UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CAMILA EMANUELE DE SOUZA

ATUALIZAÇÃO DO BACKEND DO SISTEMA OPEN SOCIAL CARE: MIGRANDO DA ARQUITETURA SERVERLESS PARA UMA API EM LARAVEL

GUARAPUAVA

2023

CAMILA EMANUELE DE SOUZA

ATUALIZAÇÃO DO BACKEND DO SISTEMA OPEN SOCIAL CARE: MIGRANDO DA ARQUITETURA SERVERLESS PARA UMA API EM LARAVEL

Social Care system backend update: migrating from serverless architecture to an API in Laravel

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, Campus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet.

Orientador: Prof. Dr. Andres Jessé Porfirio

Coorientador: Gustavo Vicari Duarte

GUARAPUAVA

2023

RESUMO

O Open Social Care é um sistema para cadastro de atendimentos sociais idealizado para ajudar assistentes sociais a terem uma melhor gestão do seu trabalho e ser um sistema de livre usabilidade e gratuito. O seu protótipo inicial foi desenvolvido usando uma infraestrutura em nuvem, inicialmente interessante para ser usado pelas assistentes sociais, porém limitada com relação ao uso real, além de ter sido idealizado para ser utilizado somente em instituições da execução penal. Então, este projeto propõe a substituição desta infraestrutura, visando melhorar a eficiência e escalabilidade do sistema, como também propor uma forma de generalização para que a aplicação possa atender outras instituições além da Execução Penal. A nova infraestrutura será desenvolvida a partir de uma *Application Programming Interface* (API) para ser robusta e flexível, permitindo que diferentes partes interessadas, seja instituições de Execuções Penais, organizações sem fins lucrativos ou outros prestadores de serviços sociais, acessem e gerenciem os registros de atendimento de forma mais eficaz. Visando atender cenários mais abrangentes a API é pensada para facilitar a realização de manutenções e personalizações. Assim, o desenvolvimento da API será um passo significativo para a melhoria do sistema Open Social Care, visando atender diferentes instituições de forma mais eficiente e versátil, contribuído para a realização de atendimentos sociais.

Palavras-chave: sistema para atendimento social; assistentes sociais; substituicao de infraestrutura em nuvem; desenvolvimento de api.

ABSTRACT

Open Social Care is a system for registering social services, designed to help social workers better manage their work and to be free and open to use. Its initial prototype was developed using a cloud infrastructure, which was initially interesting for use by social workers, but limited in terms of actual use, as well as being designed to be used only in penal institutions. Therefore, this project proposes replacing this infrastructure in order to improve the efficiency and scalability of the system, as well as proposing a form of generalization so that the application can serve institutions other than the Penal Service.The new infrastructure will be developed on the basis of an API to be robust and flexible, allowing different stakeholders, be they penal institutions, non-profit organizations or other social service providers, to access and manage care records more effectively. In order to address broader scenarios, the API is designed to facilitate maintenance and customization. Thus, the development of the API will be a significant step towards improving the *Open Social Care* system, aiming to serve different institutions in a more efficient and versatile way, contributing to the provision of social care.

Keywords: social service system; social workers; cloud infrastructure replacement; development api.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
API	*Application Programming Interface*
CCG	Conselho da Comunidade da Comarca de Guarapuava
CRAS	Centro de Referência de Assistência Social
CREAS	Centro de Referência Especializado de Assistência Social
ER	Entidade relacionamento
HTML	*HyperText Markup Language*
IU	Interface do usuário
NoSQL	*Not Only SQL*
PHP	*Hypertext Preprocessor*
PPLs	Pessoas Privadas de Liberdade
SDK	*Software Development Kit*
SQL	*Structured Query Language*
SUAS	Sistema Único de Assistência Social
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

Esta sessão explana uma breve descrição do atual sistema, quais seus problemas e logo propondo uma solução.

1.1 Considerações iniciais

O Conselho da Comunidade é um dos órgãos da Execução Penal, regulado pela Lei 7.210, de 11/07/1984, - Lei de Execução Penal - Estas instituições representam a real possibilidade de intervir nas relações sociais dentro e fora da prisão [...] (OLIVEIRA, 2012). O Conselho da Comunidade existe para auxiliar juízes durante a execução penal, gerenciando documentação e garantindo os direitos às Pessoas Privadas de Liberdade (PPLs), como o recebimento de roupas e sapatos, como também que os direitos humanos sejam respeitados. Os conselhos também prestam atendimentos para a emissão de antecedentes criminais, obtenção de documentação civil, busca de materiais e recursos solicitados por/para PPLs ou familiares, além de elaborar projetos para remição de pena voltado para atividades de caráter humanizado. Na cidade de Guarapuava - PR, uma das atribuições do Conselho da Comunidade é o serviço social, onde as assistentes sociais realizam atendimentos às PPLs, sendo de grande importância o registro e o armazenamento dessas informações de forma segura, eficiente, disponível e gerenciável.

Atualmente a gestão dos atendimentos realizados pelas assistentes sociais têm muitas brechas, pois são registrados em papel ou em arquivos digitais de texto. Estes procedimentos são falhos, gerando dificuldades na busca por informações, como também podendo gerar a perda de documentos. O Open Social Care é um sistema que visa auxiliar nas atividades desenvolvidas pelas assistentes sociais, provendo um melhor gerenciamento das informações coletadas nos atendimentos das assistentes sociais e eventuais outros usuários.

O sistema teve sua primeira fase de desenvolvimento na disciplina de Sistemas Distribuídos da Universidade Tecnológica Federal do Paraná (UTFPR), câmpus Guarapuava, onde foi proposta uma plataforma básica (um protótipo) apenas para verificar os requisitos do sistema, bem como se a proposta do *software* seria útil para os usuários, neste caso, às assistentes sociais. Com a aplicação finalizada e apresentada às assistentes sociais, foi possível identificar algumas melhorias para o projeto como por exemplo sobre a arquitetura do sistema.

Durante esta fase inicial do desenvolvimento, o sistema apresentou potencial para ser usado em outras instituições e, também, por um assistente social sem necessariamente pertencer a um Conselho da Comunidade. Então, como o desenvolvimento foi feito voltado somente ao Conselho da Comunidade da Comarca de Guarapuava (CCG), é proposto tornar a aplicação mais robusta para que outros perfis de usuário ou instituições possam também fazer uso.

Atualmente como o sistema segue um modelo de arquitetura *Serverless*[1], fortemente atrelado à plataforma onde ele foi construído, o que causa dificuldades de generalização[2] e restrições em relação à normas de segurança e privacidade. Diante disso, considera-se necessária uma refatoração da arquitetura do sistema, optando então por tecnologias que permitam uma gestão transparante do sistema e dos dados gerados por ele.

A nova versão do sistema proporcionará aos usuários a personalização de formulários para cadastro de atendimentos, não ficando mais acoplado a Conselhos da Comunidade. Também irá contar com cadastros de organizações, podendo atender várias instituições em um servidor, fazendo com que as assistentes sociais e demais usuários tenham um único acesso e possam participar dentro de outras entidades. Outro ponto a destacar é que a nova versão irá contar com diferentes tipos de permissões para os usuários, onde cada perfil tenha acessos a diferentes lugares da aplicação, tornando-a mais segura e menos suscetível ao erro. A alternativa a ser explorada é a construção de uma API[3] que visa fornecer os recursos atualmente disponibilizados pela arquitetura *Serverless*.

1.2 Objetivos

Esta seção apresenta os objetivos do trabalho.

1.2.1 Objetivo geral

Desenvolver uma API para o sistema Open Social Care e aplicá-la em substituição à infraestrutura proprietária usada no modelo *Serverless*.

1.2.2 Objetivos específicos

- Avaliação do *feedback* (parecer) obtido com o protótipo anterior;
- Revisão dos requisitos do software dadas as sugestões apontadas pelas assistentes sociais;
- Planejamento, na forma de diagramas e *mockups* (representação gráfica), das alterações a serem realizadas na nova versão do *software*;
- Elaboração de um modelo para o banco de dados;

[1] *Serverless*: Arquitetura sem um servidor próprio, em geral, utilizando infraestrutura em nuvem gerenciada e fornecida por empresas privadas.

[2] Generalização: Ampliar o uso do sistema para diferentes tipos de organizações, maiores detalhes podem ser observados no Capítulo 5 seção 5.2

[3] API (*Application Programming Interface*): conjunto de funções e procedimentos que permitem a integração de sistemas.

- Implementação de uma API *Restful*[4] que substitua os acessos realizados na plataforma proprietária do modelo *Serverless*;
- Implementação de testes para as funcionalidades desenvolvidas;
- Configuração de um servidor de *staging* (servidor de teste) para demonstração e coleta de *feedback* da nova versão do sistema;
- Escrita da documentação do *software*.

1.3 Justificativa

No CCG são realizados atendimentos semanalmente, gerando muitos arquivos e dados para a instituição, todos esses atendimentos tem um padrão a ser seguido e atualmente as assistentes sociais, para realizarem um atendimento, utilizam arquivos de texto ou são feitos na mão em papel sendo armazenados no *Google Drive* ou *One Drive*. Esse método de trabalho gera vários problemas, como ao levantar dados ou relatórios, estes precisam ser feitos manualmente, como também podem gerar a perda de documentos ou dificuldade em encontrar os arquivos, isso faz com que o trabalho se torne exaustivo, demorado e suscetível ao erro.

A estrutura do Open Social Care foi desenvolvida em um modelo em nuvem que não contemplava o gerenciamento de um servidor. A aplicação inicial foi construída utilizando o *Firebase* [5], considerado como um *Backend*, um modelo de serviço que oferece toda a infraestrutura voltada para o funcionamento interno do *software*, como sistemas, banco de dados, envio e recebimento de informações, armazenamento, entre outros. Isso quer dizer que o profissional não precisará desenvolver todo o sistema de forma manual, uma vez que o *Firebase* oferece esse serviço de forma mais automatizada (REMESSA, 2021).

O problema do protótipo ter sido feito dessa forma, é que fica obrigatório tudo estar centralizado na infraestrutura e nas tecnologias da *Google* (proprietária do *Firebase*), cada instituição que fosse utilizar a aplicação precisaria ter uma conta *Google* e realizar a configuração e manutenção do sistema por conta própria, gerando responsabilidade sobre o armazenamento e segurança das informações na conta pessoal de um único usuário (quem implementou e configurou o servidor do sistema). Ademais, a personalização das configurações na plataforma *Firebase* é mais limitada se comparada à executar a aplicação com servidor gerenciado por conta própria, onde a instituição pode ter amplos poderes para gerir a forma como os dados são armazenados.

[4] API *Restful*: É um padrão de arquitetura de software para requisções HTTP(*Hypertext Transfer Protocol*) e HTTPS(*Hyper Text Transfer Protocol Secure*)

[5] *Firebase*: É uma plataforma que possui uma infraestrutura de *backend* pronta para quem desenvolve aplicativos, possui banco de dados, autenticação e integração para aplicativos de celular. É um serviço fornecido pela empresa *Google*.

Outro ponto a ressaltar, é que o sistema inicial (protótipo) foi desenvolvido exclusivamente para atender à demanda do CCG, sendo limitado em relação ao uso em outras instituições que também realizam atendimentos sociais, dificultando a expansão do sistema para demais áreas que realizam atendimento social. Logo uma das melhorias propostas, é fazer um *software* que possa ser usado de forma genérica, isso é, a inclusão de cadastro de múltiplas instituições, onde também cada uma poderá realizar personalizações a fim de atender diferentes demandas em seus cadastros de atendimentos, possibilitando o uso do sistema para diferentes tipos de instituições. Afim de atender a demanda de personalizações para que o sistema atenda vários tipos de organizações, será desenvolvido um mecanismo de modelos de formulários, os questionários personalizados para a realização de atendimentos, cada organização vai poder criar o seu próprio formulário de atendimento, com os campos de perguntas próprios para cada objetivo e demanda, permitindo diferentes cenários de uso para o sistema, não ficando mais acoplado a necessidade do CCG.

1.4 Estrutura do trabalho

O trabalho está estruturado em cinco seções principais. No Capítulo 1, foi apresentado uma descrição do projeto de forma geral, apresentando também seus problemas e proposto soluções, além de os objetivos do trabalho. Em sequência no Capítulo 2 são apresentados os documentos e plataformas para gestão de atendimentos sociais, similares ao Open Social Care. No Capítulo 3, é abordado o protótipo Open Social Care, destacando as suas funcionalidades de infraestrutura. Em seguida, no Capítulo 4, é descrito os Materiais e Métodos que serão usados para o desenvolvimento das atividades. Finalmente no Capítulo 6, é concluído o trabalho mostrando os pontos importantes de desafios e soluções.

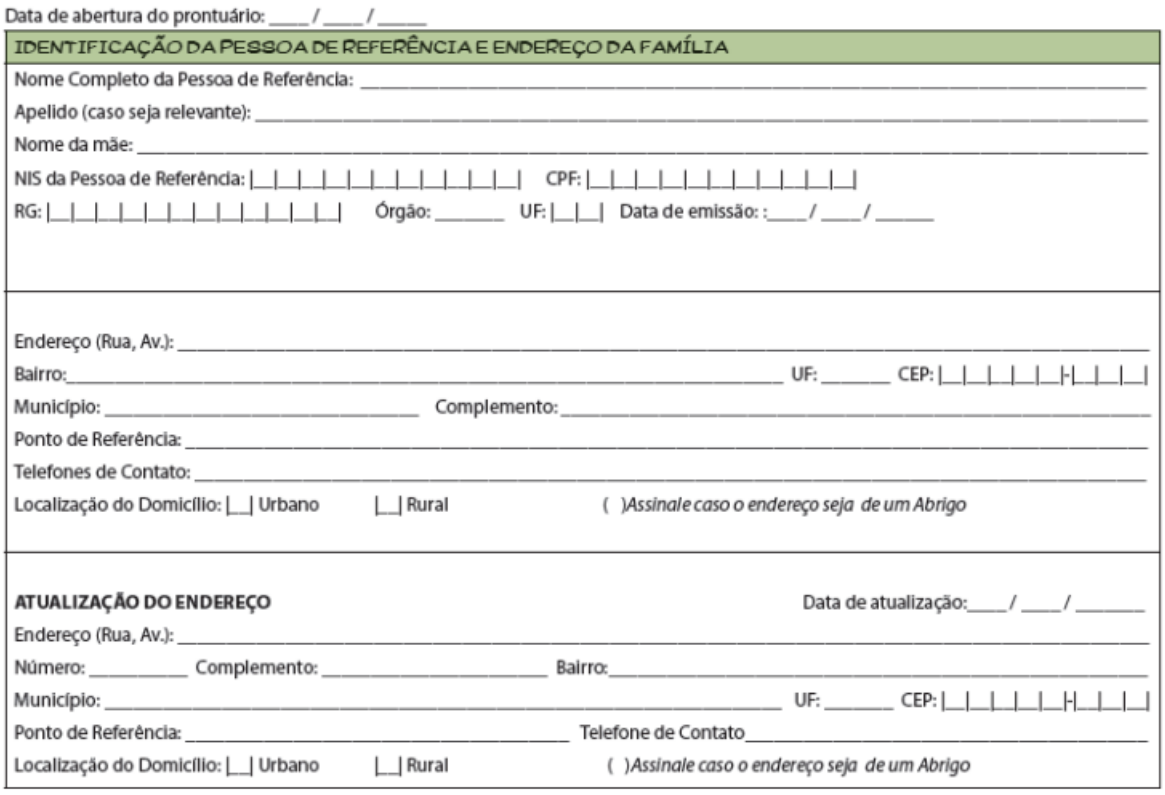
2 TRABALHOS RELACIONADOS

Nesta seção serão apresentados documentos e plataformas existentes para gerenciamento de atendimentos sociais ou são utilizadas por assistentes sociais para realização de atendimentos.

2.1 Prontuário SUAS

O Prontuário Sistema Único de Assistência Social (SUAS), é um documento utilizado em todo o Brasil para armazenar informações relacionadas à assistência social prestada a indivíduos e famílias em situação de vulnerabilidade ou risco social, na Figura 1 é possível visualizar uma versão do formulário. O Prontuário Eletrônico é uma ferramenta que auxilia o trabalho dos profissionais dos Centro de Referência de Assistência Social (CRAS), Centro de Referência Especializado de Assistência Social (CREAS) e Unidades de Acolhimento para Crianças e Adolescentes no registro dos atendimentos realizados às famílias e indivíduos, e que permite qualificar o atendimento social e analisar de forma sistematizada as informações sobre o território e a população atendida. Sua utilização permite manter um histórico dos atendimentos, agilizando assim o trabalho dos profissionais e facilitando a vida dos usuários do SUAS (SUAS, 2023).

Figura 1 – Documento Prontuário SUAS.



Data de abertura do prontuário: ____ / ____ / _____

IDENTIFICAÇÃO DA PESSOA DE REFERÊNCIA E ENDEREÇO DA FAMÍLIA

Nome Completo da Pessoa de Referência: _____

Apelido (caso seja relevante): _____

Nome da mãe: _____

NIS da Pessoa de Referência: |__|__|__|__|__|__|__|__|__|__|__| CPF: |__|__|__|__|__|__|__|__|__|__|__|

RG: |__|__|__|__|__|__|__|__|__|__|__| Órgão: _____ UF: |__|__| Data de emissão: :____/ ____/ _____

Endereço (Rua, Av.): _____

Bairro: _____ UF: _____ CEP: |__|__|__|__|__|-|__|__|__|

Município: _____ Complemento: _____

Ponto de Referência: _____

Telefones de Contato: _____

Localização do Domicílio: |__| Urbano |__| Rural ( )*Assinale caso o endereço seja de um Abrigo*

ATUALIZAÇÃO DO ENDEREÇO Data de atualização:____ / ____ / _____

Endereço (Rua, Av.): _____

Número: _____ Complemento: _____ Bairro: _____

Município: _____ UF: _____ CEP: |__|__|__|__|__|-|__|__|__|

Ponto de Referência: _____ Telefone de Contato _____

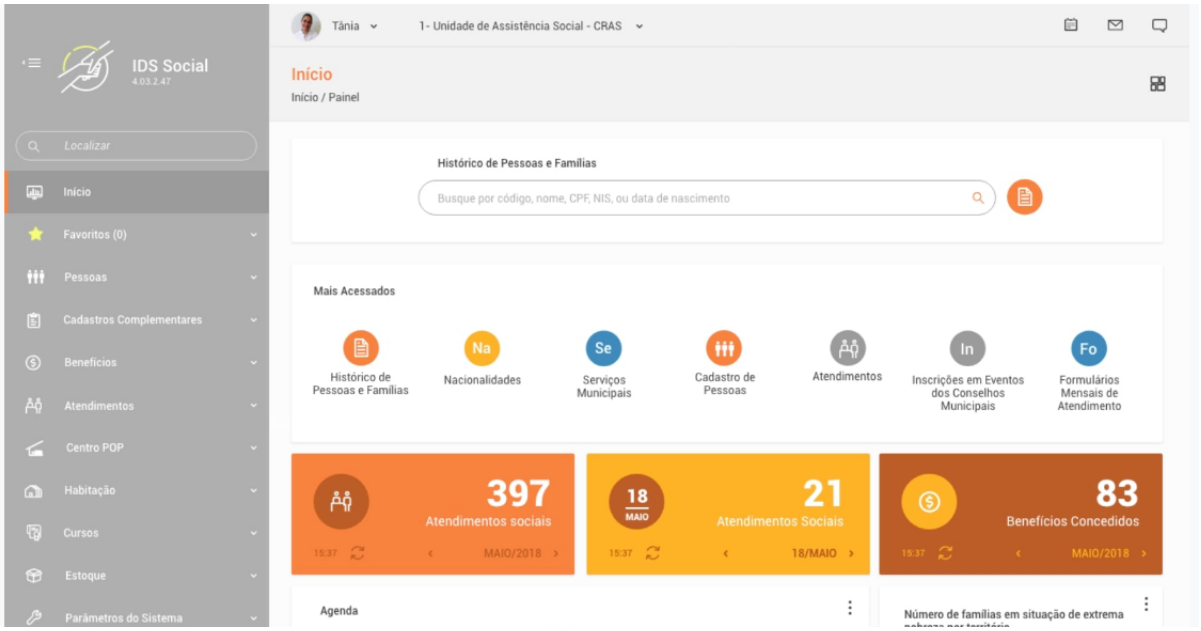
Localização do Domicílio: |__| Urbano |__| Rural ( )*Assinale caso o endereço seja de um Abrigo*

Fonte: (Ministério do desenvolvimento social e combate à fome, 2013).

2.2 IDS Social

É um *software* voltado a realização de atendimentos sociais. Ele oferece Cadastros unificados, Atendimento adequado ao Prontuário SUAS online, Informações de Atendimento e Acompanhamento Georreferenciadas, entre outras coisas. Com o IDS Social todas as necessidades e acompanhamentos da área social municipal são administradas com qualidade, segurança e centralidade de dados. O IDS Social garante integridade da base de cadastro entre setores, unidades de Assistência Social do município, CRAS e CREAS, disponibilizando um histórico único e detalhado para acompanhamento e efetividade dos planos de ações das áreas de Assistência Social municipais (IDS Social, 2023). Na Figura 2 é possível ver uma tela inicial do sistema, mostrando gráficos e algumas funcionalidades disponíveis no sistema.

Figura 2 – Tela inicial IDS Social



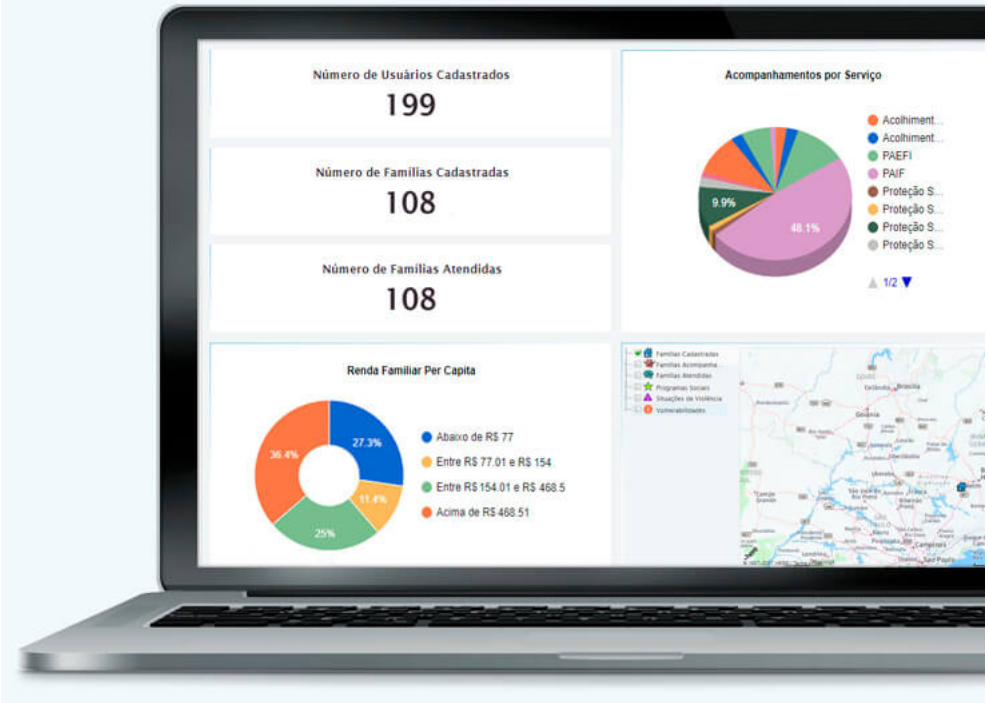
Fonte: (IDS Sistemas - Brasil Gestão Pública, 2018).

2.3 GESUAS

O Gesuas ou Gestão do Sistema Único de Assistência Social, é um sistema para gestão e acompanhamento das ações e políticas de assistência social do SUAS. O Gesuas é a primeira versão online do prontuário SUAS. Com ele não há limite de prontuários e a impressão dos mesmos ficará a critério e necessidade dos técnicos. É o mesmo prontuário físico disponibilizado pelo Ministério do Desenvolvimento Social, só que online. O Gesuas auxilia os gestores na efetiva implementação da vigilância, gerando informações de forma territorializada (georreferenciada) e exibindo indicadores para gestão. O painel do gestor, mostrado na Figura 3, permite monitoramento em tempo real das atividades e ações realizadas nos equipamentos. Os relatórios para monitoramento, como o Registro Mensal de Atendimento (RMA) são gerados em

segundos. O Gesuas auxilia os gestores na efetiva implementação da vigilância, gerando informações de forma territorializada (georreferenciada) e exibindo indicadores para gestão (GESUAS, 2023).

Figura 3 – Tela de monitoramento em tempo real GESUAS



Fonte: (GESUAS, 2023).

2.4 Comparativo

Apresentado os documentos e as tecnologias similares ao Open Social Care, é destacado que essas ferramentas são de utilização paga, sendo inviável o uso por muitas instituições de serviços sociais ou até mesmo por assistentes sociais, pois a maioria dos trabalhos sociais realizados é voluntário, como no caso do CCG, logo o Open Social Care seria livre para uso, contribuindo para a área de atendimento social. Outro ponto é que essas ferramentas citadas podem ter um uso muito específico no cadastro de atendimentos, como uma das propostas do sistema é a generalização, a aplicação irá poder atender uma área mais abrangente. Além disso, vale destacar que o Prontuário SUAS, embora forneça recursos para o cadastro de pessoas, não atende as especificidades dos atendimentos das assistentes sociais com relação aos PPLs.

3 O OPEN SOCIAL CARE

O Open Social Care é um sistema para gerenciamento de atendimentos sociais, ele foi criado para auxiliar o dia a dia de trabalho de assistentes sociais do CCG. Devido aos detalhes supracitados que dificultam a realização do trabalho destes funcionários sendo importante a utilização de um sistema para suprir as necessidades do cadastro de atendimentos.

O sistema surgiu como um projeto de disciplina de Sistemas Distribuídos da UTFPR, câmpus Guarapuava, quando uma assistente social do CCG entrou em contato com o professor da disciplina e sugeriu a ideia de informatização do processo até então utilizado no CCG. Através de reuniões e discussões, foi feita a coleta dos requisitos e planejado como o sistema teria que funcionar, por fim, foi desenvolvido um protótipo como um objeto de estudos na disciplina.

Dada a característica e os tópicos da disciplina, optou-se pelo estudo e experimentação da abordagem *Serverless* de modo que cada aluno pudesse configurar a infraestrutura e trabalhar na aplicação sem precisar de um servidor *Backend*. Na ocasião, as principais tecnologias utilizadas foram *Firebase* e *React.JS*.

Com o *Firebase*, foi possível fazer uso de serviços prontos, como o banco de dados, autenticação de usuários e a hospedagem de arquivos e páginas. Conforme ilustrado na Figura 4, utilizou-se o *Firestore Database*, que é um banco de dados *NoSQL*, nele é possível armazenar dados em documentos que contêm mapeamentos de campos para valores. Esses documentos são armazenados em coleções, que são contêineres de documentos que podem ser usados para organizar dados e criar consultas. Os documentos suportam muitos tipos de dados diferentes, desde *strings*[1] e números simples a objetos complexos e aninhados. Também é possível criar subcoleções dentro dos documentos e criar estruturas de dados hierárquicas que podem ser escalonadas à medida que o banco de dados cresce (FIREBASE, 2023a).

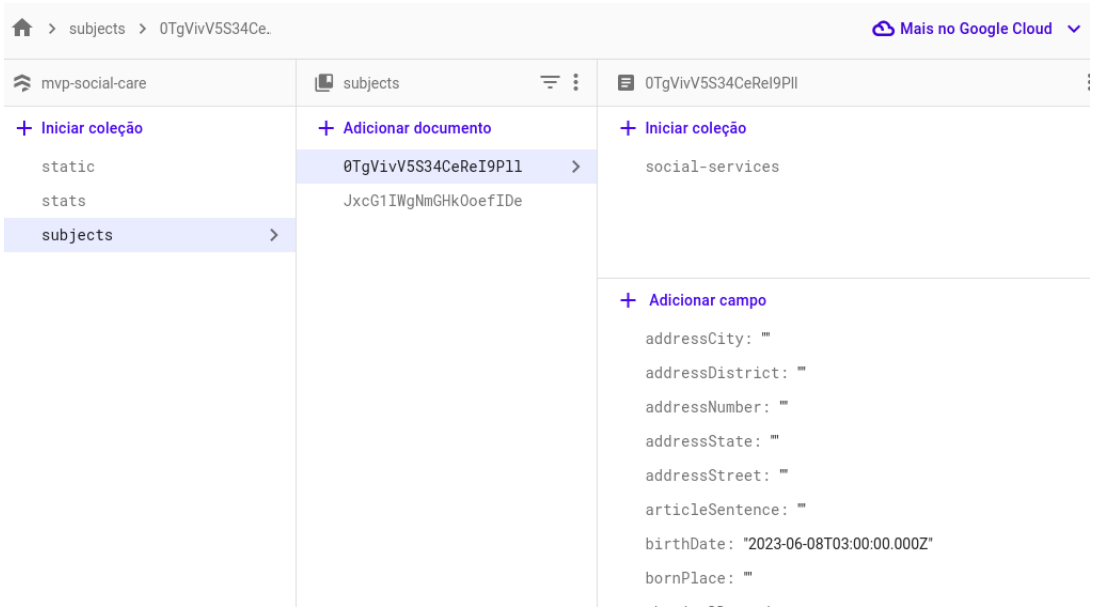
Para autenticação foi usado o *Authentication*, ele fornece serviços de *Backend*, *Software Development Kit* (SDK) e bibliotecas de Interface do usuário (IU) prontas para autenticar usuários no seu aplicativo. Ele oferece suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como *Google*, *Facebook* e *Twitter*, entre outros (FIREBASE, 2023c). Além disso, foi usado para salvar imagens e arquivos o *Firebase Storage*, um serviço de armazenamento de objetos avançado, simples e econômico criado para a escala do *Google*. Com os SDKs do *Firebase* para *Cloud Storage*, é possível usar a segurança do *Google* para fazer *upload* e *download* de arquivos nos aplicativos do *Firebase*, independentemente da qualidade da rede (FIREBASE, 2023b).

De modo geral, o experimento com a arquitetura *Serverless* foi bastante válido no contexto da disciplina, entretanto, conforme já mencionado, o modelo sofre limitações em relação ao seu uso prático em cenário real. Ademais, a generalização do sistema, que irá possibilitar a utilização dele para demais instituições além do CCG e a possibilidade de hospedá-lo em um

[1] Sequências de caracteres alfanuméricos (letras, números e/ou símbolos)

servidor próprio, sem a dependência de tecnologias de terceiros, aumenta a contribuição e as possibilidades de aplicação do projeto.

Figura 4 – Tela de gerenciamento de banco de dados do Open Social Care no Firebase.



Fonte: Autoria própria (2023).

4 MATERIAIS E MÉTODOS

A ênfase deste capítulo está em descrever as principais ferramentas utilizadas no projeto e o processo de organização utilizado. Este capítulo está subdividido em duas seções, 4.1 Materiais e 4.2 Métodos.

4.1 Materiais

Nesta seção são apresentadas as principais ferramentas e tecnologias utilizadas para o desenvolvimento da API do sistema.

4.1.1 Git e Github

O *Git* se trata de um sistema de controle de versionamento de código, possibilitando que possua diferentes versões e funcionalidades sendo desenvolvidas simultaneamente e independentes entre si (GIT, 2023), ao permitir criar várias instâncias do código, um sistema de controle de versionamento ajuda no trabalho em equipe, pois permite que cada pessoa possa trabalhar a sua parte individualmente, também permite rastrear e gerenciar alterações no código, como ao encontrar algum *bug* (erro), com o *Git* é possível achar o *commit* (envio no Git) que levou ao *bug*, analisar as mudanças e fazer a correção, sendo muito mais prático e rápido.

Já o *Github* é uma aplicação web que possibilita a hospedagem de repositórios *Git* (AQUILES, 2014). Para o desenvolvimento é bastante importante ter o código hospedado em um repositório, pois assim facilita o compartilhamento, o controle de versionamento, como mantém o projeto seguro e salvo. O *Git* e o *Github* trabalham em conjunto para auxiliar no desenvolvimento de um projeto, são duas ferramentas bastante utilizada por desenvolvedores.

4.1.2 PHP e Laravel

O *Hypertext Preprocessor* (PHP) é uma linguagem de código aberto de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do *HyperText Markup Language* (HTML) (PHP, 2023). Devido ao fácil aprendizado e desenvolvimento, a linguagem PHP facilita o caminho do profissional que escolhe estruturar sites utilizando a plataforma. Dessa maneira, as edições e configurações são feitas de uma maneira muito mais simplificada [...] de forma prática, é uma linguagem sem custos e com melhorias constantes, pois qualquer programador pode melhorar seu código, retirando possíveis erros ou adicionando mais funções (CONTEIGE, 2023).

O *Laravel* é um *framework* feito para o PHP, [...] um *framework* compreende um conjunto de classes ou funções implementadas em uma linguagem de programação especifica usadas

para auxiliar o desenvolvimento de um *software* (GABARDO, 2017). A utilização da linguagem de programação PHP para desenvolvimento de aplicações *Web* é facilitada com o uso do *Laravel* uma vez que diversos recursos são fornecidos já implementados, além disso, a adoção do *framework* contribui para a organização do projeto visto que padrões e nomenclaturas precisam ser adotados e padronizados. Outro beneficio do uso do *framework* é a garantia da segurança da aplicação, visto que procedimentos de autenticação já são fornecidos e testados pela comunidade de desenvolvedores.

4.1.3 Docker

O Docker é um software de código aberto usado para implantar aplicativos dentro de *containers* (ambiente isolado) virtuais. A conteinerização permite que vários aplicativos funcionem em diferentes ambientes complexos (C.), 2023). Ao permitir criar e gerenciar contêineres é possível configurar todas as dependências da aplicação, o que ajuda a manter um padrão e facilitar a implementação em diferentes ambientes, contribuindo para o trabalho em equipe e implementação do sistema em diversos servidores.

4.1.4 PostgreSQL

PostgreSQL é um sistema de banco de dados com código aberto, altamente estável que fornece suporte a diferentes funções de *Structured Query Language* (SQL), como chaves estrangeiras, sub consultas, *triggers* (eventos), e diferentes tipos e funções definidas pelo usuário. Ele aumenta ainda mais a linguagem SQL oferecendo várias características que meticulosamente escalam e reservam cargas de trabalho de dados. É usada principalmente para armazenar dados para muitos aplicativos móveis, web, geoespaciais e analíticas (KINSTA, 2023).

O PostgreSQL utiliza de transações Atomicidade, Consistência, Isolamento, Durabilidade (ACID), as transações ACID maximizam a confiabilidade e a integridade dos dados. Os dados permanecem consistentes mesmo se a operação for concluída apenas parcialmente. Por exemplo, se cair a energia inesperadamente durante a gravação em uma tabela em um banco de dados, sem transações ACID, pode ser que apenas parte dos dados seja salva, e o restante, não. O resultado é um banco de dados inconsistente, tornando a recuperação difícil e demorada (DATABRICKS, 2023). Com isso a utilização do PostgreSQL contribui para a segurança dos dados, criando uma aplicação confiável e estável.

4.2 Métodos

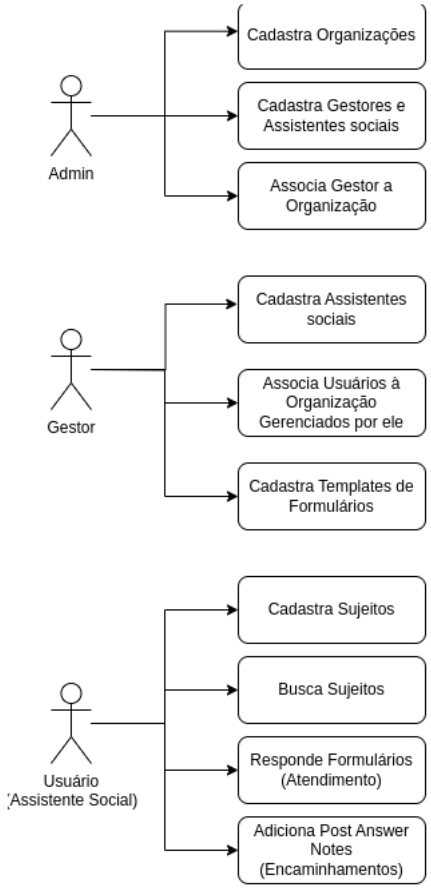
Nesta seção são apresentados os métodos de planejamento para o desenvolvimento do projeto.

4.2.1 Análise e coleta de requisitos

Para a coleta de requisitos do sistema, realizou-se uma apresentação do protótipo do sistema, e por meio de reuniões foi discutido suas funcionalidades e melhorias. Conforme o levantamento dos requisitos, separamos com base em requisitos funcionais, técnico e operacional e utilizamos a metodologia *MoSCoW*(*Must-Have, Should-Have, Could-Have e Won't-Have*). O método *MoSCoW* é uma técnica de priorização usada na gestão de projetos e desenvolvimento de softwares com o intuito de encontrar um entendimento em comum entre as partes interessadas sobre a importância que elas atribuem a cada requisito (PIRES, 2019), ele se baseia em etapas, o que o sistema precisa ter, deveria ter, poderia ter e não teria.

Além disso, fluxos de casos de uso foram pensados para melhor entender as funcionalidades que o sistema deveria ter a fim de validar a utilização e o comportamento da aplicação com base nos diferentes perfis de usuário, um exemplo de diagrama de casos de uso pode ser visto na Figura 5. O caso de uso descreve o comportamento do sistema sob diversas condições conforme o sistema responde a uma requisição de um chamado ator primário. O ator primário inicia uma interação com o sistema para atingir um objetivo. O sistema responde, protegendo o interesse de todos (COCKBURN, 2005).

Figura 5 – Diagrama de casos de uso para os perfis do sistema

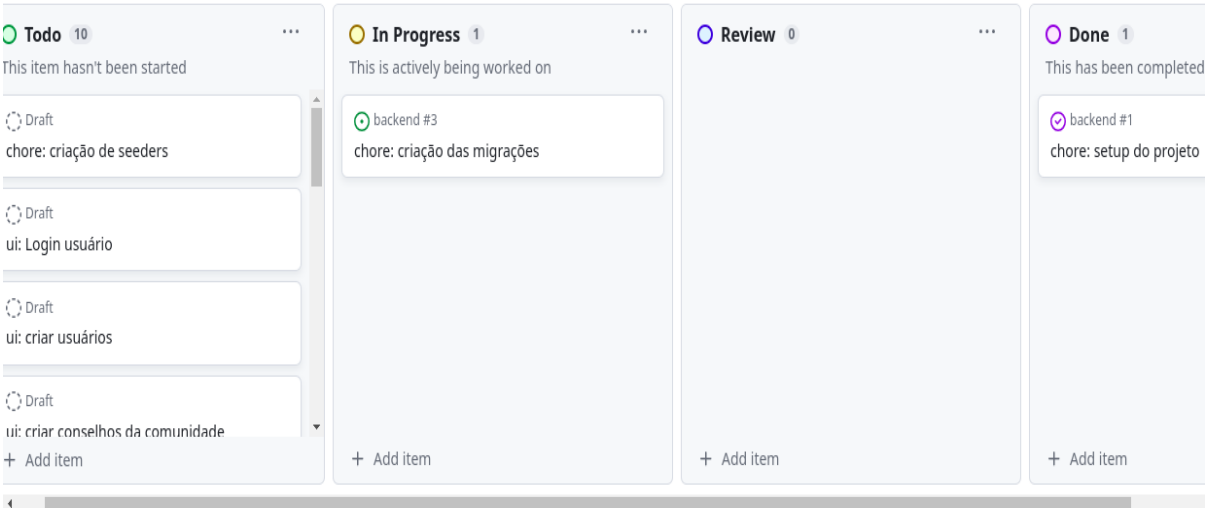


Fonte: Autoria própria (2023).

4.2.2 Organização de tarefas para o desenvolvimento

Para o desenvolvimento das atividades, é feita a utilização da metodologia Kanban. A metodologia Kanban pode ser resumida em uma maneira de organizar os processos que envolvem as equipes de uma organização, ressaltando a priorização das tarefas e tornando o foco bem definido para todos. Dessa maneira, é possível identificar e resolver problemas no fluxo de trabalho (AMORIM, 2023). Na separação das atividades, foi utilizado o recurso *projects* do *Github*, onde é possível separar as atividades em *cards* e organizar com base na sua etapa, *To do* (para fazer), *In Progress* (Em progresso), *Review* (Revisão) e *Done* (Feito), conforme expresso na Figura 6.

Figura 6 – Quadro kanban das atividades no github



Fonte: Autoria própria (2023).

4.2.3 Planejamento do banco de dados

Devido às características do problema, optou-se pelo uso de um modelo relacional de banco de dados. Um banco de dados armazena e fornece acesso a pontos de dados relacionados entre si. O modelo relacional é uma maneira intuitiva e direta de representar dados em tabelas. Em um banco de dados relacional, cada linha na tabela é um registro com uma ID exclusiva chamada chave. As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados (ORACLE, 2023).

Também realizou-se uma modelagem do banco de dados, onde foi utilizada a abordagem de modelo Entidade relacionamento (ER), um diagrama ER é um tipo de fluxograma que ilustra “entidades” como pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema. Diagramas ER são mais utilizados para projetar ou depurar bancos de dados relacionais nas áreas de engenharia de software, sistemas de informações empresariais, educação

e pesquisa. Também conhecidos como DERs, ou modelos ER, usam um conjunto definido de símbolos, tais como retângulos, diamantes, ovais e linhas de conexão para representar a interconectividade de entidades, relacionamentos e seus atributos. Eles espelham estruturas gramaticais, onde entidades são substantivos e relacionamentos são verbos (LUCID, 2023).

5 RESULTADOS

5.1 Requisitos

O Open Social Care é um projeto que foi apresentado para as assistentes sociais do CCG, o protótipo inicial teve uma breve fase de testes, e no decorrer da utilização do sistema, foram reveladas dificuldades ao realizar alguns passos, além das preocupações técnicas, como permissão dos usuários, acessos ao sistema e a infraestrutura. Portanto, através disso foram feitas reuniões para validar os requisitos e coletar funcionalidades que o sistema poderia suprir. Além disso, foram feitas reuniões mais técnicas para validar a estrutura do sistema para chegar em um melhor resultado.

Durante essa fase de validação do protótipo, como dito anteriormente, foi vista a necessidade de mudar sua estrutura inicial, migrando de uma arquitetura sem servidor para uma API, logo com essa remodelagem foi preciso fazer uma nova análise com base nos requisitos já coletados para o protótipo e também melhorias para o sistema. Na Tabela 1, é possível ver os requisitos coletados nessa etapa inicial. Esses requisitos foram classificados pelo método MoSCoW, com as prioridades definidas a partir do protótipo anterior.

Requisito	Tipo (MoSCoW)	Descrição
01	M	Possuir tipos de perfis de acesso e permissões diferentes para cada (admin, gestor e assistente social)
02	M	Possuir cadastro, edição, listagem e visualização de sujeitos
03	M	Possuir cadastro, edição, listagem e visualização de atendimentos com base em um sujeito
04	S	Possuir cadastro, edição, listagem e visualização de organizações
05	M	Possuir cadastro, edição, visualização de questionários personalizados para atendimentos
06	M	Possuir cadastro, edição, listagem e visualização de usuários
07	M	Possuir login por usuário
08	C	Possuir servidor *staging*
09	C	Possuir upload de arquivos e imagens no cadastro de atendimentos
10	W	Possuir recursos para geração de relatórios

Tabela 1 – Listagem de requisitos básicos do sistema

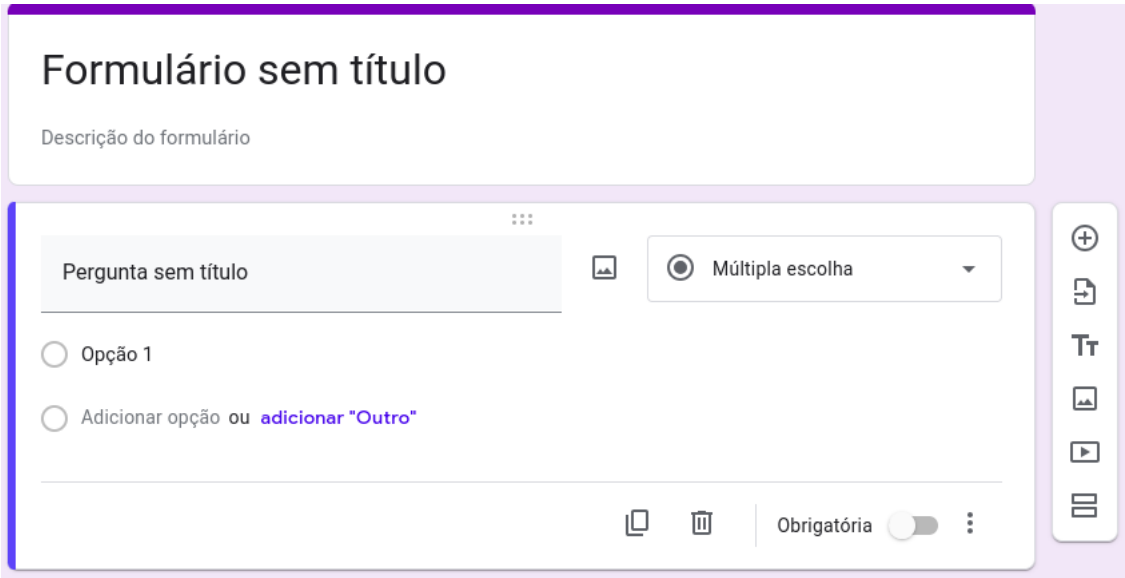
5.2 Generalização do sistema

O protótipo do Open Social Care foi feito visando apenas as necessidades do CCG, tornando o sistema fortemente engessado as demandas e informações que são convenientes para eles. Durante a apresentação do sistema e reuniões para discutir o uso e validação, foi visto um potencial para que a aplicação pudesse ser utilizada por mais instituições além do CCG, com isso surgiu a ideia de ter um sistemas mais flexível e generalizado para uso de instituições de forma geral.

Para a construção de uma aplicação que possa ser utilizada por mais instituições, pensou-se no desenvolvimento de cadastro de múltiplas organizações, pois o protótipo do Open Social Care só pode ser utilizado por uma única organização, ao ter um cadastro para isso, o sistema poderia ser utilizado por várias instituições de uma única vez, onde cada instituição

teria os seus sujeitos e atendimentos separados, logo esse seria o primeiro passo para a generalização do sistema. O segundo passo é a ideia de criação de questionários dinâmicos, onde seria possível criar vários modelos de formulários para cadastros de atendimentos, não deixando mais o sistema com campos fixos que serviriam apenas uma instituição. Um exemplo de questionários dinâmicos seria o cadastro de formulário da empresa *Google*, onde as perguntas são personalizadas, desde opções e tipos de perguntas podendo ser descritiva ou de escolha, como mostrado na Figura 7.

Figura 7 – Tela para cadastro de um formulário *Google*



Fonte: Autoria própria (2023).

A generalização do sistema é uma solução tecnológica para atender variedade de necessidades institucionais. A flexibilidade dele pode trazer economia de tempo, pois as instituições podem fazer uso do sistema para diferentes tipos de atendimentos, pode ajudar também na padronização, visto que ao criar um questionário ele pode ser usado por mais instituições, o que permite a troca de informações e boas práticas e ao construir um sistema que possa servir de forma genérica as organizações, é construído um sistema com possibilidade de crescimento e adaptação a mudanças de normas e regras, sendo uma aplicação que irá ser independente para atender vários objetivos. A generalização irá desempenhar um papel fundamental para o crescimento e abrangênciado do Open Social Care.

5.3 Segurança

Segurança é uma das características principais em qualquer sistema de informações, ela garante a proteção dos dados, a prevenção de ameaças e o gerenciamento de riscos. Garantir a segurança é fundamental para preservar a integridade, confidencialidade e disponibilidade das informações. No caso do Open Social Care que pode possuir dados sigilosos, uma das maiores preocupações é sobre essa segurança.

O primeiro passo para garantir a segurança seria a implementação de *tokens* (senhas) para autenticação, sendo mais comum o nome de *login* (acesso ao sistema), isso irá validar a identidade de usuários. O login funciona como uma chave. Com ele, somente pessoas com as credenciais em mãos podem acessar computadores, desbloquear celulares, entrar em uma conta de rede social, entre outros casos. Dessa forma, o método confere mais segurança aos usuários (BLASI, 2022). Para a aplicação será usado a emissão de *tokens* do *Sanctum*, "*que é um pacote de autenticação API para o Laravel [...] Sanctum permite que você emita tokens de API/tokens de acesso pessoal que podem ser usados para autenticar solicitações de API para seu aplicativo*" (LARAVEL, 2023b). Para garantir a segurança dos dados os *tokens* são criptografados usando *hash SHA-256* (função de resumo criptográfico que converte dados em uma sequência alfanumérica fixa de 256 *bits*), além de possuir *Middlewares*, eles funcionam como filtros para validação de acesso dos usuários, são usados para verificar se uma solicitação recebida é autenticada com um *token* ao qual foi concedida uma determinada habilidade (LARAVEL, 2023b), se o usuário tiver um *token* inválido ou não possuir permissões para realizar ações dentro do sistema, ele é bloqueado.

Para evitar o vazamento de informações, foi pensado também na criação de *logs* (histórico) de acesso, quando um usuário fizer determinada ação, como por exemplo, o acesso a uma área do sistema ou cadastros, essa informação será salva. Desta forma seria possível ter um controle do que está acontecendo no sistema e caso tenha algum vazamento de informação ser possível identificar o usuário. Essa funcionalidade poderá ser implementada criando uma tabela no banco de dados, quando um usuário realizar uma ação no sistema, como por exemplo, visualização de um sujeito, essa informação seria salva na tabela de *logs*, contendo as informações do usuário que acessou, qual foi o sujeito e quando isso ocorreu, sendo possível então ter um controle de gestão do sistema. *Logs* são registros em forma de arquivos de texto puro em linhas, onde essas linhas contém informações relativas a data e hora em que ações importantes ocorreram na aplicação. "*Ou seja, o que e quando ocorreu determinado evento[...] Com o uso de logs fica mais fácil obter a visibilidade da integridade da infraestrutura de TI (Tecnologia da Informação). As informações contidas nestes registros podem variar desde o desempenho do sistema, dados sobre auditoria, alertas de intrusão ou até transações e atividades de usuários*" (ROCHA, 2021).

Outro ponto a ser falado sobre segurança, é a utilização do *framework Laravel*, o *Laravel* possui muitos recursos que tornam a aplicação segura e facilitam o desenvolvimento. Para autenticação ele utiliza *providers* (provedores) e *guards* (guardas) para facilitar o processo de autenticação. O objetivo dos *guards* é autenticar os usuários para cada solicitação que eles fazem, enquanto os *providers* facilitam a recuperação dos usuários do banco de dados (POPOVSKI, 2019). Um ataque comum também pode acontecer ao banco de dados e para isso o *Laravel* também tem proteções contra injeções no banco de dados, a ferramenta *Eloquent ORM (Object-Relational Mapping ou Mapeamento objeto-relacional)* usa ligação de parâmetros *PDO* (PHP Data Objects ou Objetos de dados PHP) para evitar injeção de *SQL*. A vinculação de pa-

râmetros garante que usuários mal-intencionados não possam transmitir dados de consulta que possam modificar a intenção da consulta (POPOVSKI, 2019). O *Laravel* também possui recursos de validação e permissões para acesso as rotas e criação de dados, entre outros recursos que garantem uma aplicação segura.

Nessa reformulação do sistema também foi pensado na criação de permissões e perfis de acesso, onde cada perfil pode acessar diferentes partes do sistema enquanto outras ficam oculta para ele. Os perfis de acesso pensado foram: administrador(a), gestor(a) e assistente social. O perfil de acesso principal seria o administrador(a), onde teria acesso a principais partes do sistema, como cadastro de usuários, organizações e manteria um controle sobre a aplicação. O perfil gestor(a) teria acesso a cadastro de assistentes sociais, sujeitos e atendimentos. O perfil assistente social seria o com menos poder no sistema, tendo acesso apenas a cadastro de sujeitos e atendimentos. Essa diferença de acesso garante uma organização no sistema e segurança também, devido a ter regras específicas de acesso.

5.4 Escopo do sistema

O sistema Open Social Care tem como principal objetivo fazer a gestão de atendimentos sociais, na aplicação deve ser possível cadastrar sujeitos, com dados básicos, como por exemplo, nome e data de nascimento e para esses sujeitos cadastrar atendimentos, onde terá dados bem abrangentes, como por exemplo, dados sobre familiares e pedidos de itens ou documentação.

A aplicação irá oferecer níveis de acesso, administrador, gestor(a) e assistente social, onde cada perfil tem acessos e permissões diferentes no sistema. Com isso o sistema oferece o cadastro de organizações, e dentro dessas organizações é possível adicionar usuários que irão cadastrar os sujeitos e os atendimentos específicos para a organização que o usuário está alocado. Exemplo de uso: o administrador cria uma organização X, que fica responsável pela gestora Y, e então a gestora Y cria uma assistente social para sua organização, e essa assistente social fica encarregada de criar sujeitos e atendimentos.

Outro detalhe importante, é que o sistema irá disponibilizar da criação de questionários personalizados para os formulários de atendimentos, para que diversas instituições possam fazer uso do sistema. Questionários personalizados podem ser criados pelo administrador e então alocados a uma organização, onde a assistente social vai fazer uso ao cadastrar atendimentos.

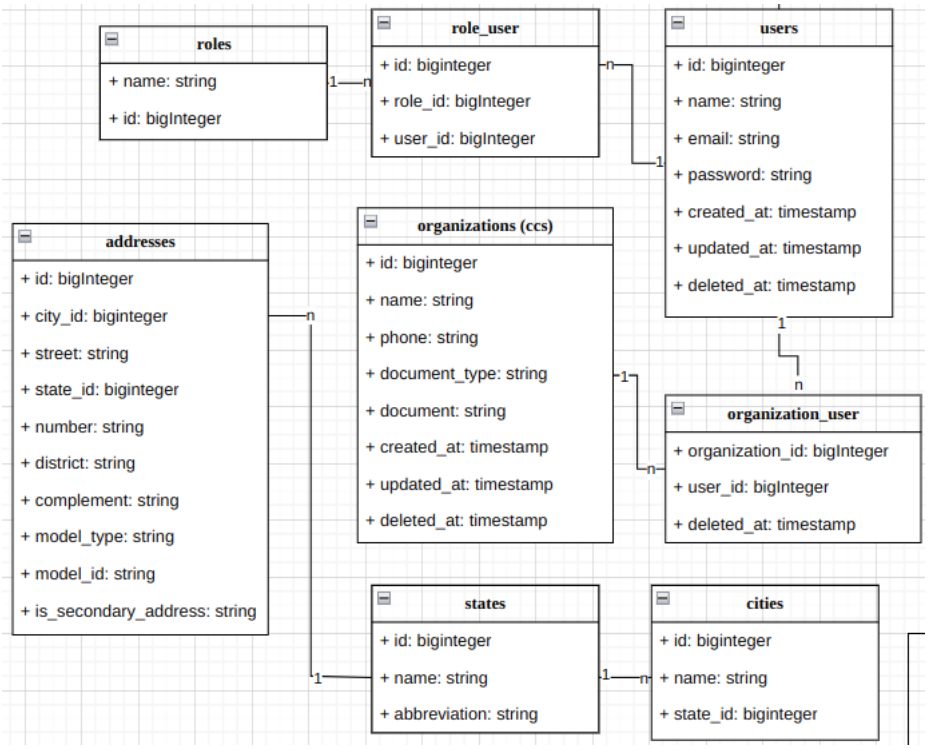
5.5 Modelagem do sistema

O banco de dados do protótipo do Open Social Care foi feito usando o *Firestore Database* do *Firebase*, e como nessa refatoração da infraestrutura será utilizado o banco de dados *PostgreSQL* junto com o *framework Laravel*, não será possível reaproveitar o modelo de dados

anterior. Por se tratarem de estruturas bem diferentes uma da outra, o *PostgreSQL* é um banco de dados objeto relacional, ele utiliza do SQL, enquanto o *Firestore Database* utiliza de dados não relacionais o *Not Only SQL* (NoSQL). O modelo de tabela relacional do SQL é altamente estruturado e requer que os dados sejam organizados em tabelas com esquemas pré-definidos. O modelo de documento do NoSQL, por outro lado, é mais flexível e permite que os dados sejam armazenados em documentos que não precisam seguir um esquema rígido (IMPACTA, 2023). Logo nessa refatoração do sistema foi preciso fazer a criação de um novo modelo de banco de dados, através de análises e reuniões levando em conta os requisitos coletados e funcionalidades, foram identificadas as entidades necessárias, como por exemplo organizações e sujeitos, definido chaves primárias e chaves estrangeiras que estabelecem os relacionamentos entre as tabelas, e com base nisso, um diagrama visual do banco de dados foi criado e está disponível no Apêndice A [1].

Tratando-se dos elementos básicos do sistema, o banco de dados conta com tabelas para o cadastro de organizações, as quais podem possuir usuários. Usuários por sua vez podem possuir *roles* (perfis) para diferenciar seus diferentes papéis e permissões. A Figura 8 demonstra a modelagem destes elementos.

Figura 8 – Fragmento do diagrama de banco de dados destacando as organizações e usuários

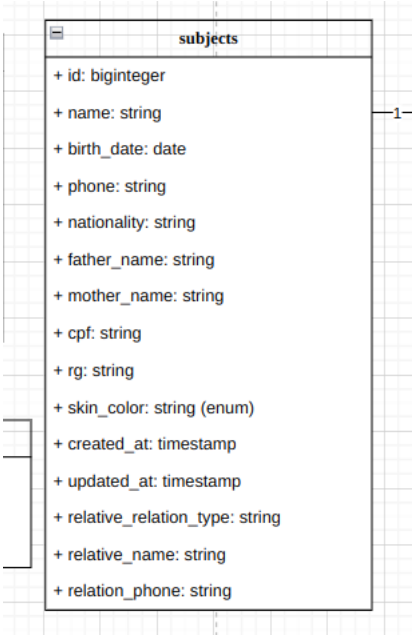


Fonte: Autoria própria (2023).

[1] Uma versão digital do diagrama também pode ser visualizada em: Link para o diagrama do banco de dados

Os sujeitos atendidos pelas assistentes sociais, neste trabalho denominados PPLs, são representados pela tabela *subjects* mostrado na Figura 9. Esta entidade é modelada separadamente dos usuários, visto que são indivíduos que não possuem login ou acesso direto ao sistema.

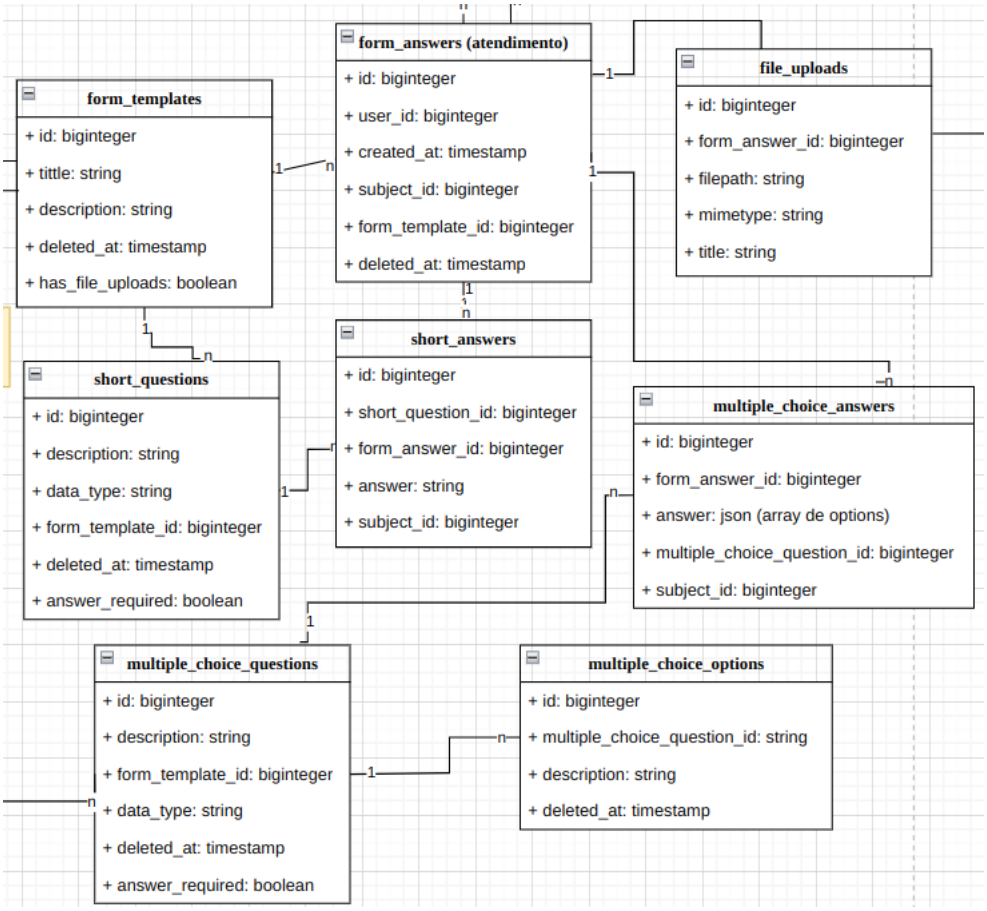
Figura 9 – Fragmento do diagrama de banco de dados destacando os sujeitos



Fonte: Autoria própria (2023).

A atividade realizada por assistentes sociais (usuários do sistema), é registrada no banco de dados por meio das tabelas $form_answers$ que simboliza os atendimentos, $short_answers$ e $multiple_choice_answers$ as quais representam os preenchimentos das perguntas cadastradas nos questionários personalizados, onde um formulário é baseado em um *template* e possui diferentes tipos de perguntas, as quais possuem tabelas específicas para o registro das respostas. Esse fluxo de banco de dados pode ser observado na Figura 10.

Figura 10 – Fragmento do diagrama de banco de dados destacando os atendimentos e questionários personalizados



Fonte: Autoria própria (2023).

5.6 Implementação

A inicialização da implementação do projeto começou com a configuração do repositório na plataforma *Github*, onde foi criada uma organização Open Social Care para que os projetos específicos do tema ficassem alocados, facilitando o armazenamento e o compartilhamento do projeto. Logo dentro dessa organização se encontra o repositório *backend* do sistema, onde será feita a implementação da API. [2] Para o inicio do desenvolvimento da API, foi configurado e inciado o projeto com o *framework Laravel* utilizando o *Sail* para a configuração do ambiente de desenvolvimento, para isso, foram utilizados os passos a passos da documentação oficial do *Laravel*, que pode ser encontrada nesse link https://laravel.com/docs/10.x/installation. O *Laravel Sail* é uma interface de linha de comando leve para interagir com o ambiente de desenvolvimento *Docker* padrão do *Laravel* (LARAVEL, 2023a), o *Sail* é um facilitador para utilizar o *Docker* no ambiente de desenvolvimento, o projeto foi criado usando a configuração inicial do *Laravel* junto com o *PostgreSql*.

[2] Link do Github para a organização: https://github.com/open-social-care

A configuração inicial usando o *Laravel Sail* vem com arquivos e funcionalidades padrões para começar a aplicação, porém como o projeto se trata de uma API algumas funcionalidades e arquivos não são necessárias, por isso foi utilizado o *Starter Kit do breeze*, um inicializador para o projeto que vem com configurações especificas para o desenvolvimento, no caso, foi usado visando apenas o *backend*, logo remove os recursos destinados às *views* do projeto e adiciona funcionalidades básicas para gestão de usuários, esses *kits* estruturam automaticamente seu aplicativo com as rotas, controladores e visualizações necessárias para registrar e autenticar os usuários do seu aplicativo (LARAVEL, 2023c).

Iniciando a implementação do sistema, seguindo o modelo de dados planejado, elaborou-se as *migrations* (migrações) [3] de todas as tabelas do banco de dados, também referenciando os relacionamentos entre elas. Na Listagem 1 é possível ver um exemplo de *migration*, ela seria para a criação da tabela *subjects* (sujeitos) com seus campos seguindo o diagrama do banco de dados. Seguindo a criação das *migrations*, foram construídos os *models* (modelos) [4] para cada tabela do banco de dados, nos *models* é feito a associação dos relacionamentos ao qual a tabela tem ou pertence, na Listagem 2 está mostrando uma classe *model* do *subject* do sistema, nela tem os seus campos e relacionamentos.

Finalizando essa etapa de construção da aplicação, realizou-se os testes, eles são importantes para garantir a qualidade e confiabilidade do sistema, os testes permitem identificar falhas, economizando tempo e recursos. Para o inicio do projeto, foram feitos os testes unitários, eles servem para testar partes individuais de um código, como por exemplo uma função ou método, no caso do Open Social Care, é utilizada para testar funcionalidades dos *models*, como por exemplo seus relacionamentos, esses testes iniciais, garantem que os *models* foram realizados da forma correta e não haverá erros, como também que a aplicação está seguindo à modelagem prevista, um exemplo de teste pode ser visto na Listagem 3, onde foi testado o relacionamento do *model* de *subject*. Para os teste unitários, precisou-se da criação das *factories*, elas são usadas para criação de dados falsos, muito utilizados nos testes, ajudam a simular um *model* em um cenário real da aplicação, sem precisar que seja acessado ou criado via banco de dados, facilitando o desenvolvimento, na Listagem 4 temos uma *factory* criada para o *model* do *subject*, é atribuído dados falsos que são gerados com a função *fake* do *Laravel* para os campos necessários do *subject* a ser usado em testes.

---

[3] Migrations: São uma forma de gerenciar as mudanças na estrutura do banco de dados de uma aplicação. Eles permitem que você crie, altere ou exclua tabelas, colunas, índices e outros elementos do banco de dados de forma consistente e controlada (KRIGER, 2023).

[4] *Model*: é uma classe para representar uma entidade do banco de dados, com ela é possível inserir, editar e recuperar dados

Listagem 1 – *Migration* criada para os *subjects*

```php
<?php

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('subjects', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->date('birth_date');
            $table->string('nationality')->nullable();
            $table->string('phone')->nullable();
            $table->string('father_name')->nullable();
            $table->string('mother_name')->nullable();
            $table->string('cpf')->nullable();
            $table->string('rg')->nullable();
            $table->enum('skin_color',
                array_column(SkinColorsEnum::cases(), 'value')
            )->nullable();
            $table->string('relative_relation_type')->nullable();
            $table->string('relative_name')->nullable();
            $table->string('relative_phone')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('subjects');
    }
};
```

Fonte: Autoria própria (2023).

Listagem 2 – *Model* criada para o *subject*

```php
<?php
class Subject extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'birth_date',
        'nationality',
        'phone',
        'father_name',
        'mother_name',
        'cpf',
        'rg',
        'skin_color',
        'relative_relation_type',
        'relative_name',
        'relative_phone',
        'created_at',
        'updated_at',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'birth_date' => 'datetime',
    ];

    public function formAnswers(): HasMany
    {
        return $this->hasMany(FormAnswer::class);
    }
}
```

Fonte: Autoria própria (2023).

Listagem 3 – Teste unitário criado para o *model* de *subject*

```php
<?php

class SubjectTest extends TestCase
{
    use RefreshDatabase;

    public function testShortQuestionHasManyFormAnswer()
    {
        $subject = Subject::factory()->createOneQuietly();
        $formAnswer1 = FormAnswer::factory()->for($subject)
            ->createOneQuietly();
        $formAnswer2 = FormAnswer::factory()->for($subject)
            ->createOneQuietly();

        $this->assertTrue($subject->formAnswers->contains($formAnswer1));
        $this->assertTrue($subject->formAnswers->contains($formAnswer2));
    }
}
```

Fonte: Autoria própria (2023).

Listagem 4 – *Factory* criada para o *model* de *subject*

```php
<?php

class SubjectFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        $skinColors = array_column(SkinColorsEnum::cases(), 'value');

        return [
            'name' => fake()->name,
            'relative_name' => fake()->name,
            'relative_relation' => fake()->name,
            'birth_date' => fake()->dateTimeThisDecade,
            'contact_phone' => fake()->phoneNumber,
            'cpf' => fake()->unique()->numerify('###.###.###-##'),
            'rg' => fake()->unique()->numerify('##.###.###-#'),
            'skin_color' => $this->faker->randomElement($skinColors),
        ];
    }
}
```

Fonte: Autoria própria (2023).

6 CONSIDERAÇÕES FINAIS

Neste trabalho é abordado o desenvolvimento de uma API para substituir a infraestrutura *backend* do protótipo Open Social Care. Diante desse desafio, foram descritas as principais ferramentas e métodos a serem utilizadas para alcançar os objetivos propostos.

O protótipo inicial é fortemente dependente de serviços prontos, diante disso, acredita-se que propor uma nova arquitetura com infraestrutura passível de gerenciamento favorecerá a implantação e a manutenção do sistema em cenários reais. Além disso, destaca-se que a infraestrutura inicial não é fácil de ser configurada, precisando ser feita individualmente para cada instituição que irá utilizar o sistema podendo tornar o protótipo inviável para algumas finalidades.

Também diante dos documentos e ferramentas apresentados, similar ao Open Social Care e que são utilizado por assistentes sociais, é destacado que todas as ferramentas são de utilização paga, sendo inviável para ser utilizado por algumas instituições, como é o caso do CCG, logo a construção do sistema irá favorecer na área por se tratar de um sistema gratuito e livre para uso. Este sistema visa atender à diferentes contextos com o uso de formulários personalizáveis, tornando-o genérico para atendimentos de assistentes sociais. Além de trazer o benéfico da generalização, o sistema poderá ser utilizado por diversas instituições sendo abrangente para atender vários objetivos e demandas, fazendo-o ter um maior alcance para sua utilização e, consequentemente, ampliando a contribuição do presente trabalho.

REFERÊNCIAS

AMORIM, S. **O que é Kanban e como usar essa metodologia**. 2023. Disponível em: https://enotas.com.br/blog/kanban/#:~:text=O%20que%20%C3%A9%20a%20metodologia%20Kanban%3F,problemas%20no%20fluxo%20de%20trabalho. Acesso em: 12 set. 2023.

AQUILES, A. **Controlando Versões com Git e GitHub**. São Paulo, SP - Brasil: Casa do Código, 2014. 199 p.

BLASI, B. G. D. **O que é login? Conheça os principais métodos de acesso no meio digital**. 2022. Disponível em: https://tecnoblog.net/responde/o-que-e-login/#:~:text=Por%20que%20o%20login%20%C3%A9,confere%20mais%20seguran%C3%A7a%20aos%20usu%C3%A1rios. Acesso em: 07 nov. 2023.

C.), D. **O Que é Docker e Como Ele Funciona?** 2023. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-docker. Acesso em: 14 nov. 2023.

COCKBURN, A. **Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software**. Porto Alegre, RS - Brasil: Bookman Editora, 2005. 256 p.

CONTEIGE. **PHP - Vantagens e Desvantagens**. 2023. Disponível em: https://conteige.cloud/php-vantagens-e-desvantagens/#:~:text=Suporte%20de%20quantidade%20de%20dados,vantagem%20dessa%20linguagem%20de%20programa%C3%A7%C3%A3o. Acesso em: 08 nov. 2023.

DATABRICKS. **Transações ACID**. 2023. Disponível em: https://www.databricks.com/br/glossary/acid-transactions. Acesso em: 08 nov. 2023.

FIREBASE. **Cloud Firestore**. 2023. Disponível em: https://firebase.google.com/docs/firestore?hl=pt-br#:~:text=O%20Cloud%20Firestore%20%C3%A9%20um%20banco%20de%20dados%20NoSQL%20hospedado,em%20APIs%20REST%20e%20RPC. Acesso em: 11 set. 2023.

FIREBASE. **Cloud Storage para Firebase**. 2023. Disponível em: https://firebase.google.com/docs/storage?hl=pt-br#:~:text=download%20de%20objetos.-,Quer%20armazenar%20outros%20tipos%20de%20dados%3F,Firebase%20e%20o%20Google%20Cloud. Acesso em: 11 set. 2023.

FIREBASE. **Firebase Authentication**. 2023. Disponível em: https://firebase.google.com/docs/auth?hl=pt-br#:~:text=O%20Firebase%20Authentication%20fornece%20servi%C3%A7os,Facebook%20e%20Twitter%2C%20entre%20outros. Acesso em: 11 set. 2023.

GABARDO, A. C. **Laravel para Ninjas**. São Paulo, SP - Brasil: Novatec Editora Ldta, 2017. 184 p.

GESUAS. **Pesquisa e redesign de sistema de apoio à assistência social**. 2023. Disponível em: https://www.gesuas.com.br/. Acesso em: 30 out. 2023.

GIT, I. **Git is a free and open source distributed version control system.** 2023. Disponível em: https://git-scm.com. Acesso em: 12 set. 2023.

IDS Sistemas - Brasil Gestão Pública. **Pesquisa e redesign de sistema de apoio à assistência social**. 2018. Disponível em: https://old.tog.design/portfolio/ids-social/. Acesso em: 30 out. 2023.

IDS Social. **Prontuário SUAS - Manual de instruções para o registro das informações especificadas**. 2023. Disponível em: https://ids.inf.br/ids-social/#:~:text=O%20IDS%20Social%20garante%20integridade,%C3%A1reas%20de%20Assist%C3%AAncia%20Social%20municipais. Acesso em: 30 out. 2023.

IMPACTA, R. **Diferenças entre SQL e NoSQL: comparando bancos de dados relacionais e não relacionais**. 2023. Disponível em: https://www.impacta.com.br/blog/diferencas-entre-sql-e-nosql-comparando-bancos-de-dados-relacionais-e-nao-relacionais/#:~:text=O%20modelo%20de%20tabela%20relacional,precisam%20seguir%20um%20esquema%20r%C3%ADgido. Acesso em: 08 nov. 2023.

KINSTA, I. **O que é PostgreSQL?** 2023. Disponível em: https://kinsta.com/pt/base-de-conhecimento/o-que-e-postgresql/. Acesso em: 13 jan. 2023.

KRIGER, B. **O que são Migrations e como utilizá-las em seu projeto?** 2023. Disponível em: https://kenzie.com.br/blog/migrations/#:~:text=Migrations%20s%C3%A3o%20uma%20forma%20de,de%20forma%20consistente%20e%20controlada. Acesso em: 10 nov. 2023.

LARAVEL. **Laravel Sail**. 2023. Disponível em: https://laravel.com/docs/10.x/sail. Acesso em: 10 nov. 2023.

LARAVEL. **Laravel Sanctum**. 2023. Disponível em: https://laravel.com/docs/10.x/sanctum. Acesso em: 08 nov. 2023.

LARAVEL. **Laravel Starter Kits**. 2023. Disponível em: https://laravel.com/docs/10.x/starter-kits. Acesso em: 10 nov. 2023.

LUCID, S. I. **O que é diagrama entidade relacionamento**. 2023. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento. Acesso em: 13 set. 2023.

Ministério do desenvolvimento social e combate à fome. **Prontuário SUAS - Manual de instruções para o registro das informações especificadas**. 2013. Disponível em: https://aplicacoes.mds.gov.br/sagi/snas/vigilancia/doc/ManualdePreenchimentoProntuario_versao_preliminar.pdf. Acesso em: 30 out. 2023.

OLIVEIRA, A. de F. G. **O conselho da comunidade e suas interfaces**. 2012. Disponível em: https://meuartigo.brasilescola.uol.com.br/atualidades/o-conselho-comunidade-suas-interfaces.htm. Acesso em: 22 ago. 2023.

ORACLE. **O que é um banco de dados relacional (RDBMS)?** 2023. Disponível em: https://www.oracle.com/br/database/what-is-a-relational-database/. Acesso em: 13 set. 2023.

PHP, T. G. **O que é o PHP?** 2023. Disponível em: https://www.php.net/manual/pt_BR/intro-whatis.php. Acesso em: 12 set. 2023.

PIRES, R. **Aprenda a usar a técnica MoSCoW nos projetos da sua agência**. 2019. Disponível em: https://rockcontent.com/br/blog/metodo-moscow/. Acesso em: 12 set. 2023.

POPOVSKI, I. **Laravel Security Features**. 2019. Disponível em: https://iwconnect.com/laravel-security-features/. Acesso em: 07 nov. 2023.

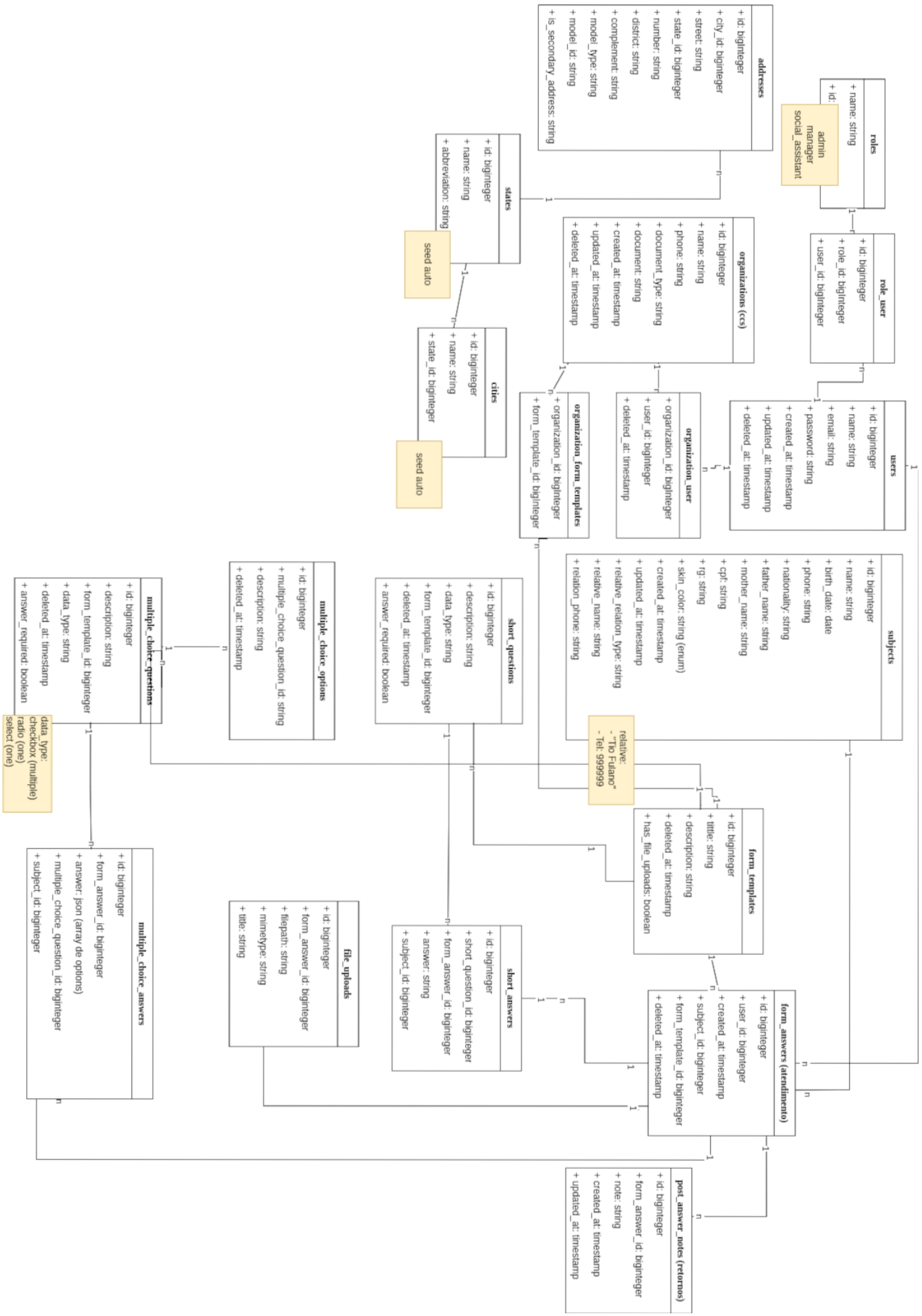
REMESSA, O. **Firebase: descubra para que serve, como funciona e como usar**. 2021. Disponível em: https://www.remessaonline.com.br/blog/firebase-descubra-para-que-serve-como-funciona-e-como-usar/. Acesso em: 05 set. 2023.

ROCHA, A. **Gerenciamento de Logs: como funciona**. 2021. Disponível em: https://www.opservices.com.br/gerenciamento-de-logs/. Acesso em: 07 nov. 2023.

SUAS, R. **Prontuário Eletrônico do SUAS**. 2023. Disponível em: https://aplicacoes.mds.gov.br/prontuario/. Acesso em: 30 out. 2023.

APÊNDICE A – Diagrama do banco de dados

Figura 11 – Diagrama do banco de dados



Fonte: Autoria própria (2023).