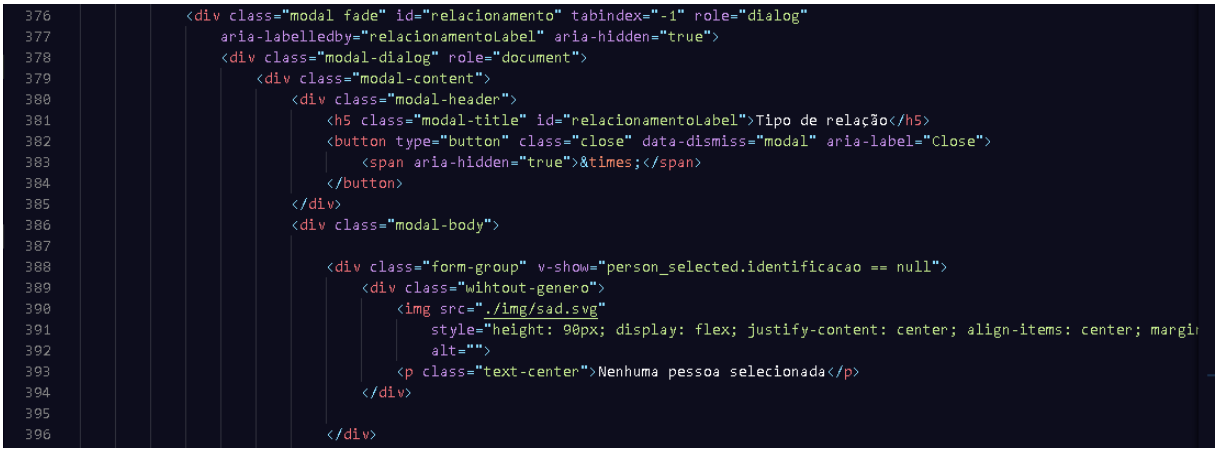

Figura 17 - Código da seleção de relações.

O desenvolvimento do bloco de seleção das relações utiliza-se de um formato para seleção de botões, onde a seleção ocorre por meio de uma *img* passando um *src* para a chamada da imagem, é realizado a chamada de *Modal* que exibe Caixas de *Select* para especificação do relacionamento a ser atribuído no objeto selecionado (Linha 380). Conforme a ilustração na Figura 17.

```
<div class="modal fade" id="relacionamento" tabindex="-1" role="dialog"
    aria-labelledby="relacionamentoLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="relacionamentoLabel">Tipo de relação</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">

                <div class="form-group" v-show="person_selected.identificacao == null">
                    <div class="wihtout-genero">
                        <img src="./img/sad.svg"
                            style="height: 90px; display: flex; justify-content: center; align-items: center; margi
                            alt="">
                        <p class="text-center">Nenhuma pessoa selecionada</p>
                    </div>

                </div>

                <div v-show="person_selected.identificacao != null">
                    <div class="form-group" v-for="(item, idx) in person_selected.relacao">
                        <label>Relacionamento Entre Familiares</label>
                        <div class="row">
                            <div class="col">
                                <select class="form-control" v-model="item.com">
                                    <option
                                        v-for="disponible_person in disponible_person_for_relation(item.com)"
                                        :value="disponible_person.identificacao">{{
                                        disponible_person.nome }}
                                    </option>
                                </select>
```

Figura 18 - Código do *Modal* da seleção de relações.

O *Modal* após exibido, traz uma listagem dos tipos de relações dentro do relacionamento selecionado, onde cada relacionamento tem uma listagem com suas respectivas relações. O usuário realiza então “por meio das caixas de seleções” a seleção da relação a ser atribuída ao objeto selecionado. Conforme ilustrado na Figura 16.

4.2 IMPLEMENTAÇÃO DE NOVOS ARTEFATOS

Um genograma tem como principal recurso a capacidade de representar graficamente os indivíduos de uma família. Para realizar essa representação, são usados símbolos que descrevem cada membro da família e sua estrutura básica. Tendo isso em vista, uma das novas implementações foi a representação do Animal de Estimação, sendo que para representá-la, basta utilizar um Losango, conforme apresentado na Figura 19.

Figura 19 - Figura representativa para Animal de Estimação.

Outra nova implementação foi a representação da Terapia ou a ligação a outras Instituições, sendo que para representá-la, basta utilizar um Retângulo, conforme apresentado na Figura 20.

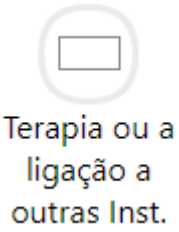

Figura 20 - Figura representativa para Terapia ou a ligação a outras Instituições.

No desenvolvimento dessas funcionalidades, assim como em outras, foi inserido no genograma o método "`diagram.nodeTemplateMap.add`" que faz parte da biblioteca GoJS (Linha 142), o qual é chamado enviando as informações definidas na função para a biblioteca, que em seguida processa essas informações e apresenta o símbolo na área de desenho do genograma. A função é composta pelo identificador do símbolo, local de posicionamento na área de apresentação, que é definida através do "go.Node", formato do símbolo definida pelo "go.Panel" e local de apresentação das patologias, através do "new go.Binding", e no final da função o "go.TextBlock" que define o local e posicionamento do nome da pessoa. Ilustração na Figura 21.

```
142             diagram.nodeTemplateMap.add(
143                 "M",
144                 $gomake(
145                     go.Node,
146                     "Vertical",
147                     {
148                         locationSpot: go.Spot.Center,
149                         locationObjectName: "ICON"
150                     },
151                     $gomake(
152                         go.Panel,
153                         {
154                             name: "ICON"
155                         },
156                         $gomake(
157                             go.Shape,
158                             "Square",
159                             {
160                                 width: 40,
161                                 height: 40,
162                                 strokeWidth: 2,
163                                 fill: "white",
164                                 portId: ""
165                             }
166                         ),
```

Figura 21 - Código da *Função* para criação dos símbolos.

Na aba Conjugalidade, onde a mesma é representada na Figura 15, são definidos os relacionamentos conjugais entre as pessoas do genograma, onde também foi realizada a implementação de algumas novas funcionalidades, como a ligação Reconciliação Conjugal após separação e também a reconciliação após divórcio (Figuras 22 e 23).

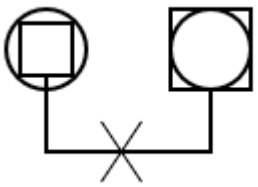

Figura 22 - Figura de ligação para representação de Reconciliação Conjugal após separação

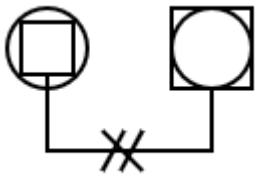

Figura 23 - Figura de ligação para representação de Reconciliação após divórcio

Para implementação das figuras de ligação foi utilizado o método “setupParents” (Linha 1753), podendo ser observado na Figura 23, que recebe as informações da função apresentada nas Figuras 22 e 23, e verifica o identificador da pessoa selecionada na área de apresentação, que será o filho, e os identificadores das pessoas selecionadas na aba de relacionamento familiar, e em seguida realiza o relacionamento entre eles através do método “addLinkData” que se encontra no final da função.

```
1753        setupParents: function () {
1754            var self = this;
1755            var model = self._diagram.model;
1756            var nodeDataArray = model.nodeDataArray;
1757
1758            for (var i = 0; i < nodeDataArray.length; i++) {
1759                var data = nodeDataArray[i];
1760                var key = data.identificacao;
1761
1762                var pais = data.pais
1763                    .filter(function (val) {
1764                        return ["", null].indexOf(val.com) === -1
1765                    })
1766                    .map(function (val) {
1767                        return val.com
1768                    });
```

Figura 24 - Código da *Função* para criação das figuras de ligação.

Na aba de Relacionamento, onde a mesma é representada na Figura 15, é definido o relacionamento entre familiares do genograma, na qual também foi realizada a implementação de algumas novas funcionalidades. Uma das novas funcionalidades foi a representação de ligação entre familiares de Focado em, assim como pode ser observado na Figura 25.

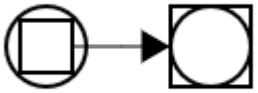

Figura 25 - Figura de ligação para para representação de Focado em

Para criação de um relacionamento familiar, basta selecionar uma pessoa na área de apresentação e em seguida clicar em “Relacionamento”. No primeiro campo são listadas todas as pessoas do genograma, e no segundo campo os tipos de relacionamentos entre familiares. Em seguida ocorre o processo do método “setupParents” já descrito anteriormente.

Assim como a implementação das novas funcionalidades descritas anteriormente, foi realizada a criação de algumas outras, que podem ser encontradas nos Apêndices deste trabalho.

Após a implementação das novas funcionalidades e alterações no layout do sistema, foi possível realizar a criação do Genograma, conforme o exemplo ilustrado na Figura 26.

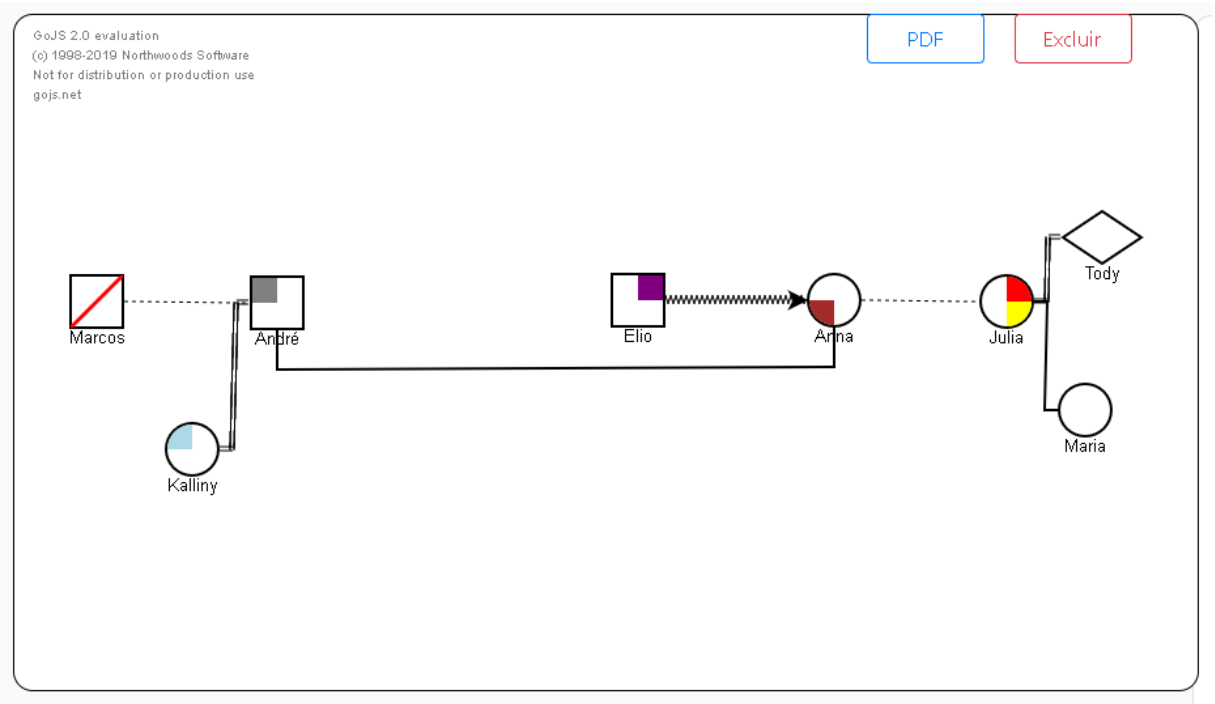

Figura 26 - Tela com exemplo de genograma

A Figura 26 mostra um exemplo de uma relação familiar, na qual, por meio do desenho do genograma, é possível identificar relações próximas de uma família de quatros pessoas e um animal de estimação, sendo que um dos membros já é falecido. No Genograma é possível identificar relações distantes, próximas, positivas e de Abuso Sexual entre os membros familiares. Assim como também a presença de patologias existentes Com base na versão do SAG 2.0 é possível realizar a criação de símbolos, padrões e a representação da história familiar em um novo padrão de layout, tendo a possibilidade de novas funcionalidades e com uma melhor experiência de usuário.

4.3 CASOS DE TESTES

Para o mapeamento dos casos de testes foi realizada primeiramente um quadro identificando todas as correções necessárias, assim como seus impedimentos até a solução a ser realizada. Para a criação da tabela, teve-se como base a versão do SAG 1.0, conforme representado na Figura 7.1, sendo que por meio desta foi pensada em uma solução viável para ser realizada em cada requisito identificado.

N°	Correções Identificadas	Simulação	Complicações	Observações	Solução
C01	Liberação de caixas de seleção (Aba Pessoa)	Ao acessar o "Entrar" no sistema, selecionar a aba "Pessoa" para poder selecionar a opção "Criar". Levando então para a liberação das caixas de seleção.	Toda vez que selecionar a opção "criar", o sistema já relaciona um novo objeto.	O sistema automaticamente cria um novo objeto (símbolo) no genograma, referenciando o mesmo.	Criar Objeto e depois trabalhar suas especificações com o objeto selecionado. Liberá-la após a seleção do objeto.
C02	Auto ajuste após criar um novo Objeto (Pessoa)	Ao "Criar" um novo objeto(símbolo) ocorre um auto ajuste no painel do genograma (área de visualização das figuras | lado direto do sistema). **Obs1.** Mais perceptível quando se cria 2 ou mais símbolos. **Obs2.** É necessário movimentar o objeto para	Devido ao padrão de estrutura utilizado do SAG 1.0 de múltiplas camadas, pode não ser possível a correção.	O sistema automaticamente ajusta todos os objetos (centraliza).Porém isso só ocorre quando altera-se o local de origem do objeto e cria-se um novo.	Criar um novo objeto não atualizando todo o painel do genograma.

		uma área qualquer do painel.			
C03	Liberação de caixas de seleção (Aba Relação Conjugal)	Ao acessar o "Entrar" no sistema, selecionar a aba "Relação Conjugal". É necessário selecionar a opção "Criar relacionamento conjugal" para liberar as caixas de seleção.	Toda vez que selecionar a opção "Criar relacionamento conjugal", o sistema vai relacionar o objeto atual com o que deseja criar o relacionamento. Para isso é necessário estar selecionando um objeto, para as caixas de select representarem com quem e que tipo vai ser esse relacionamento.	O sistema automaticamente cria um relacionamento (símbolo) no genograma, referenciando o objeto selecionado com o objeto informado nas caixas de seleção.	Utilizar figuras (símbolos) para representarem esse relacionamento, podendo após selecionar o símbolo apenas ligar quem com quem será o relacionamento.
C04	Liberação de caixas de seleção (Aba Família)	Ao acessar o (Entrar) no sistema, selecionar a aba "Familia". É necessário selecionar a opção "Criar relacionamento Familiar" para liberar as caixas de seleção.	Toda vez que selecionar a opção "Criar relacionamento familiar", o sistema vai relacionar o objeto atual com o que deseja criar o relacionamento. Para isso é necessário estar selecionando um objeto, para as caixas de select representarem com quem e que tipo vai ser esse relacionamento.	O sistema automaticamente cria um relacionamento (símbolo) no genograma, referenciando o objeto selecionado com o objeto informado nas caixas de seleção.	Utilizar figuras (símbolos) para representarem esse relacionamento, podendo após selecionar o símbolo apenas ligar quem com quem será o relacionamento.
C05	Liberação de caixas de seleção (Aba Relacionamento)	Ao acessar o (Entrar) no sistema, selecionar a aba "Relacionamento". É necessário selecionar a opção "Criar relacionamento entre familiares" para liberar as caixas de seleção.	Toda vez que selecionar a opção "Criar relacionamento entre familiares", o sistema vai relacionar o objeto atual com o que deseja criar o relacionamento. Para isso é necessário estar selecionando um	O sistema automaticamente cria um relacionamento (símbolo) no genograma, referenciando o objeto selecionado com o objeto informado nas caixas de seleção.	Utilizar figuras (símbolos) para representarem esse relacionamento, podendo após selecionar o símbolo apenas ligar quem com quem será o relacionamento.

			objeto, para as caixas de select representarem com quem e que tipo vai ser esse relacionamento.		
C06	Bug na ligação do genograma após selecionar apenas um campo(Aba Relação Conjugal, Família, Relacionamento).	Selecionar a aba "Relação Conjugal e etc" ao selecionar a opção "Criar relacionamento conjugal" quando seleciona-se uma das duas caixas de select, acontece um bug visual no painel, podendo ser observado no relacionamento entre os dois símbolos em questão.	O motivo disso é que o sistema necessita de duas informações, sendo elas: com quem vai ser o relacionamento, e que tipo de relacionamento, porém essa ligação visual já ocorre após selecionar apenas com quem vai ser o relacionamento., Não necessitando do tipo de relacionamento para completar a ligação.	Não afeta o desempenho ou funcionalidade do sistema, apenas é algo evitável.	Utilizar figuras (símbolos) para representarem esse relacionamento, podendo após selecionar o símbolo apenas ligar quem com quem será o relacionamento, ou seja esse processo atual que ocorre nas duas caixas de select fica transparente.
C07	Bug Visual nos relacionamentos	Ao manipular(mover) os objetos(símbolos) quando se tem relacionamentos formados com dois ou mais objetos, os relacionamentos também são alterados.	O motivo disso é que o sistema acaba tentando ajustar a posição ou forma dos relacionamentos de acordo com a posição do objeto, que está sendo manipulado (posição).	Não afeta o desempenho ou funcionalidade do sistema, apenas é algo evitável que causa um bug visual.	Relacionar o objeto não atualizando todo o painel do genograma.

Quadro 1 - Mapeamento de correções

Após a especificação do quadro, foram elaborados casos de teste referentes a cada requisito apresentado, tendo em vista a implementação da solução proposta. O Caso de teste tem como objetivo a validação da solução proposta. Ilustração do Caso de Teste C02 no quadro 2.

Caso de Teste	**C02**

Pré-condições	Página do Genograma carregada
Procedimentos	1. Usuário cria um objeto “Pessoa”. 2. Usuário move o objeto “Pessoa” na área de desenho. 3. Usuário cria um novo objeto “Pessoa”.
Resultados esperados	• O novo objeto “Pessoa” é criado sem afetar o objeto existente.
Dados de entrada	N/A
Prioridade	Alta
Técnica	Manual
Iteração	Não

Quadro 2 - Caso de Teste C02

Após a criação dos casos de teste, foi realizada a execução e validação dos testes propostos, sendo possível identificar se as soluções propostas foram validadas de acordo com os resultados esperados. Assim como o que se pode observar no Quadro 3.

Executor	Pré Requisitos	Caso de Teste	Resultado Esperado	Nº de Execuções	Status da Execução	Data Execução	Observações
André Costa	Formulário Carregado	C01	O novo objeto “Pessoa” é criado e posteriormente os campos Nome, Sobrenome e etc. são liberados para trabalhar as especificações do objeto selecionado.	1	Sucesso	08/06/2021	N/A
André Costa	Formulário Carregado	C02	O novo objeto “Pessoa” é criado sem afetar o objeto existente.	1	Falhou	08/06/2021	A estrutura de múltiplas camadas que está sendo ultilizada, não é suportada na biblioteca GoJs.

André Costa	Formulário Carregado	C03	O novo objeto “Pessoa” é criado e posteriormente a opção “Conjugalidade” fica disponível para utilização.	1	Sucesso	08/06/2021	N/A
André Costa	Formulário Carregado	C04	O novo objeto “Pessoa” é criado e posteriormente a opção "Parentalidade" fica disponível para utilização.	1	Sucesso	08/06/2021	N/A
André Costa	Formulário Carregado	C05	O novo objeto “Pessoa” é criado e posteriormente a opção “Relacionamento” fica disponível para utilização.	1	Sucesso	08/06/2021	N/A
André Costa	Formulário Carregado	C06	A ligação entre os objetos é criada automaticamente.	1	Sucesso	08/06/2021	N/A

Quadro 3 - Resultado dos casos de Testes.

Com base no resultado dos casos de testes pode-se observar uma falha ocorrida apenas no Caso de Teste C02, em que devido a utilização de uma estrutura de múltiplas camadas que não é suportada na biblioteca GoJs, acaba impossibilitando a correção para a solução proposta no mapeamento das correções. Isso ocorre devido ao motivo de que ao criar um novo nó, toda a árvore, sendo a mesma o conjunto de objetos utilizados para compor o genograma é reestruturada e forçada a ir ao centro, o erro é causado devido uma inconsistência na estruturação que foi criada no código anteriormente, que força com que cada nó seja sempre reordenado ao centro da página.

O restante dos casos de testes, assim como as novas funcionalidades descritas anteriormente, podem ser encontrados nos Apêndices deste trabalho.

Nesta seção foram apresentados parte dos resultados obtidos no desenvolvimento do SAG 2.0, exibindo as novas funcionalidades implementadas no sistema e os respectivos testes para as correções propostas. No capítulo seguinte são apresentadas as considerações finais

sobre o processo de desenvolvimento do SAG 2.0 e expectativas para possíveis trabalhos futuros, em continuação deste sistema. Assim como também as dificuldades e impedimentos encontrados.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo a correção e adaptação do SAG 1.0, assim como o desenvolvimento de novas funcionalidades. O desenvolvimento do sistema teve como motivação a necessidade de adequação do *software* para que esse pudesse ser utilizável por profissionais e estudantes, tornando possível mapear a evolução familiar através da criação de genogramas.

Para que fosse alcançado o propósito final, foram adotados processos para a adequação e desenvolvimento do software, iniciando-se o estudo sobre os métodos de manutenção corretiva e adaptativa do software, assim como também as relações familiares e sobre o genograma.

Foi utilizado como objeto de estudo do genograma, seus símbolos, padrões e a representação da história familiar, para que se alcançasse o desenvolvimento das novas

funcionalidades, utilizando ferramentas e tecnologias que permitissem criar uma interface responsiva e intuitiva para a construção de genogramas.

Como resultado, obteve-se o SAG 2.0 - Sistema para Automação de Genogramas, que em sua versão atual pode ser utilizado por acadêmicos e profissionais de Psicologia. No entanto, durante o processo de testes e correção, foi identificado a impossibilidade de correção para uma das correções propostas. Esse impedimento pode ocasionar em uma má experiência do usuário, podendo ser considerado um impedimento para utilização do sistema.

Foram realizadas pesquisas em sites e fóruns da própria biblioteca, e tendo em vista as respostas dos usuários e de especialistas na linguagem, para correção do problema teria que ser refeito a forma de criação do genograma, já que para a criação de cada nó, é realizado uma atualização automática. Conforme especificado no Caso de Teste C02 demonstrado nos resultados.

Para trabalhos futuros, uma reestruturação do sistema seria viável para correção da problemática encontrada no Caso de Teste C02, trazendo então uma melhor experiência do usuário. Assim como também a inclusão de um manual de utilização do sistema e um sistema de autenticação com login e senha.

REFERÊNCIAS

ABRAHAMSSON, P.; SALO, O. **Agile software development methods – review and analysis**. Espoo: VTT Publications 478, 2002

BARROS, Gabryela Santana et al. **Análise experimental entre as técnicas TDD e Test-Last no processo de manutenção corretiva de software**. 2019. 26

BODUCH, Adam. **Flux architecture**. Packt Publishing Ltd, 2016.

BECK, K. et al. **Manifest for Agile Software Development**. Manifest for Agile Software Development, 2001.

CARTER, B., & MCGOLDRICK, M. (1995). **As mudanças no ciclo de vida familiar**. Porto Alegre, RS: Artes Médicas. (Original publicado em 1989).

BAYER, Priscila. **Manutenção preventiva sob a ótica da Indústria 4.0 - Um estudo de caso: Problemas na coleta de dados para a aplicação de análise preditiva**, São Paulo, 2019

DELAMARO, Marcio; JINO, Mario; MALDONADO, Jose. **Introdução ao teste de software**. Elsevier Brasil, 2013.

FACEBOOK INC. **Flux: Application Architecture For Building User Interfaces. Flux Documentation**, 2014.

FOURNIER, Roger. **Guia prático para o desenvolvimento e manutenção de sistemas estruturados**. São Paulo: Makron Books, 1994.

GOMES, Ramon Diogo Gondim Miaja et al. **Desenvolvimento de uma rede social utilizando Erlang e a arquitetura Flux**. 2016.

GRUBB, A. A. T. P. **Software Maintenance: Concepts And Practice**. 2. ed. [S.l.]: World Scientific, 2003. 26, 29

GAMMA, Erich et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Westford: Addison-Wesley, 2009.

HIGHSMITH, J.; COCKBURN, A. **Agile Software Development: The Business of Innovation**. 9. ed. [S.l.]: IEEE, v. 34, 2001.

HIGHSMITH, J. **Agile Software Development Ecosystems**. Addison Wesley, 2002.

JALOTE, P. (1997), **An integrated approach to software engineering**, Springer, 2a.edição.

KOLM, Everton L. **Sistema para gerenciamento de testes funcionais de software**. 2001.

RICARDO, José R. **FDD (Feature Driven Development)**. 2014. https://medium.com/@jrobaski/fdd-feature-driven-development-7d08c5c24c8f.

LIENTZ, B. P.; SWANSON, E. B.; TOMPKINS, G. E. **Characteristics of application software maintenance**. Commun. ACM, ACM, New York, NY, USA, v. 21, n. 6, p. 466–471, jun. 1978. ISSN 0001-0782. Disponível em: . 26

MAMONE, S. **The ieee standard for software maintenance**. SIGSOFT Softw. Eng. Notes, ACM, New York, NY, USA, v. 19, n. 1, p. 75–76, jan. 1994.

MALL, R. **Fundamentals of Software Engineering**. 5. ed. [S.l.]: PHI Learning Pvt. Ltd.,2018. 26

MCGOLDRICK, M., & GERSON, R. (1995). **Genetogramas e o ciclo de vida familiar** (M. A. V. Veronese, Trad.). In B. Carter & M. McGoldrick, M. (Eds.), As mudanças no ciclo de vida familiar – Uma estrutura para a terapia familiar (2. ed.). Porto Alegre, RS: Artes Médicas.

MARCHETTI-MERCER, M. C., & CLEAVER, G. (2000). **Genograms and family sculpting: An aid to cross-cultural understanding in the training of psychology students in South Africa**. The Counseling Psychologist, 28(1), 61-80.

MALDONADO, José Carlos et al. **Gestão de configuração na atividade de teste.** In: Workshop Qualidade de Software, 1., 1997, Fortaleza. Anais do Workshop Qualidade de Software: SBC, 1997. p. 107-116.

MULLER, James. McClure, Carma. **Técnicas estruturadas e CASE**. São Paulo: Makron Books, 1995.

NIEWEGLOWSKI, V. H. (2004). **Unidade de terapia intensiva pediátrica: Vozes e vivências da família**. Dissertação de Mestrado não-publicada, Programa de Pós-Graduação em Psicologia, Universidade Federal de Santa Catarina, Florianópolis, SC.

NICHOLS, M. P., & SCHWARTZ, R. C. (1998). **Terapia familiar: Conceitos e métodos**. Porto Alegre, RS: Artmed.

REZENDE, D. A. **Engenharia de Software e Sistemas de Informação**. 3. ed. [S.l.]: Brasport Livros e Multimídia Ltda., 2005. 26

SHIA, Khaohun et al. **Sistema multiagente para análise de aderência e melhoria do processo de desenvolvimento ágil FDD baseado no modelo de qualidade CMMI**. 2015.

SZYMANSKI, H. (2004). **A pesquisa-intervenção participante com famílias de baixa renda: Um projeto participativo de atenção psicossocial.** In C. R. Althoff, I. Elsen & R. G. Nitschke (Eds.), Pesquisando a família: Olhares contemporâneos (pp. 115-125). Florianópolis, SC: Papa Livro.

WENDT, N. C. (2006). **Fatores de risco e de proteção para o desenvolvimento da criança durante a transição para a parentalidade**. Dissertação de Mestrado não-publicada, Programa de Pós-Graduação em Psicologia, Universidade Federal de Santa Catarina, Florianópolis, SC.

WENDT, Naiane Carvalho; CREPALDI, Maria Aparecida. **A utilização do genograma como instrumento de coleta de dados na pesquisa qualitativa**. Psicologia: Reflexão e crítica, v. 21, n. 2, p. 302-310, 2008.

APÊNDICES

APÊNDICE A - Novas funcionalidades implementadas não apresentadas nos resultados do trabalho

Uma outra nova implementação foi o Símbolo Representativo para Segredo da família, sendo que para representá-la, basta utilizar um Triângulo preenchido, conforme apresentado na Figura 24.

Figura 27 - Figura representativa para Segredo da família.

Na aba Conjugalidade, onde são definidos os relacionamentos conjugais entre as pessoas do genograma, foi realizada a implementação de algumas novas funcionalidades, sendo as mesmas listadas logo abaixo.

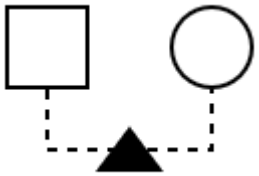

Figura 28 - Figura de ligação para Caso de Amor Secreto

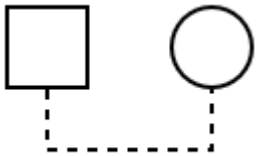

Figura 29 - Figura de ligação para Relação sexual vivendo juntos

Na aba Relacionamento onde são definidos os relacionamentos entre familiares, também foi realizada a implementação de algumas novas funcionalidades, sendo as mesmas listadas logo abaixo.

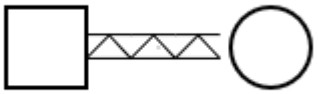

Figura 30 - Figura de ligação para Próximo-hostil

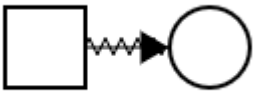

Figura 31 - Figura de ligação para Focado negativamente

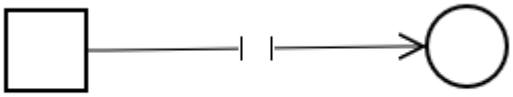

Figura 32 - Figura de ligação para Rompimento

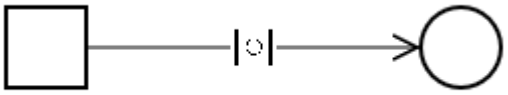

Figura 33 - Figura de ligação para Rompimento Separado

Figura 34 - Figura de ligação para Cuidador

Figura 35 - Figura de ligação para Conexão espiritual ou afinidade

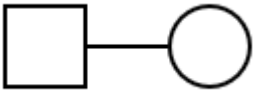

Figura 36 - Figura de ligação para Relação positiva

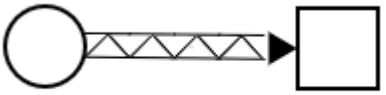

Figura 37 - Figura de ligação para Abuso Sexual

APÊNDICE B - Casos de Testes

Caso de Teste	**C01**
Pré-condições	Página do Genograma carregado
Procedimentos	1. Usuário cria um objeto “Pessoa”.
Resultados esperados	● O novo objeto “Pessoa” é criado e posteriormente os campos Nome, Sobrenome e etc. são liberados para trabalhar as especificações do objeto selecionado.
Dados de entrada	N/A
Prioridade	Alta
Técnica	Manual
Iteração	Não

Quadro 4 - Caso de Teste C01

Caso de Teste	**C03**
Pré-condições	Página do Genograma carregado
Procedimentos	1. Usuário cria um objeto “Pessoa”.
Resultados esperados	● O novo objeto “Pessoa” é criado e posteriormente a opção “Conjugalidade” fica disponível para utilização.
Dados de entrada	N/A
Prioridade	Alta
Técnica	Manual
Iteração	Não

Quadro 5 - Caso de Teste C03

Caso de Teste	**C04**
Pré-condições	Página do Genograma carregado
Procedimentos	2. Usuário cria um objeto “Pessoa”.
Resultados esperados	● O novo objeto “Pessoa” é criado e posteriormente a opção "Parentalidade" fica disponível para utilização.
Dados de entrada	N/A
Prioridade	Alta
Técnica	Manual
Iteração	Não

Quadro 6 - Caso de Teste C04

Caso de Teste	**C05**
Pré-condições	Página do Genograma carregado
Procedimentos	3. Usuário cria um objeto “Pessoa”.
Resultados esperados	● O novo objeto “Pessoa” é criado e posteriormente a opção “Relacionamento” fica disponível para utilização.
Dados de entrada	N/A
Prioridade	Alta
Técnica	Manual
Iteração	Não

Quadro 7 - Caso de Teste C05

Caso de Teste	**C06**
Pré-condições	Página do Genograma carregado
Procedimentos	4. Usuário cria mais de um objeto “Pessoa”.

	5. Usuário seleciona um tipo de relacionamento (Conjugalidade, Parentalidade ou Relacionamento) 6. Usuário vincula o relacionamento a um objeto “Pessoa” existente.
Resultados esperados	● A ligação entre os objetos é criada automaticamente.
Dados de entrada	N/A
Prioridade	Alta
Técnica	Manual
Iteração	Não

Quadro 8 - Caso de Teste C06

APÊNDICE C - Link de Acesso ao Sistema

Link de acesso: https://sag2.netlify.app/