CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GUILHERME DO NASCIMENTO E SILVA

MARIA LAURA DE SOUZA E SOUZA

APLICATIVO PARA CADASTRO E CONSULTA DE OCORRÊNCIAS DE DISCRIMINAÇÃO VIA GEOLOCALIZAÇÃO

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Carlos Alberto Lucas

FRANCA/SP

2022

APLICATIVO PARA CADASTRO E CONSULTA DE OCORRÊNCIAS DE DISCRIMINAÇÃO VIA GEOLOCALIZAÇÃO

Guilherme Nascimento Silva[1]

Maria Laura Souza[2]

Resumo

Este trabalho apresenta um aplicativo desenvolvido com o intuito de tornar visível para os usuários e autoridades responsáveis os locais onde mais ocorrem casos de discriminação, assédio ou qualquer tipo de constrangimento na cidade. Incidências desses tipos são frequentes em espaços públicos e privados, e a denúncia não ocorre por insegurança, vergonha e outros motivos. Isso foi confirmado por meio de pesquisa, em forma de formulário, destinada aos alunos do curso de Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Franca Dr. Thomaz Novelino. Com base nesses dados, foi desenvolvido um aplicativo que permite inserir e consultar ocorrências de qualquer forma de preconceito, segregação ou constrangimento; sendo possível visualizar os relatos através de um mapa, filtrando por localização. Além disso, pode-se realizar chamadas para contatos de emergência ou até mesmo para polícia, em caso de necessidade. Espera-se que, após o lançamento do aplicativo, tais ocorrências sejam reduzidas, tornando a cidade um lugar mais seguro para toda a população.

Palavras-chave: Aplicativo. Discriminação. Localização. Segurança.

Abstract

This paper presents an app developed with the intention of highlighting to users and oficial authorities, the city places with the highest incidence of discrimination, harrasement, and other forms of persecussion. Occurrences of that kind are frequent in spaces public and private, and their public reporting often doesn't happen due to fear, shame amongst other reasons. That was confirmed by means of a survey in questionnaire form intended for students of the course of Analysis and Systems Development from Faculdade de Tecnologia Franca Dr. Thomaz Novelino. Based on these findings, the app was developed to allow the insertion and consultation of all reports of bigotry, segregation, and persecution. Beyond that, it is also possible to make calls to emergency contacts or the police in extreme cases. One hopes that, following the launch of the app, occurrences of this kind are reduced, making the city a safer place for all population.

Keywords: *App. Discrimination. Location. Safety*

---

[1] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: guinasc7@gmail.com.

[2] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: mlaura2souza@gmail.com.

1 Introdução

Não é de hoje que os casos de assédio, discriminação e violência, motivados por diferenças étnicas, sexuais, sociais, entre outros; têm se tornado cada vez mais evidentes.

Segundo Bond (2020), em uma publicação disponibilizada pelo Fórum Brasileiro de Segurança Pública, os casos de feminicídio aumentaram em 22,2% entre Março e Abril no ano de 2020. Outro levantamento, apresentado no *Atlas da Violência* (2020), mostra que a taxa de homicídios de pessoas negras cresceu 11,5% entre 2008 e 2018. Uma terceira pesquisa, realizada pela Catho (2020), verificou que mais de 43% das mulheres sofrem assédio moral no ambiente de trabalho. Como apresentado, tornou-se comum a ocorrência destes casos nos mais variados ambientes, tais como na rua, local de trabalho e, até mesmo, dentro das próprias casas isso muitas vezes é notado.

Tendo em vista este cenário, buscou-se reunir dados, por meio de uma pesquisa, com o propósito de responder às seguintes questões problemas: Levando em conta o alto índice de assédio, discriminação e violência motivados por diversos fatores, como minimizar e evidenciar esses casos para segurança da sociedade? Como aumentar a segurança em toda a cidade e evitar que casos de assédio/discriminação continuem se propagando? Em uma situação de risco, como prover meios para que a vítima consiga escapar em segurança?

Nesse sentido, visando a redução destas ocorrências e a possibilidade de tornar a nossa cidade mais segura para toda a população, este trabalho tem como objetivo o desenvolvimento de um aplicativo para a inclusão e consulta de ocorrências de qualquer forma de preconceito, segregação ou constrangimento; tais informações poderão ser visualizadas em um mapa, podendo ser filtradas por endereço. O aplicativo também contará com uma opção para que o usuário possa realizar chamadas para contatos de emergência, em caso de necessidade. Dessa forma, os usuários e autoridades responsáveis poderão visualizar em quais locais os casos ocorrem com mais frequência.

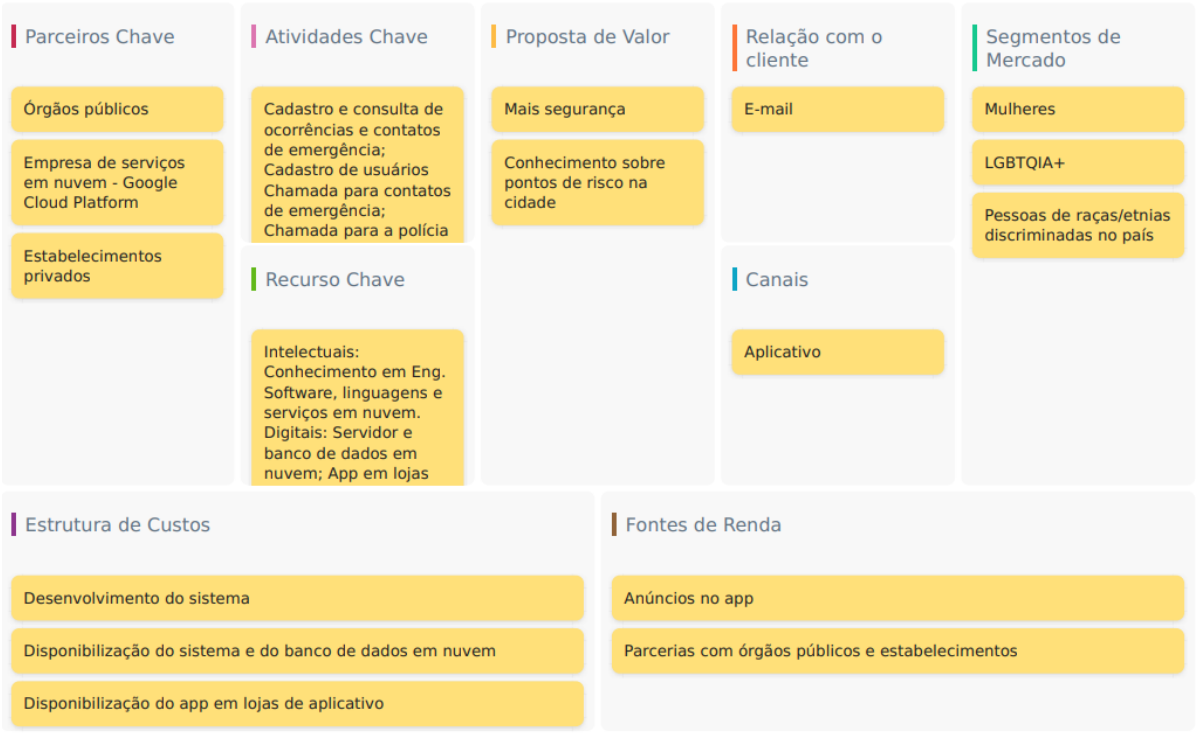
2 Viabilidade do projeto

Para validar a viabilidade e todos os pontos fundamentais do projeto, foi utilizado o *Business Model Canvas*. Este diagrama representa um modelo de negócios de forma interativa, possibilitando a visualização do negócio e do encaixe estratégico entre suas diferentes áreas em uma única página.

Para Pereira (2016), à primeira vista o *Business Model Canvas* parece um simples diagrama. A facilidade de análise que ele traz só passa a ser realmente compreendida à medida que sua visualização e utilização passam a ser aplicadas no dia-a-dia.

Na Figura 1 é possível visualizar o modelo de negócios desenvolvido para este projeto.

Figura 1 - Modelo de Negócios Canvas



Fonte: Produzido pelos autores

3 Levantamento de Requisitos

Esta seção apresenta a técnica utilizada para a elicitação e especificação dos requisitos, a modelagem dos processos da aplicação, a definição das

funcionalidades, bem como as regras de negócio, a representação das ações dos usuários no sistema e também a modelagem de Banco de Dados.

3.1 Elicitação e especificação dos Requisitos

Segundo Pressman (2011), requisitos são características que o sistema deve possuir para atender o que o cliente deseja, buscando identificar e resolver de maneira viável os problemas da organização.

O levantamento dos requisitos, bem como a validação da ideia, foram realizados por meio de uma pesquisa, em formato de formulário, com os alunos do curso de Análise e Desenvolvimento de Sistemas da Fatec Franca; onde foram sugeridos uma rede de denúncias para facilitar o processo, a necessidade da disponibilização de informações sobre casos de assédio/discriminação em estabelecimentos e locais abertos e meios para que a pessoa possa se sentir mais segura nos mais determinados locais. Os detalhes relacionados à pesquisa serão abordados no capítulo 4.

3.2 BPMN

O BPMN (*Business Process Model and Notation*) é uma representação que permite a visualização de processos graficamente, através de diagramas. A partir desta representação, é possível modelar diferentes fluxos de processos em vários níveis de detalhamento, por meio de um conjunto de símbolos e regras.

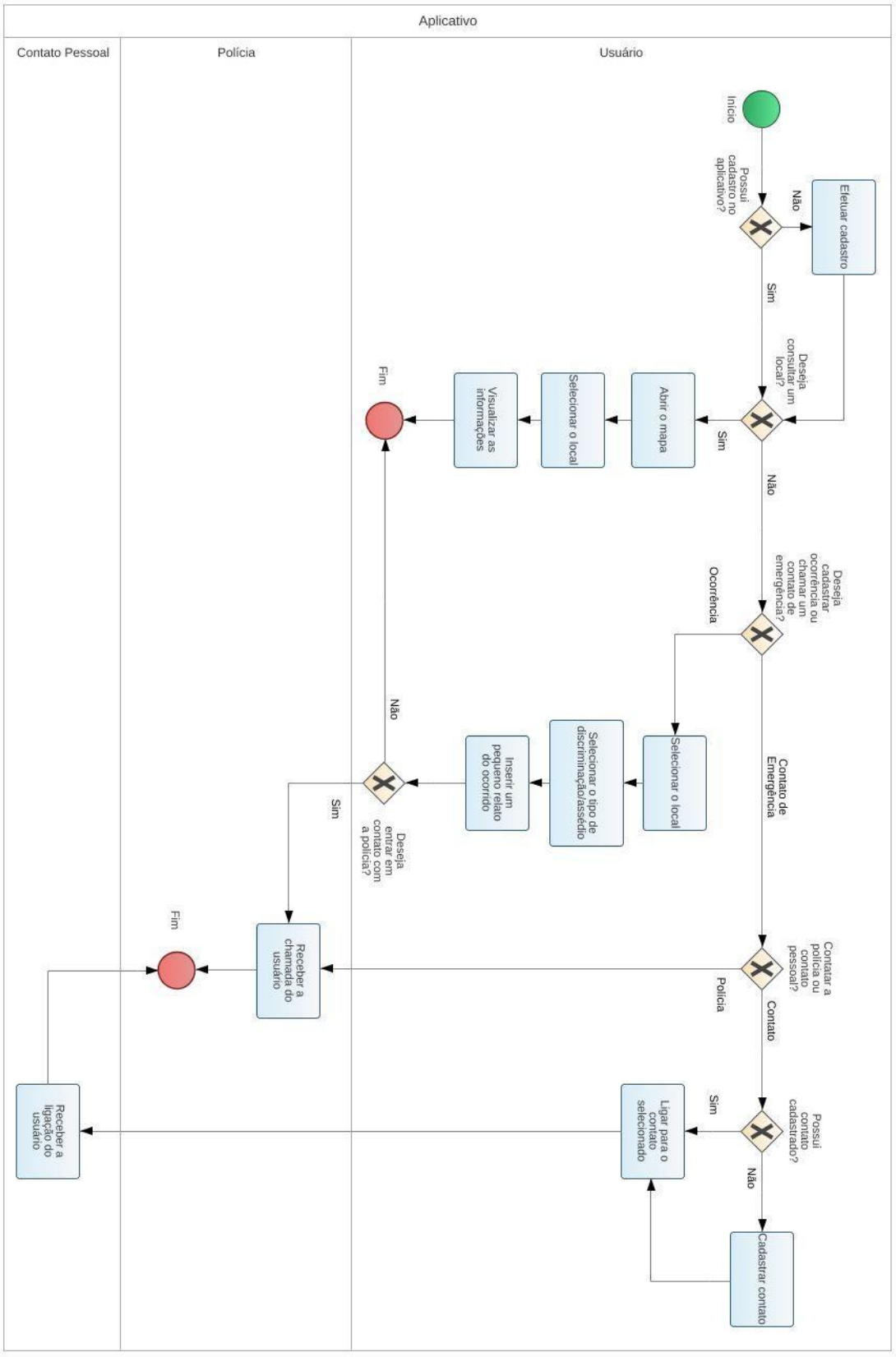
Vale ressaltar que o nível de detalhamento do processo varia de acordo com o objetivo da modelagem, porém o ideal é que os processos sejam modelados em uma visão ponta a ponta. Desta maneira, é possível observar todo o fluxo de forma holística, compreendendo que o resultado é fruto do envolvimento de diversos setores, gerando valor ao cliente.

Sobre o BPMN, Reis (2008) destaca que, além de mais moderna que notações como IDEF e UML, também possui um rico *set* de elementos gráficos para representação de uma série de situações que acontecem nos fluxos dos processos.

Algumas vantagens na utilização desta ferramenta são: comunicação - o BPMN atua com uma comunicação universal, pois utiliza uma linguagem compreendida por todos os envolvidos no processo; versatilidade - pois pode ser aplicada a diversos tipos diferentes de processos.

A Figura 2 apresenta o BPMN desenvolvido para este projeto.

Figura 2 - BPMN do sistema



Fonte: Produzido pelos autores

3.3 Requisitos Funcionais

Segundo Sommerville (2011), requisitos funcionais são declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações.

O documento de requisitos, de acordo com Paula Filho (2019), deve ser preenchido informando o ID do Requisito, como por exemplo "[RF001]" ou "[RNF001]" e assim sucessivamente conforme a criação de novos requisitos, junto do Nome do Requisito, escrito de forma clara e com o verbo no infinitivo. Deve conter também uma descrição altamente detalhada explicando a funcionalidade do requisito. Além disso, é preciso também indicar a categoria para identificar se o requisito é evidente para o usuário ou não; e apontar também qual a sua prioridade.

Quadro 1 – Requisitos Funcionais do sistema

RF001- Cadastrar usuário	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário cadastre seu perfil com informações básicas, como nome do perfil, e-mail e senha.		
RF002- Cadastrar contatos de emergência	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário cadastre até 2 contatos de emergência de sua escolha. Cada contato deve conter nome e número de telefone.		
RF003- Fazer ligação	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve conter a opção de realizar ligações para os contatos de emergência cadastrados ou para a polícia.		
RF004- Consultar informações em determinado local	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário consulte os índices de ocorrências de determinado local.		

RF005- Registrar ocorrência	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário registre uma ocorrência em determinada localização. A ocorrência deverá ser classificada de acordo com as categorias existentes: assédio, racismo, machismo, xenofobia, transfobia e outros; e também será necessário preencher a data e horário do ocorrido.		

3.4 Requisitos Não Funcionais

Para Sommerville (2011) requisitos não funcionais são restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de *timing*, restrições no processo de desenvolvimento e restrições impostas pelas normas.

Quadro 2 – Requisitos Não Funcionais do sistema

RNF001- Aplicativo	O sistema será desenvolvido exclusivamente no formato de aplicativo mobile.	Tipo Plataforma	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF002- Funcionamento com GPS	O sistema só deverá funcionar tendo o GPS do smartphone ativo, para que a aplicação possa buscar o local de origem do usuário.	Tipo Funcionamento	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF003- Conexão com internet	O sistema necessita de conexão com a internet para o seu funcionamento	Tipo Conexão	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF004- Linguagem	O sistema será desenvolvido na linguagem Dart, com o framework Flutter.	Tipo Linguagem	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF005- Usabilidade	As telas desenvolvidas devem ser intuitivas e possibilitar uma fácil compreensão sobre as funcionalidades e os caminhos que o usuário pode percorrer.	Tipo Usabilidade	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF006- Banco de dados	Todas as informações serão armazenadas em banco de dados relacional MySQL	Tipo Armazenamento	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF007- Plataforma	O sistema será compatível com dispositivos Android e iOS.	Tipo Compatibilidade	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

RNF008- Consulta de ocorrências	O usuário poderá realizar consultas de ocorrências apenas se estiver logado.	Tipo Permissão	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

Matriz de Rastreabilidade RF x RNF

Tabela 1 - Matriz de Rastreabilidade entre os requisitos do sistema

	RF001	RF002	RF003	RF004	RF005
RNF001	X	X	X	X	X
RNF002				X	X
RNF003	X	X	X	X	X
RNF004	X	X	X	X	X
RNF005	X	X	X	X	X
RNF006	X	X	X	X	X
RNF007	X	X	X	X	X
RNF008				X	

3.5 Regras de Negócio

Ainda de acordo com Paula Filho (2019), as Regras de Negócio são restrições impostas pelo negócio que determinam o comportamento de um procedimento operacional.

Quadro 3 – Regras de Negócio do sistema.

RN001 - Contato da polícia
Descrição: O sistema deverá ter o contato da polícia cadastrado para todos os usuários como contato de emergência.
RN002 - Apenas um cadastro por usuário
Descrição: O sistema deverá permitir apenas um cadastro por e-mail e telefone.
RN003 - Contatos de Emergência
Descrição: O sistema deverá permitir o cadastro de até dois contatos de emergência, além do contato da polícia, previamente cadastrado.
RN004 - Consulta de ocorrências
Descrição: O sistema pesquisará o local desejado através da geolocalização do dispositivo, utilizando um serviço de pesquisa e visualização de mapas.
RN005 - Registro de ocorrência

Descrição: O sistema registrará a ocorrência inserida pelo usuário no banco de dados e deverá mostrar a opção para a chamada com a polícia ou contato de emergência após este registro.
RN006 - Opções após o login
Descrição: Somente após o usuário efetuar o login, o sistema deverá habilitar as funcionalidades existentes no aplicativo

Matriz de Rastreabilidade RF X RN

Tabela 2 - Matriz de Rastreabilidade entre os requisitos e regras do sistema

	RF001	RF002	RF003	RF004	RF005
RN001			X		
RN002	X				
RN003		X			
RN004				X	
RN005					X
RN006	X	X	X	X	X

3.6 Casos de Uso

Índice de casos de uso e Diagrama de casos de uso

UC001 - Cadastrar usuário

UC002 - Consultar ocorrências de um local

UC003 - Selecionar o local

UC004 - Registrar ocorrência

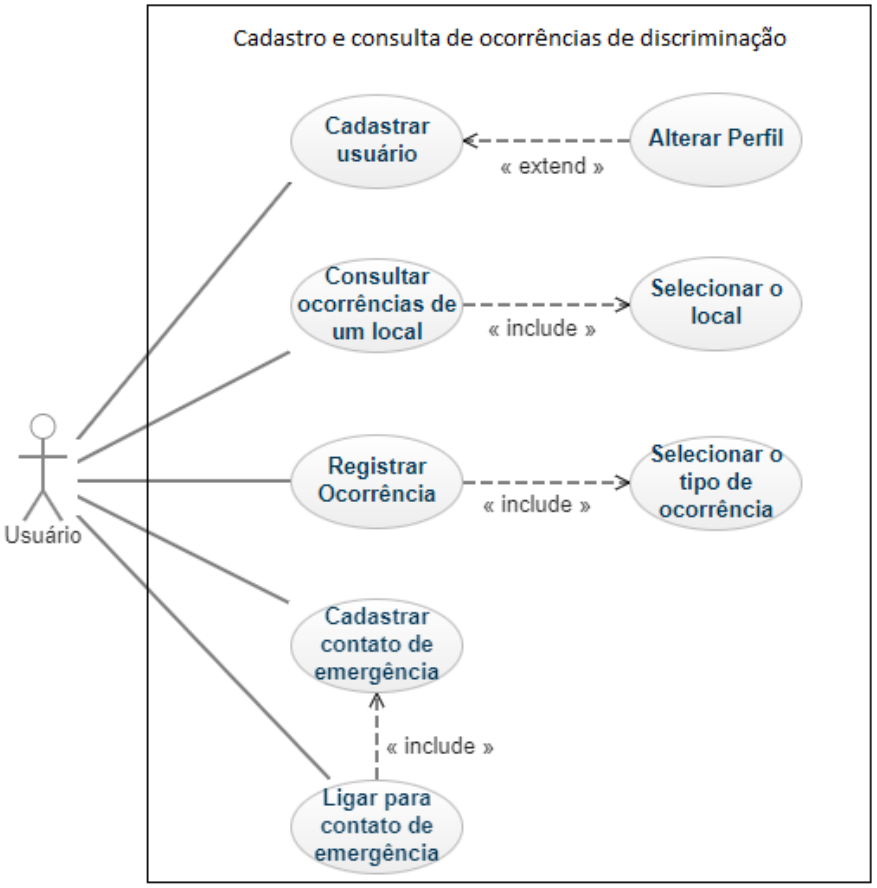
UC005 - Selecionar o tipo de ocorrência

UC006 - Cadastrar contato de emergência

UC007 - Fazer ligação para contato de emergência

UC008 - Alterar Perfil

Figura 3 - Diagrama de Casos de Uso do sistema



Fonte: Produzido pelos autores

Especificação de cada um dos casos de uso

Quadro 4 – Use Case Cadastrar Usuários

Caso de Uso – Cadastrar usuário	
ID	UC 001
Descrição	Este caso de uso tem por objetivo cadastrar o usuário.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção “Cadastrar”. 2. O sistema carrega o formulário de cadastro de usuário. 3. O usuário informa o nome, e-mail, senha e número de telefone. 4. O sistema salva os dados. 5. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	3a – Nome de perfil não foi preenchido

	3a. O sistema informa ao usuário de que o atributo nome de perfil é obrigatório. 3b – E-mail não foi preenchido 3b. O sistema informa ao usuário de que o atributo e-mail é obrigatório. 3c – Senha não foi preenchido 3c. O sistema informa ao usuário de que o atributo senha é obrigatório. 3d – Número de telefone não foi preenchido 3d. O sistema informa ao usuário de que o atributo número de telefone é obrigatório.

Quadro 5 – Use Case Consultar ocorrências

Caso de Uso – Consultar ocorrências de um local	
ID	UC 002
Descrição	Este caso de uso tem por objetivo consultar ocorrências de um determinado local.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção de “consultar ocorrências”. 2. O sistema inclui o UC003 - Selecionar local 3. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	2a – O sistema inclui o cenário alternativo do UC003 - Selecionar local

Quadro 6 – Use Case Selecionar local

Caso de Uso – Selecionar local	
ID	UC 003
Descrição	Este caso de uso tem por objetivo selecionar um determinado local para visualizar ocorrências registradas no mesmo
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção para consultar ocorrências. 2. O sistema carrega o mapa, conforme localização do dispositivo, e o apresenta na tela. 3. O usuário seleciona o local digitando o endereço desejado. 4. O sistema carrega as ocorrências registradas nas proximidades do local escolhido, apresentando a opção para que o usuário possa selecionar e visualizar o tipo de ocorrência, a data e hora. 5. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	2a – A localização do dispositivo está desligada. 2a. O sistema informa ao usuário de que a localização deve estar ligada para uma melhor performance durante o processo. 3a – Usuário digita o endereço desejado. 3a. O sistema considera somente as ocorrências registradas nas proximidades do endereço digitado.

Quadro 7 – Use Case Registrar ocorrência

Caso de Uso – Registrar ocorrência	
ID	UC 004
Descrição	Este caso de uso tem por objetivo registrar ocorrências de discriminação/assédio baseando-se na localização do evento.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção para cadastrar ocorrência; 2. O sistema carrega o formulário para que o usuário insira o endereço ou selecione a localização atual, a data e hora; 3. O sistema sistema carrega os tipos de ocorrências, e os apresenta na tela. 4. O sistema inclui o UC005 - Selecionar o tipo de ocorrência; 5. O sistema também carrega um campo, não obrigatório, para que o usuário possa inserir a descrição do caso. 6. O sistema disponibiliza as opções para que o usuário possa entrar em contato com a polícia, caso necessário, após registrar a ocorrência; ou apenas salvar a ocorrência.

	7. O usuário seleciona a opção desejada. 8. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	2a – Local não preenchido. 2a. O sistema informa ao usuário de que o atributo local é obrigatório. 2b – Data não preenchida. 2b.O sistema informa ao usuário de que o atributo data é obrigatório. 2c – Hora não preenchida. 2c. O sistema informa ao usuário de que o atributo hora é obrigatório. 4a – Usuário não seleciona um tipo de ocorrência. 4a. O sistema informa ao usuário de que é necessário informar o tipo de ocorrência para prosseguir. 6a – Usuário seleciona a opção para entrar em contato com a polícia. 6a. O sistema inicia a chamada com o contato da polícia, já cadastrado previamente como contato de emergência.

Quadro 8 – Use Case Selecionar o tipo de ocorrência

Caso de Uso – Selecionar o tipo de ocorrência	
ID	UC 005
Descrição	Este caso de uso tem por objetivo selecionar o tipo de ocorrência.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção de tipo de ocorrência; 2. O sistema carrega os tipos de ocorrências para escolha: assédio, racismo, machismo, xenofobia, transfobia, entre outros; 3. O usuário seleciona a opção deseja; 4. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	3a – Tipo de ocorrência não informado 3a. O sistema informa ao usuário de que é obrigatório selecionar algum tipo de ocorrência.

Quadro 9 – Use Case Cadastrar contato de emergência

Caso de Uso – Cadastrar contato de emergência	
ID	UC 006
Descrição	Este caso de uso tem por objetivo cadastrar contatos de emergência.
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção “cadastrar contato de emergência”.

	2. O usuário digita o nome e o número de telefone do seu contato. 3. O usuário seleciona a opção “salvar”. 4. O sistema salva os dados informados. 5. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	1a – Limite de contatos 1a. O sistema informa ao usuário de que é permitido apenas cadastrar dois contatos de emergência na tentativa do usuário cadastrar um terceiro número. 3a – Nome ou o número de telefone não preenchido 3a. O sistema informa que é obrigatório o preenchimento dos atributos: nome ou número de telefone. 3b – Contatos duplicados 3b. O sistema informa ao usuário que é obrigatório o cadastro de dois contatos de emergência diferentes.

Quadro 10 – Use Case Fazer ligação para contato de emergência

Caso de Uso – Fazer ligação para contato de emergência	
ID	UC 007
Descrição	Este caso de uso tem por objetivo fazer ligação para o contato de emergência previamente cadastrado
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção “chamar contato de emergência”. 2. O usuário seleciona o contato que deseja ligar; 3. O sistema realiza uma ligação para o contato selecionado; 4. O sistema encerra o use case.
Pós-condição	Nenhuma
Cenário Alternativo	2a – Contato de emergência não cadastrado 2a. O sistema informa ao usuário que é obrigatório o cadastro do contato de emergência.

Quadro 11 – Use Case Alterar perfil

Caso de Uso – Alterar perfil	
ID	UC 008
Descrição	Este caso de uso tem por objetivo alterar o perfil do usuário cadastrado
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma

Cenário Principal	1. O use case inicia quando o usuário acessa o seu perfil 2. O sistema carrega o formulário de cadastro de usuário com os dados preenchidos 3. O usuário pode editar o nome de perfil, número de telefone e senha. 4. O sistema salva os dados 5. O sistema encerra o use case
Pós-condição	Nenhuma
Cenário Alternativo	3a – Nome de perfil não foi preenchido 3a. O sistema informa ao usuário de que o atributo nome de perfil é obrigatório. 3b – Número de telefone não foi preenchido 3b. O sistema informa ao usuário de que o atributo número de telefone é obrigatório. 3c – Senha não foi preenchida 3c. O sistema informa ao usuário de que o atributo senha é obrigatório.

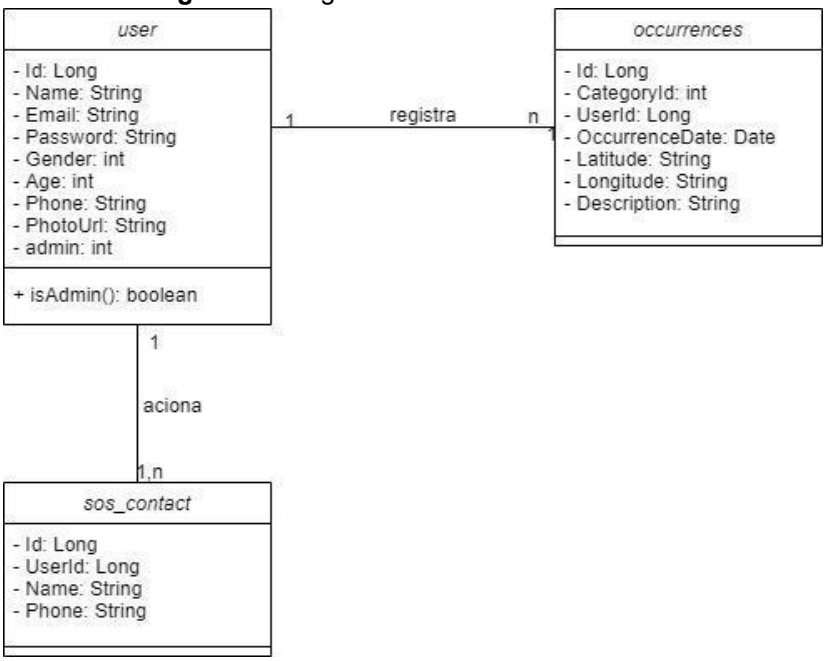
Matriz de Rastreabilidade RF x UC

Tabela 3 - Matriz de Rastreabilidade entre os requisitos e casos de uso do sistema

	RF001	RF002	RF003	RF004	RF005
UC001	X				
UC002				X	
UC003					
UC004					X
UC005				X	
UC006		X			
UC007		X	X		
UC008	X				

3.7 Diagrama de Classes

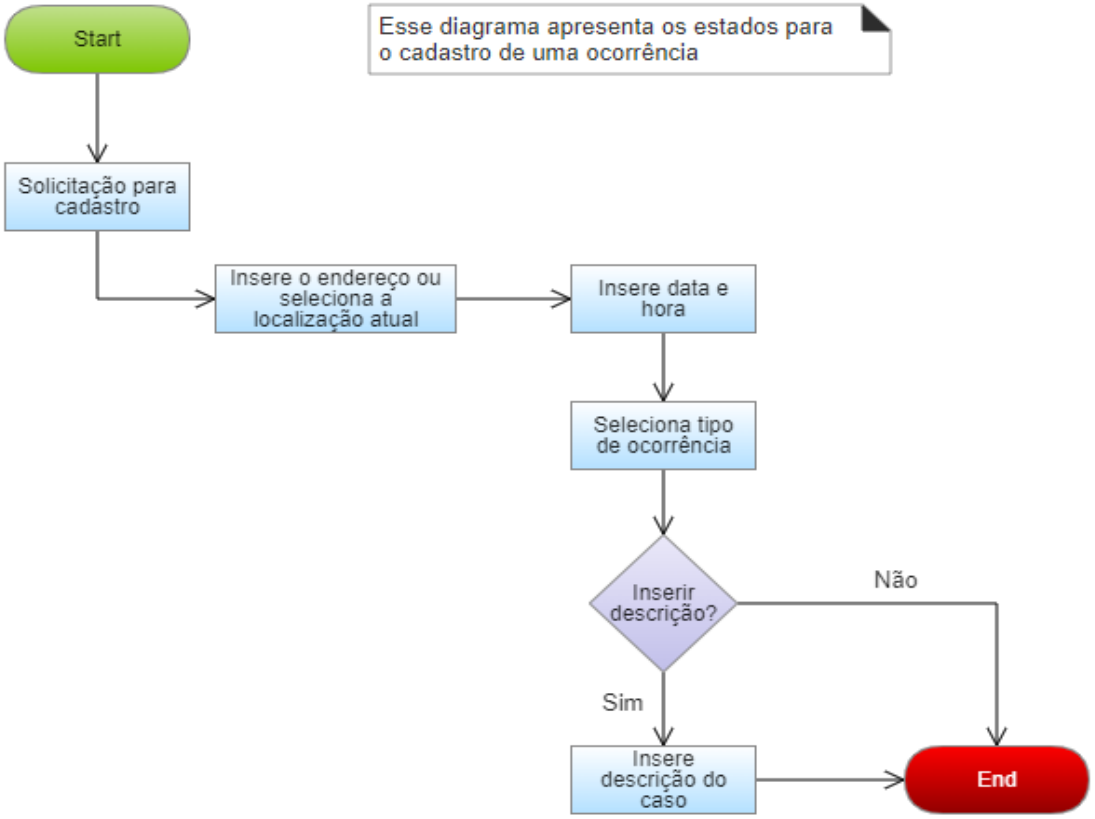
Figura 4 - Diagrama de Classes do sistema



Fonte: Produzido pelos autores

3.8 Diagrama de Estados

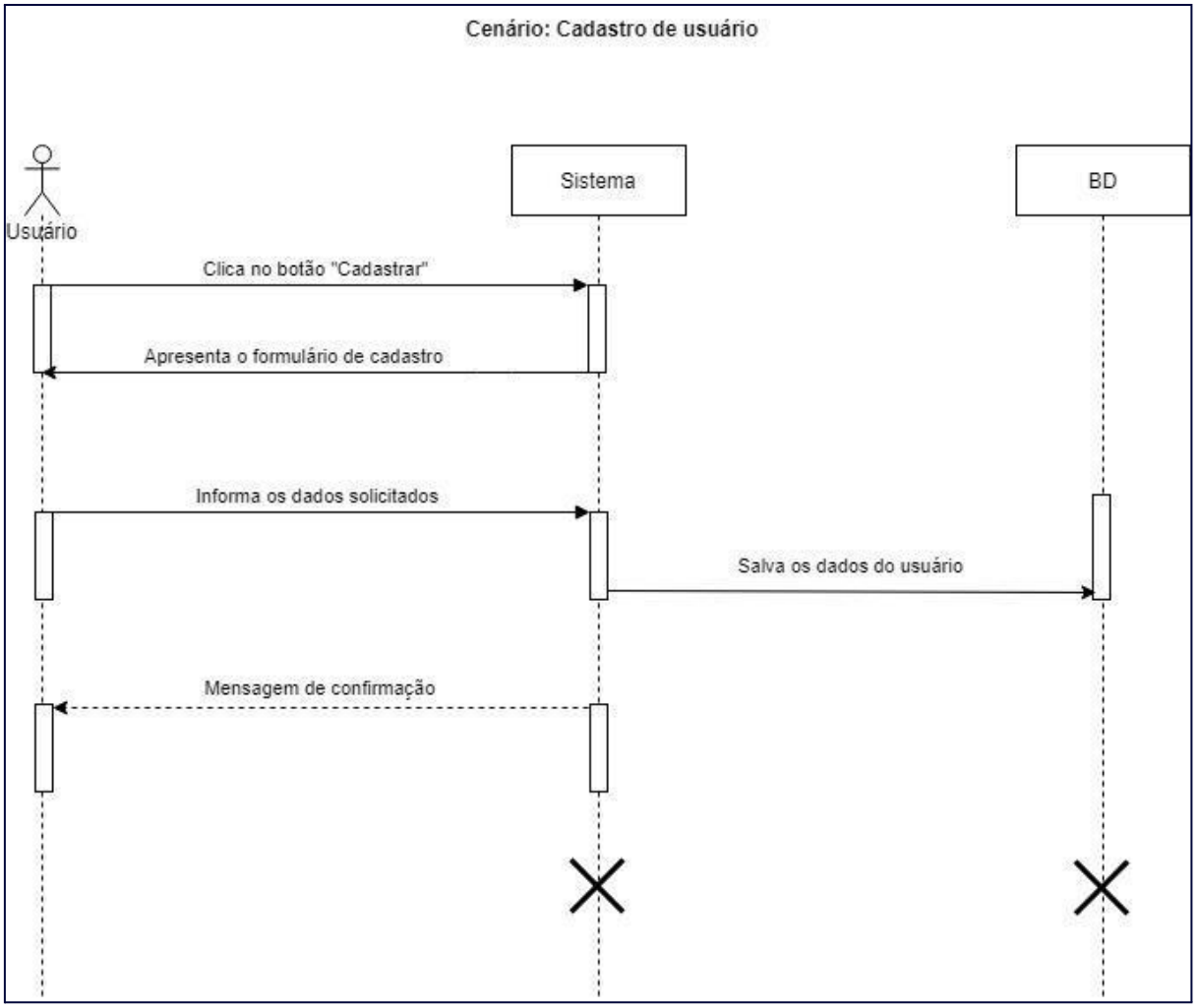
Figura 5 - Diagrama de Estados do sistema



Fonte: Produzido pelos autores

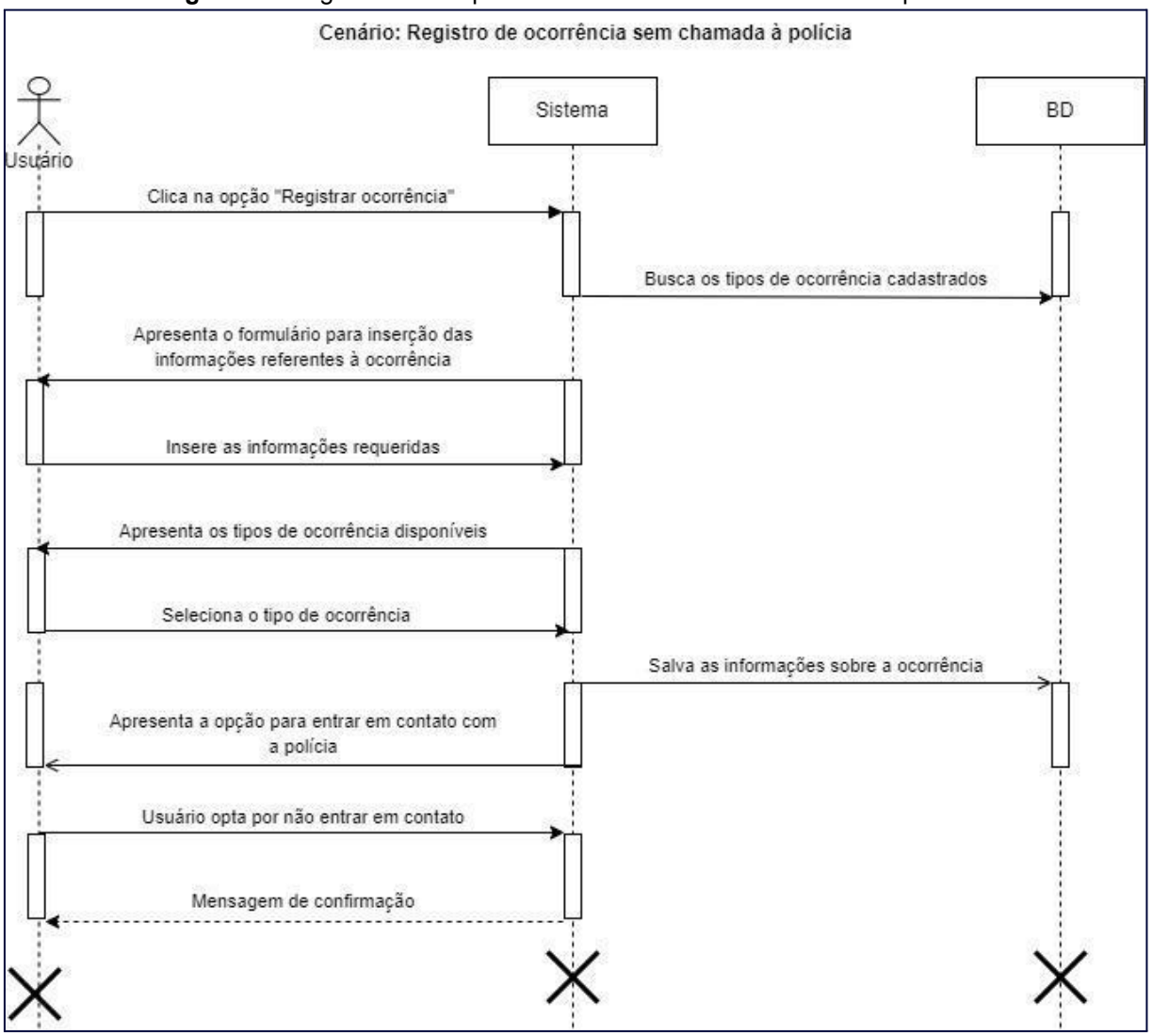
3.9 Diagrama de Sequência

Figura 6 - Diagrama de Sequência 1: Cadastro de Usuário



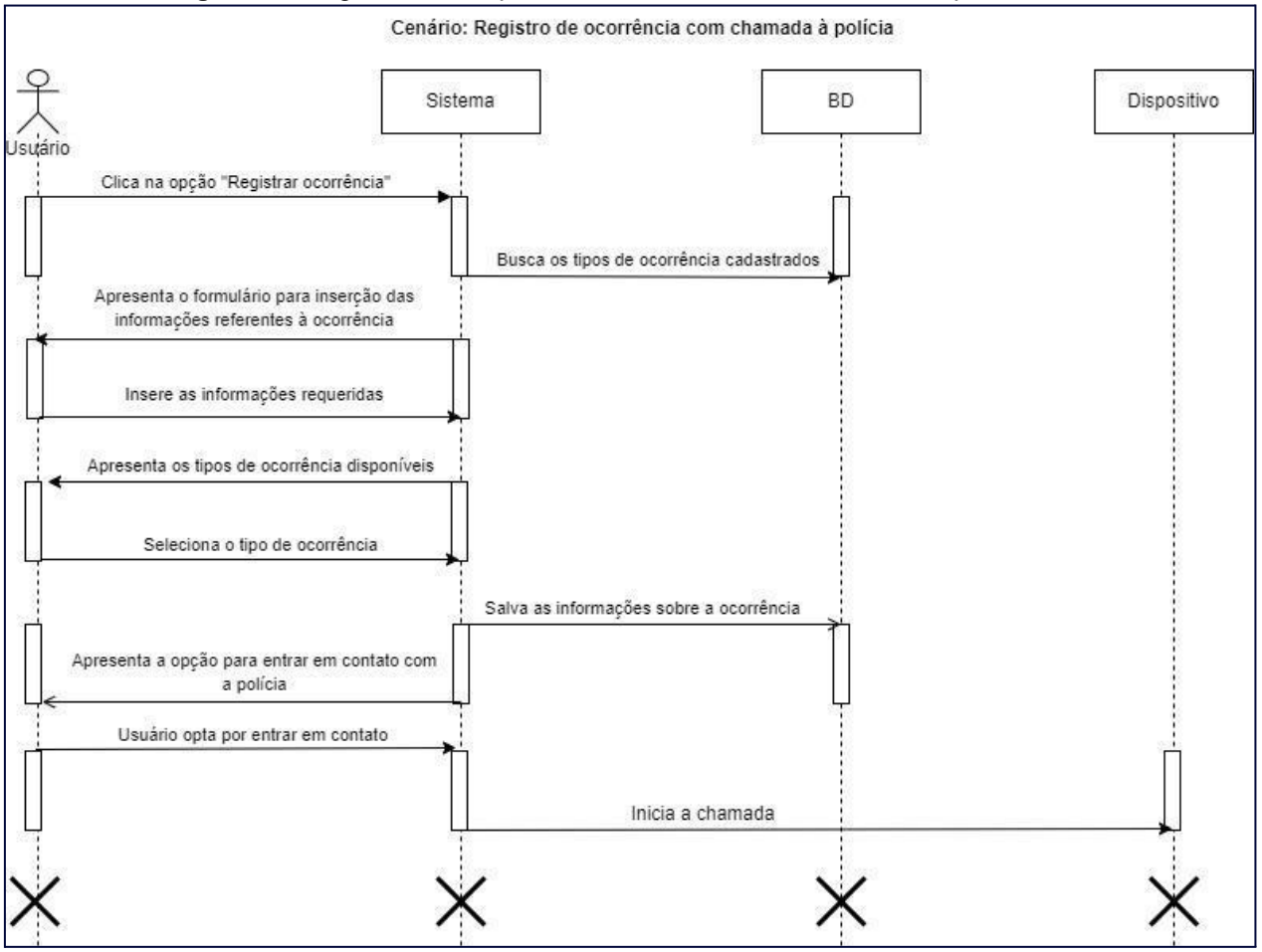
Fonte: Produzido pelos autores

Figura 7 - Diagrama de Sequência 2: Ocorrência sem chamada à polícia



Fonte: Produzido pelos autores

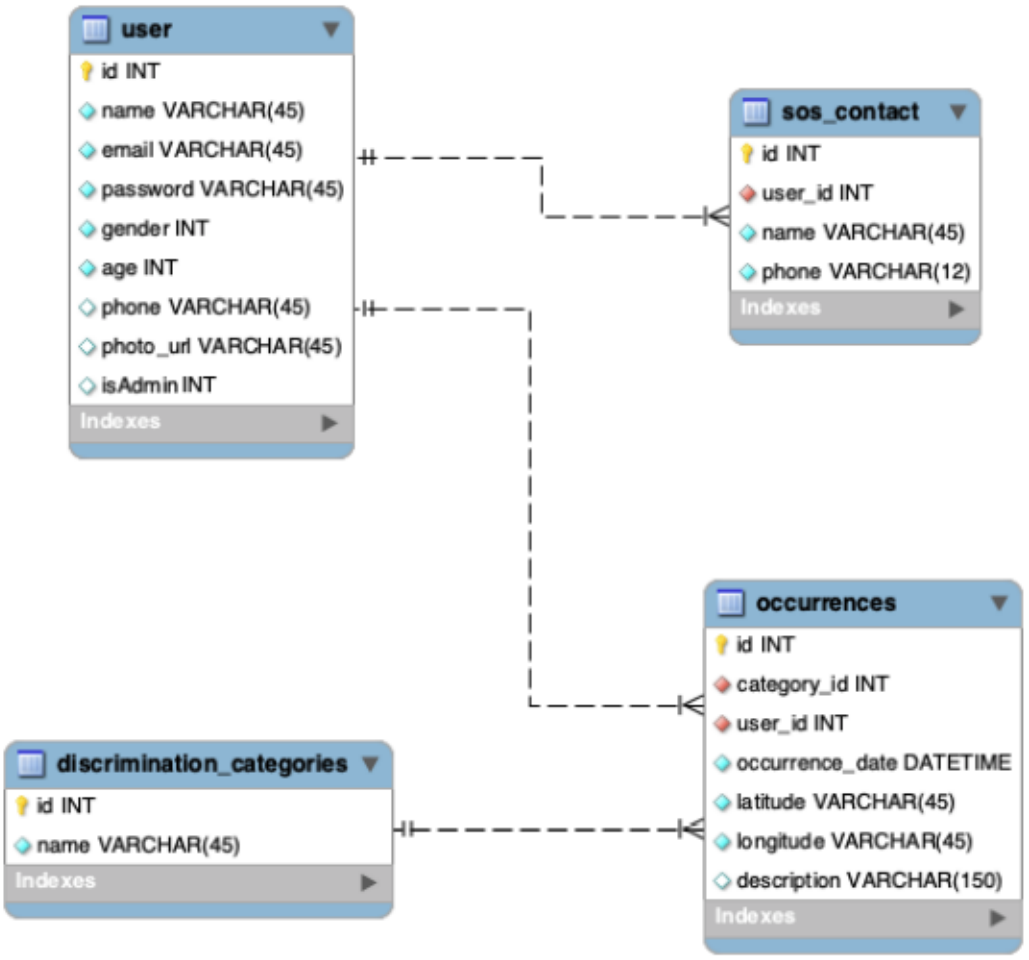
Figura 8 - Diagrama de Sequência 3: Ocorrência com chamada à polícia



Fonte: Produzido pelos autores

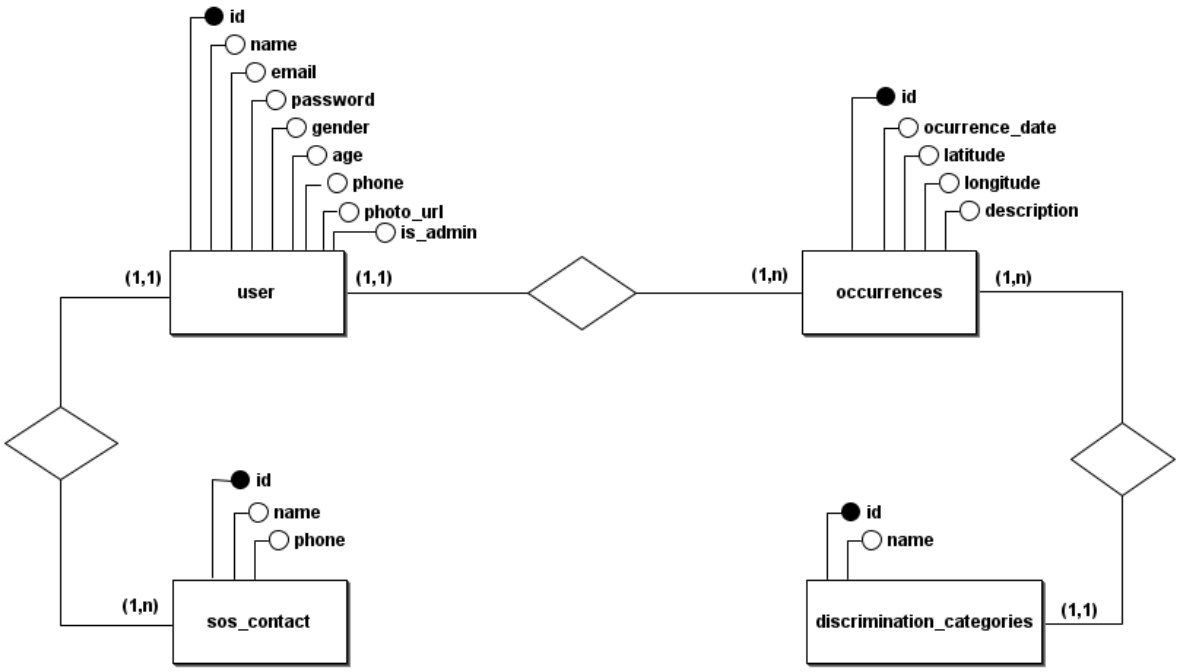
3.10 Modelagem do Banco de Dados

Figura 9 - Diagrama Entidade-Relacionamento: Modelo Lógico



Fonte: Produzido pelos autores

Figura 10 - Diagrama Entidade-Relacionamento: Modelo Conceitual



Fonte: Produzido pelos autores

4 Validação

Esta seção contempla os detalhes dos resultados obtidos na pesquisa aplicada para levantamento dos requisitos e validação da ideia.

4.1 Amostragem

A ideia central do projeto foi validada por meio de uma pesquisa destinada aos alunos do curso de Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Franca Dr. Thomaz Novelino, realizada entre os dias 25/03/2022 e 27/03/2022. A pesquisa foi enviada para os 12 representantes de classe do curso e 9 confirmaram o encaminhamento da mesma às suas respectivas turmas. Dentre as turmas alcançadas, foi efetuado um levantamento da quantidade total de alunos junto aos seus representantes e, a partir deste, constatou-se que a pesquisa foi recebida por 264 alunos. Dentre o total de alunos que receberam a pesquisa, 162 responderam. Vale ressaltar que, para um nível de confiança de 95%, foi necessário que a pesquisa obtivesse ao mínimo 160 respostas.

Segundo (Pastana e Abreu, 2022), é possível obter o valor do tamanho mínimo da amostra através da equação $n = \frac{N\ x\ n_0}{N + n_0}$, onde n representa o tamanho da amostra, N é o tamanho da população e n_0 é a primeira aproximação.

O primeiro passo para a resolução é encontrar o valor de n_0 através da equação $n_0 = \frac{1}{E_0^2}$, onde E_0 é o erro amostral considerado. Para um erro de 5%, tem-se $n_0 = \frac{1}{0{,}05^2} => 400$. Com o valor da primeira aproximação, é possível calcular o tamanho mínimo para a amostragem. Desta forma, $n = \frac{264\ x\ 400}{264 + 400} => 159{,}04$ ou seja, tem-se um tamanho mínimo de aproximadamente 160 respondentes.

Visando facilitar a visualização e com base na idade dos participantes, a quantidade total foi segmentada por faixa etária, através do cálculo da variável quantitativa contínua.

Para esta finalidade, a idade de cada participante foi ordenada de forma ascendente e notou-se que o menor valor deste conjunto era igual a 17 e o maior era igual a 58.

A construção da tabela para a variável quantitativa contínua é realizada em três passos:

1. Cálculo da Amplitude Total de uma sequência, através da equação $At = X_{max} - X_{min}$; onde At é a Amplitude Total a ser calculada, X_{max} e X_{min} são, respectivamente, o maior e menor valor do conjunto. Neste caso, $At = 58 - 17 => 41$.
2. Cálculo da quantidade de classes, pela equação $K = \sqrt{n}$, onde n é a quantidade total de valores do conjunto. Neste caso, $K = \sqrt{162} => 12{,}73$. Para a definição da quantidade de classes, pode-se escolher o melhor valor entre o conjunto (k-1, k , k+1).
3. Cálculo do intervalo de classes $\frac{At}{k}$. Vale ressaltar que para este cálculo, pode-se utilizar o maior valor próximo a At que seja divisível por algum dos valores do conjunto de k. Neste caso, será utilizado o 44 (maior valor após o 41) que é divisível por 11 (k-1). Desta forma,

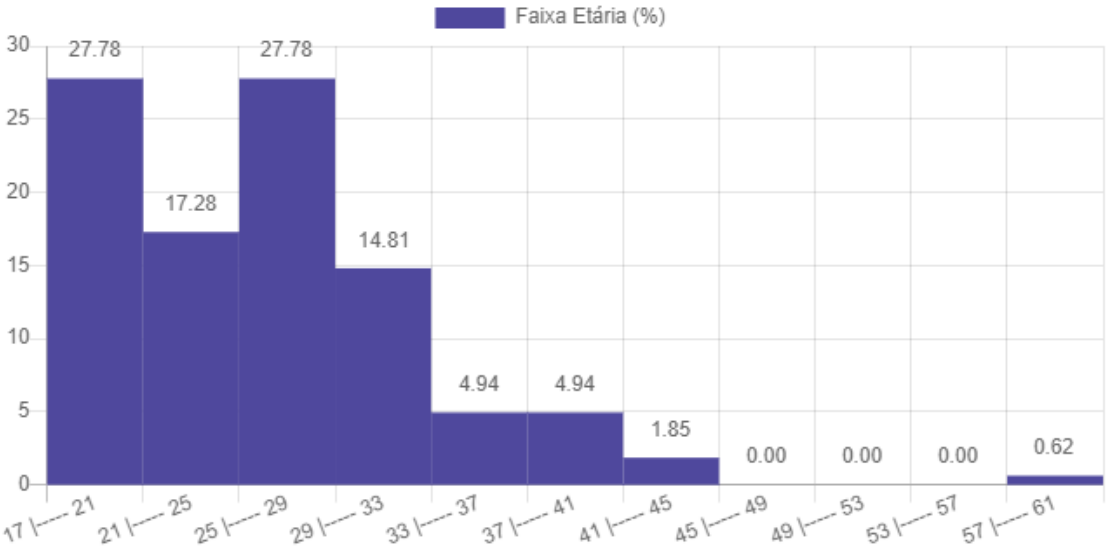
tem-se $\frac{44}{11} = 4$. Ou seja, a tabela da faixa etária terá onze classes com intervalos de 4 números para cada.

Tabela 4 - Tabela de frequência com a faixa etária dos participantes

Faixa Etária	Frequência (Fi)
17 |---------- 21	45
21 |---------- 25	28
25 |---------- 29	45
29 |---------- 33	24
33 |---------- 37	08
37 |---------- 41	08
41 |---------- 45	03
45 |---------- 49	00
49 |---------- 53	00
53 |---------- 57	00
57 |---------- 61	01
Total	162

O histograma referente à faixa etária dos participantes, baseado na tabela anterior, pode ser conferido a seguir.

Gráfico 1 - Faixa etária dos entrevistados



Fonte: Produzido pelos autores

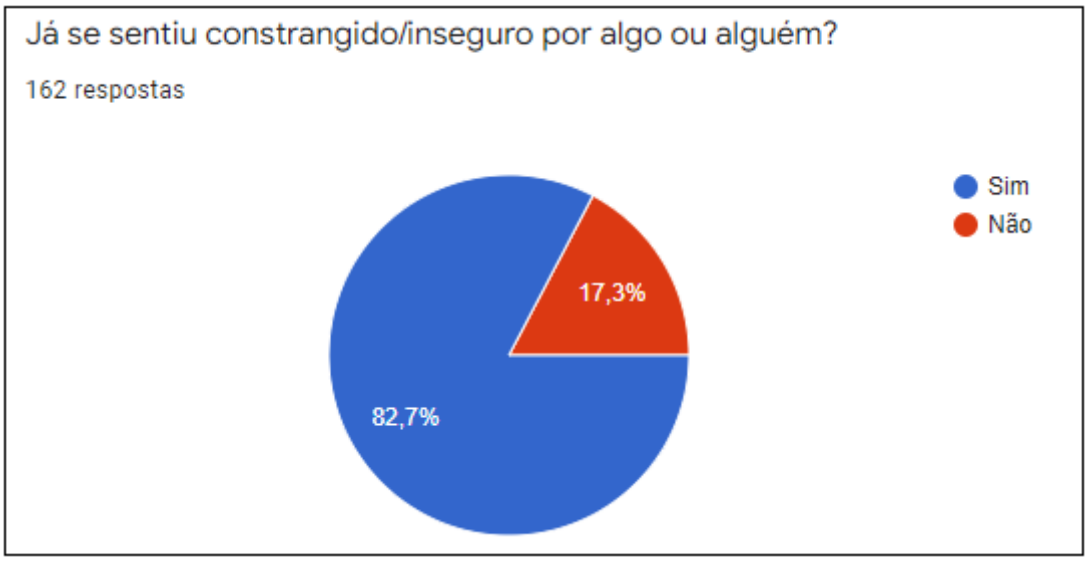
4.2 Resultados

Esta seção apresenta os resultados provenientes da pesquisa aplicada para validar a ideia do projeto. Para compor o questionário foram elaboradas oito perguntas para identificar, na percepção dos entrevistados, temas relacionados a segurança e discriminação. O formulário pode ser acessado através do link: https://docs.google.com/forms/d/e/1FAIpQLSfR9hKGxEzj3n9cpQup5FXpKE6HdQS6fw4NNgQUiOTkltyHDw/viewform; ou pelo *QR Code* abaixo:

O gráfico 2 demonstra o resultado obtido quanto ao questionamento: “Já se sentiu constrangido/inseguro por algo ou alguém?”. Nota-se que 82,7%

responderam que sim, enquanto que apenas 17,3% disseram que não. Vale ressaltar que nesta pergunta, 100% dos participantes responderam:

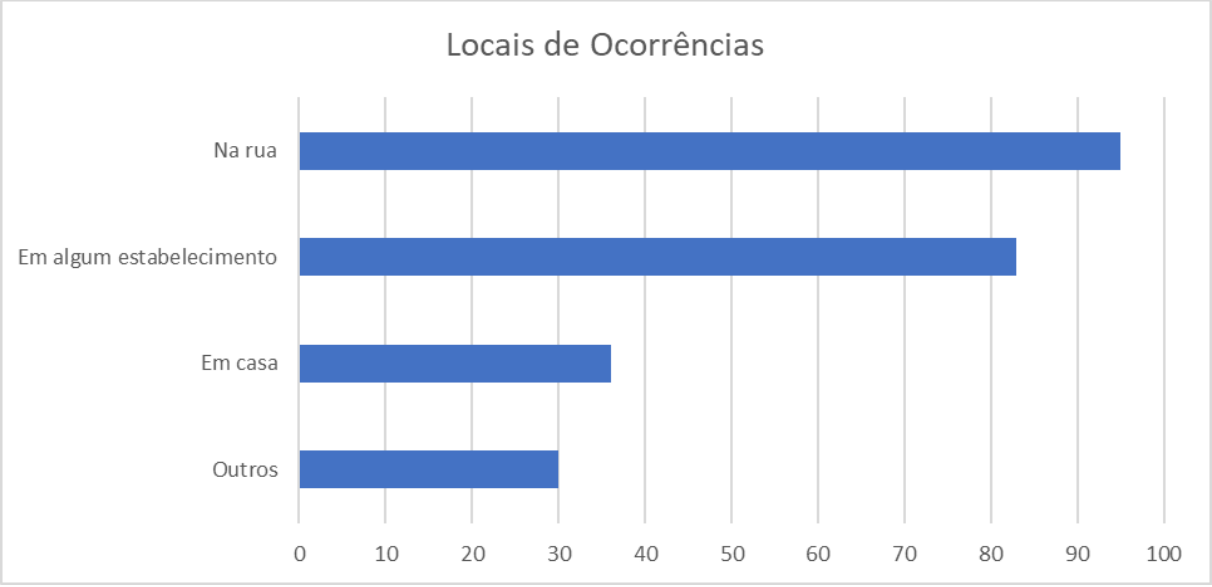
Gráfico 2 - Proporção de entrevistados que já se sentiram constrangidos/inseguros por algo/alguém



Fonte: Gerado automaticamente pelo Google Forms

Na sequência, conforme ilustra o gráfico abaixo, os entrevistados que já se sentiram inseguros ou constrangidos em determinadas situações, compartilharam em quais locais ocorreram esses casos:

Gráfico 3 - Locais de ocorrências que os entrevistados já se sentiram inseguros/constrangidos



Fonte: Produzido pelos autores

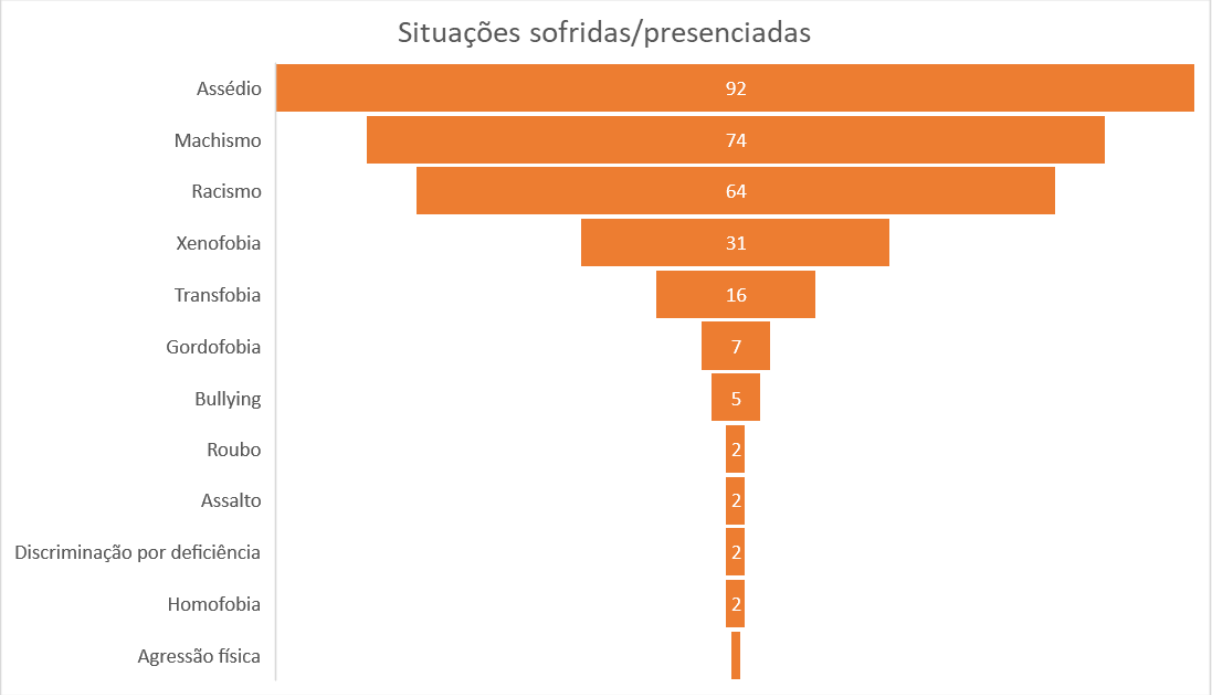
Nesta pergunta, dos 162 entrevistados, 137 apontaram pelo menos um local. Com base no gráfico, percebe-se que “Na rua”, com 95 respostas, seguido de

“Algum estabelecimento”, com 83, são os lugares que mais ocorrem este tipo de situação. Logo na sequência, com 36 respostas, “Em casa”, também apresenta um dado alarmante. Como “Outros”, foram citados diversos locais como “Trabalho”, com 10 respostas, “Escola/Faculdade” com 16, “Transporte Público” com 2 e “Academia” com somente uma resposta.

Outra pergunta contida no formulário é com relação a situações sofridas ou presenciadas pelos entrevistados. Além das opções de Assédio, Racismo, Machismo, Xenofobia e Transfobia, também era possível incluir demais ocorrências.

No gráfico 4 é apresentado uma visão geral das situações apontadas pelos entrevistados:

Gráfico 4 - Situações sofridas ou presenciadas pelos entrevistados



Fonte: Produzido pelos autores

Observa-se que o Assédio e o Machismo são as condutas mais incidentes, com 92 e 74 respostas, respectivamente. O Racismo também é bem evidente, com 64 respostas. Xenofobia e Transfobia também estão presentes nas respostas, mas com menor incidência, com 31 e 16 respostas.

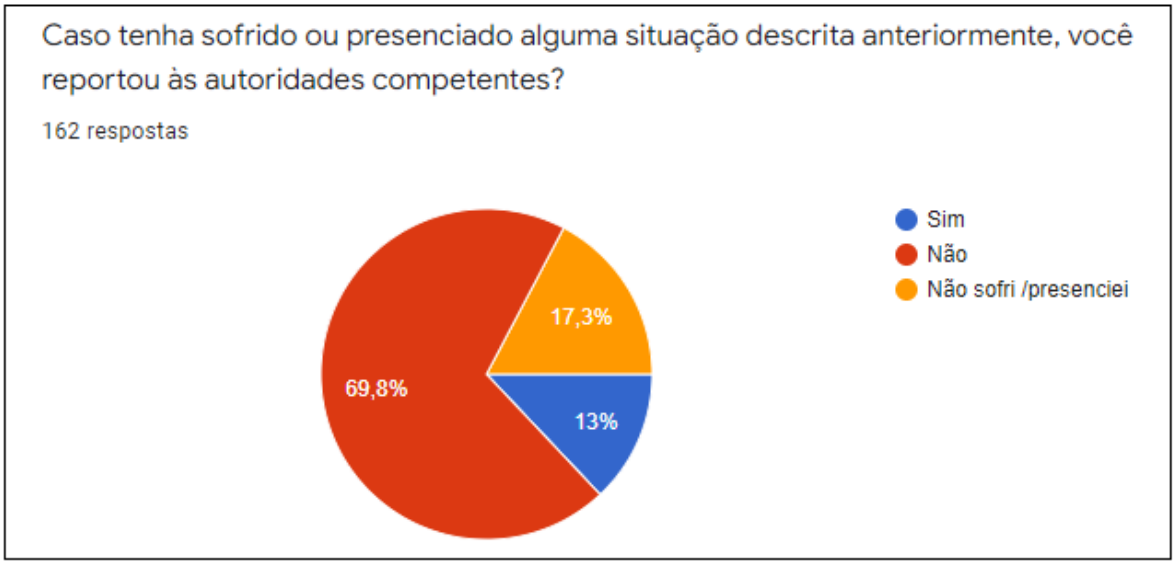
Os entrevistados também compartilharam outras situações sofridas ou presenciadas como Gordofobia, com 7 respostas, *Bullying* com 5 menções, Roubo,

Assalto, Discriminação por deficiência e Homofobia com 2 respostas cada e também Agressão física com apenas 1 resposta.

É importante ressaltar que nesta pergunta, 100% dos participantes responderam com pelo menos uma das alternativas.

Dos 162 entrevistados que relataram sofrer ou presenciar situações envolvendo as questões citadas acima, além de muitas outras carregadas de intolerância, questionou-se quanto ao relato dos acontecidos às autoridades competentes e 69,8% disseram não ter reportado, enquanto que 17,3% informaram não ter sofrido ou presenciado nenhum caso e 13% notificaram às autoridades:

Gráfico 5 - Reporte de ocorrências às autoridades competentes



Fonte: Gerado automaticamente pelo Google Forms

Ainda neste tópico, questionou-se quanto aos motivos que levaram os entrevistados a não reportarem as ocorrências às autoridades. Nesta questão, 104 participantes justificaram as suas respostas. Para melhor visualização dos relatos, foi gerada uma nuvem de palavras que aponta os motivos narrados, além de evidenciar aqueles mais recorrentes:

Figura 11 - Nuvem de Palavras: Motivos que levaram ao silêncio das vítimas

Fonte: Produzido pelos autores

É possível perceber que o principal motivo pelo qual as pessoas não reportam tais ocorridos é o medo, por várias razões; seguido pela insegurança e a vergonha. Algumas respostas obtidas neste questionamento foram: “Medo em alguns casos, falta de consciência com a situação em outros”; “Medo de perseguição”; “Não saber como ou a quem reportar”; “Não acho canal adequado”; “Pela idade e inexperiência não saber a quem reportar e se deveria ou não reportar”; entre outras.

Para finalizar o levantamento, foi solicitado aos entrevistados que descrevessem, em poucas palavras, algo que os daria mais segurança em relação às situações abordadas. Com as respostas obtidas, foi gerada uma nova nuvem de palavras contendo os termos mais comentados.

Figura 12 - Nuvem de Palavras: Soluções para maior segurança

Fonte: Produzido pelos autores

A partir da figura 12, pode-se perceber que os entrevistados buscam por lugares mais seguros, há também anseio para que haja leis mais eficazes e maior atenção a estes cenários por parte das autoridades competentes; além de comportamentos que deveriam ser básicos para todo ser humano, como respeito e educação. Algumas respostas nesta seção foram: “Que houvesse câmeras nos locais, que tenham mais pessoas perto e que já tivesse um tipo de histórico no local desse tipo de acontecimento”; “Algum tipo de aplicativo que permitisse realizar a denúncia da pessoa ou do local, que reportasse a situação para as autoridades de forma fácil e rápida, sem necessariamente precisar ir a delegacia fazer um boletim de ocorrência no primeiro momento”; “Ter uma rede de apoio e segurança”; “Ter acesso a uma rede de denúncia mais eficiente e menos invasiva”; “Alguma forma de poder reportar o ocorrido sem ser exposto; Evitar lugares onde isso ocorra frequentemente”; entre outras.

4.3 Análise dos Dados

Com base nos resultados obtidos, percebe-se que casos de discriminação ou assédio podem ser extremamente comuns na sociedade e que, na maioria das

ocasiões, a vítima prefere não reportar o ocorrido por medo ou pela incerteza de que terá algum tipo de justiça ou conforto. Ademais, fica evidente que uma forma simples de divulgação e informação retroativa sobre a incidência destes casos seria bem-recebida e altamente útil para a sociedade; o que condiz com a proposta apresentada neste projeto.

5 Ferramentas e Desenvolvimento

Nesta seção, serão abordadas as ferramentas utilizadas e todo o processo de desenvolvimento do aplicativo.

5.1 Ferramentas

Nesta subseção serão apresentadas as ferramentas necessárias para o desenvolvimento do aplicativo, bem como a importância de cada e o motivo que levou à sua escolha.

5.1.1 Banco de Dados

Para a estruturação e desenvolvimento do Banco de Dados, foi utilizada a linguagem SQL e o Banco de Dados MySQL.

O SQL (*Standard Query Language*, ou Linguagem de Consulta Estruturada) é uma linguagem declarativa padrão para bancos de dados relacionais (baseados em tabelas).

A linguagem surgiu em meados da década de 70, sendo o resultado dos estudos do então pesquisador da IBM Edgar Frank Codd, que visava desenvolver uma linguagem de banco de dados para o modelo relacional. Segundo Juliano (2008), o primeiro sistema de um banco de dados baseado em SQL foi comercializado no final dos anos 70 e em 1982 era lançada a primeira versão padronizada desta linguagem.

Ainda de acordo com Juliano (2008), o SQL possui papel fundamental atualmente nos Sistemas de Gerenciamento de Banco de Dados (SGBD), podendo ter vários enfoques como: Linguagem de Manipulação de Dados (DML); Linguagem de Definição de Dados (DDL); Linguagem de Controle de Dados (DCL) ou Linguagem de Consultas de Dados (DQL).

A linguagem SQL é utilizada de forma relativamente semelhante nos principais bancos de dados relacionais do mercado: Oracle, MySQL, MariaDB,

PostgreSQL, Microsoft SQL Server e outros. Atualmente os SGBDs MySQL e PostgreSQL figuram entre os mais populares, pois possuem versões gratuitas e de código aberto.

O MySQL, assim como outros bancos de dados relacionais, segundo Pisa (2012), armazena os dados divididos em várias áreas, que são as tabelas, sem agrupar tudo em uma grande unidade de armazenamento.

Em 1995, a empresa sueca chamada MySQL AB criou o MySQL e durante os seus primeiros anos, o foco de desenvolvimento era velocidade e produtividade, em vez de conjunto de recursos. No entanto, novos recursos foram implantados a cada lançamento. Com o passar dos anos, de acordo com uma publicação no Atlantic.Net, devido às suas significativas vantagens de desempenho e facilidade de uso, o MySQL tornou-se um banco de dados muito visado. A partir de 2001, novos recursos foram adicionados, o que atraiu o mercado corporativo e contribuiu novamente para o seu crescimento.

De acordo com Boyett (2022), a *Sun Microsystems* comprou a MySQL AB e mais tarde, em 2010, a *Oracle* adquiriu a empresa. Atualmente, as mais recentes versões do MySQL são muito ricas em recursos e mantêm uma forte capacidade de desempenho. Algumas das vantagens do MySQL comparado aos outros bancos do mesmo porte são a facilidade para programar, funções mais simples, permissão para modificação, entre outros.

O MySQL *Workbench* é uma ferramenta de gerenciamento gráfico, desenvolvida pela Oracle, que interage com o banco de dados do servidor MySQL.

Segundo Góis (2021), das funcionalidades que o MySQL *Workbench* fornece, algumas delas são: Desenvolvimento SQL, que permite executar consultas SQL, criar e gerenciar conexões com os servidores de banco de dados; Modelagem de dados, em que é possível criar modelos do banco de dados Schema graficamente e editar tabelas, colunas, índices, gatilhos, etc; Administração do Servidor, para administrar usuários, inspecionar dados de auditoria e realizar *backups*; Migração de dados, que permite migrar do Microsoft SQL *Server*, SQLite, Microsoft *Access*, PostgreSQL, Sybase ASE, SQL *Anywhere* e outras tabelas, objetos e dados para o MySQL.

O MySQL *Workbench* possui licença GNU (*General Public License*) e é uma ferramenta gratuita, multiplataforma, que faz conexão direta com o banco de dados, além de ser excelente para documentação.

A escolha destas tecnologias se deve pelo fato de que a linguagem SQL é a mais popular para operações envolvendo Banco de Dados e o Sistema de Gerenciamento também é um dos mais populares, além de fornecer ótimas soluções de forma gratuita.

5.1.2 Backend

Para o desenvolvimento do *backend* da aplicação, foi utilizada a linguagem Java com o *framework Spring Boot*.

Criada em 1992 pela empresa Sun *Microsystems*, em um projeto liderado por James Gosling, a linguagem Java foi idealizada inicialmente para o desenvolvimento de pequenos aplicativos e programas de controle para aparelhos domésticos. Pouco tempo após o início do projeto, a empresa percebeu que poderia utilizar a ideia inicial para possibilitar a execução de pequenas aplicações em *browsers*, ampliando sua cobertura para a internet.

De acordo com Deitel e Deitel (2005), “através desse pensamento que o Java 1.0 foi lançado: focado em transformar o *browser* de apenas um cliente burro para uma aplicação que possa também realizar operações, não apenas renderizar html”.

Atualmente Java não é somente uma linguagem de programação, mas uma plataforma completa de desenvolvimento e execução, composta pelos pilares: JVM (Máquina Virtual Java); Conjunto completo de APIs e a linguagem em si. Além de ser uma tecnologia independente de *hardware* ou sistemas operacionais. Atualmente se faz presente nos sistemas operacionais mais utilizados do mercado: Windows, Linux, Unix, Mac e Solaris; além de ser utilizado no desenvolvimento de aplicações *web*, *desktop*, *mobile* e sistemas embarcados.

Para Faria (2009), “Java é simples, de fácil aprendizagem; Orientada a objetos; Familiar para os programadores de C/C++; Robusta; Segura pois é executada em um ambiente próprio (JRE), o que inviabiliza a intrusão de código malicioso; portável, pois pode ser executada em qualquer máquina que possua JRE”.

Para Afonso (2017, pg. 12) “O *Spring* não é um *framework* apenas, mas um conjunto de projetos que resolvem várias situações do cotidiano de um programador, ajudando a criar aplicações Java com simplicidade e flexibilidade”. Desta forma, pode-se definir o *Spring* como um ecossistema de soluções para projetos desenvolvidos com a linguagem Java.

O ecossistema *Spring* possui soluções direcionadas para acesso a banco de dados (*Spring Data*), prover segurança (*Spring Security*), além de um componente base para as aplicações (*Spring Framework*).

O *Spring Framework* foi desenvolvido para simplificar o processo de configuração e o desenvolvimento, para que o foco da aplicação seja maior nas regras de negócio do que na infraestrutura. Dentre as principais funcionalidades desta ferramenta, é possível destacar o *Spring* MVC, suporte para JDBC e JPA e a Injeção de Dependências.; sendo que o último pode ser considerado a base do *Spring Framework*.

Ainda segundo Afonso (2017, pg. 14) “Injeção de dependências (ou *Dependency Injection* – DI) é um tipo de inversão de controle (ou *Inversion of Control* – IoC) que dá nome ao processo de prover instâncias de classes que um objeto precisa para funcionar.”

Para a injeção de dependências é utilizada a anotação *@Autowired*, responsável por comunicar ao *Spring* quando uma instância de alguma classe deve ser injetada. Desta forma, é possível manter um baixo acoplamento entre classes dentro de um projeto.

Para este projeto, foram escolhidos a linguagem Java e o *framework Spring Boot* pela sua robustez, segurança e facilidade que suas ferramentas fornecem ao desenvolvedor.

5.1.3 Mobile

Para o desenvolvimento da aplicação *mobile*, foi escolhido o *framework Flutter*, que utiliza a linguagem Dart.

Dart é uma linguagem de programação multiparadigma e multiplataforma criada pela *Google* em 2011. Apesar de ser uma linguagem recente e que ainda está se consolidando no mercado do desenvolvimento, por suas características, é vista como uma excelente opção para desenvolvedores que buscam uma opção multi plataforma robusta e eficiente.

As principais vantagens do Dart que merecem atenção por sua importância no desenvolvimento são a facilidade de aprendizado, já que sua estruturação e sintaxe é similar ao utilizado no JS; a ampla documentação, o que permite solucionar qualquer questão que surja durante o desenvolvimento; a alta performance, já que programas desenvolvidos na linguagem tendem a ser mais

rápidos e a estabilidade e eficiência, podendo ser utilizada para produzir aplicações ágeis e de escala.

Com isso, grandes empresas já adotaram a aplicação do Dart em suas soluções, principalmente através do *Flutter*. O que justifica o fato de ser comum essas duas tecnologias sendo utilizadas em conjunto é que a linguagem de programação padrão utilizada pelo *Flutter* é o Dart.

O *Flutter* utiliza uma abordagem única para lidar com os componentes nativos de cada plataforma, em que cada um deles é implementado pelo próprio *framework* e apresentado ao usuário por um motor de renderização próprio. Atualmente é possível utilizar o *Flutter* para o desenvolvimento não apenas de aplicativo de celular, mas também de aplicações para *web* e para sistemas operacionais.

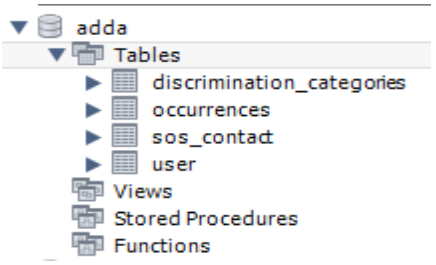
A sua construção contempla a criação de aplicações de forma rápida, pois conta com o recurso *Stateful Hot Reload*, que realiza a atualização de forma automática do aplicativo no momento em que é salvo no arquivo do projeto e também ajuda a adicionar recursos e corrigir *bugs* rapidamente. Além disso, é possível utilizar uma série de *widgets* customizáveis, já desenvolvidos de forma reativa, para construir a interface do usuário. Há um grande catálogo com uma coleção de *widgets* de animações, *inputs*, *scrolling*, *styling*, entre outros. Outro ponto é com relação a criação de interfaces flexíveis, em que é possível fazer o controle de cada *pixel* na tela, já que ele traz os *widgets* renderizados, animações e gestos para *frameworks*.

Tendo isso em vista, a escolha dessas tecnologias para o desenvolvimento desse projeto se dá pela versatilidade, baixa curva de aprendizado e praticidade na implementação, através da enorme quantidade de *widgets*.

5.2 Métodos ou Desenvolvimento

Após a modelagem lógica das entidades e relacionamentos (Figura 9), realizada através da ferramenta MySQL *Workbench*, o banco de dados e suas respectivas tabelas foram exportados a partir deste modelo. Vale ressaltar que tal exportação também é um recurso da ferramenta mencionada. A estrutura do banco de dados pode ser conferida a seguir.

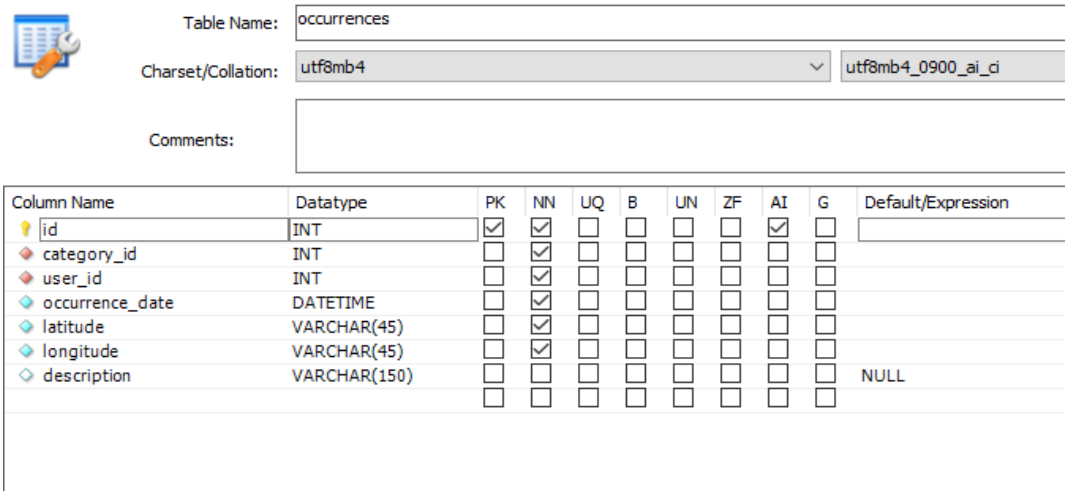
Figura 13 - Estrutura do Banco de Dados no MySQL Workbench



Fonte: Produzido pelos autores

As tabelas foram exportadas corretamente, bem como os relacionamentos criados na modelagem. Na Figura 14 é possível observar a estrutura da tabela "*occurrences*", que armazena as informações referentes às ocorrências.

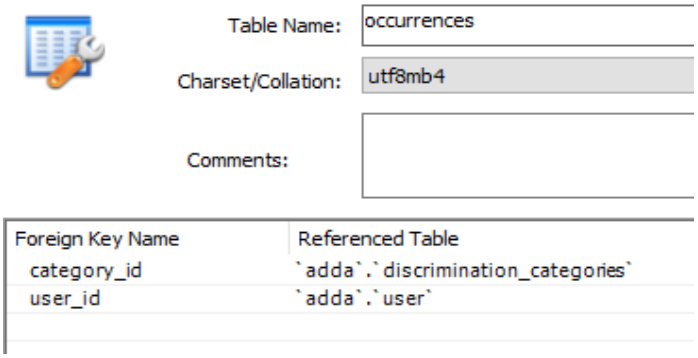
Figura 14 - Estrutura da tabela "occurrences" no MySQL Workbench



Fonte: Produzido pelos autores

Já na Figura 15, tem-se as colunas definidas como chaves estrangeiras ("*category_id*" e "*user_id*") desta mesma tabela e suas respectivas referências nas tabelas originais.

Figura 15 - Chaves estrangeiras da tabela "occurrences" no MySQL Workbench



Fonte: Produzido pelos autores

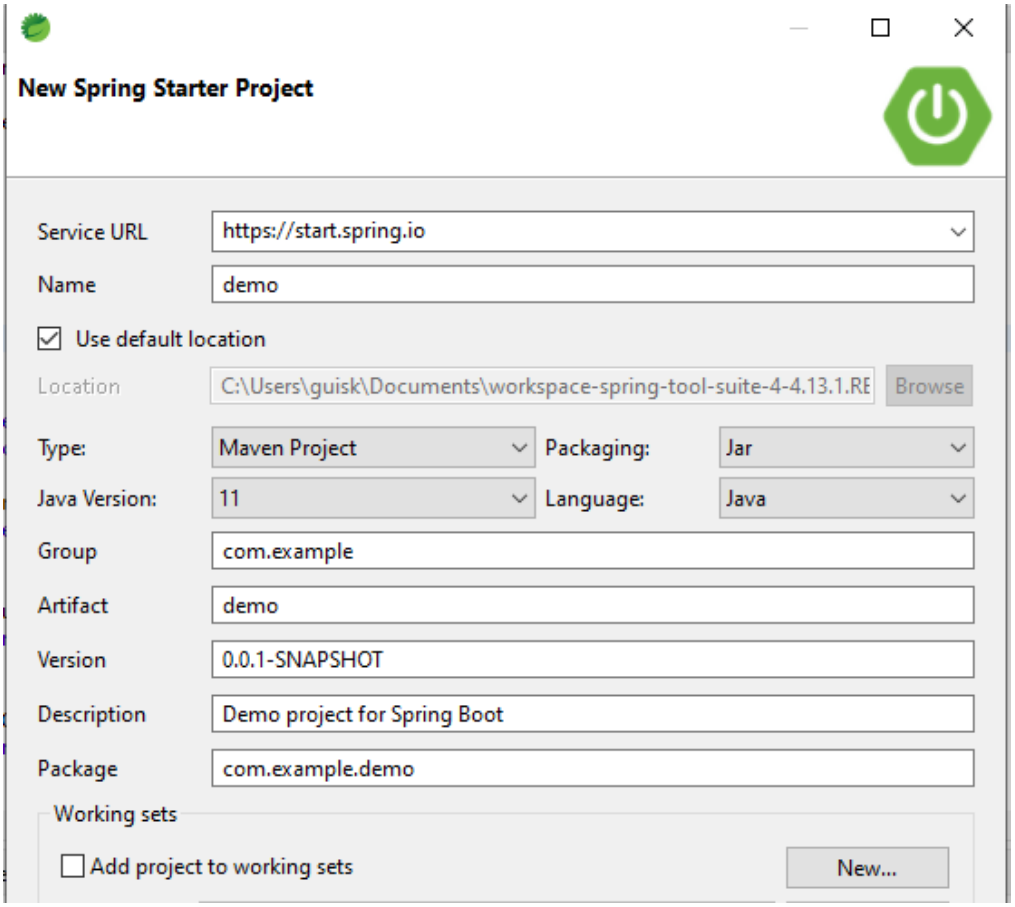
Com a base de dados estruturada, o próximo passo foi a criação do projeto Java, em sua versão 11, para o *backend* da aplicação.

O *backend* é responsável pela aplicação das regras de negócio, segurança, autenticação e comunicação com o banco de dados. Em outras palavras, atua como uma API (*Application Programming Interface*) RESTful, sendo responsável por manter a integridade da aplicação e expor as informações necessárias ao aplicativo *mobile*.

O projeto foi criado através da ferramenta *Spring Initializr*, presente na IDE (*Integrated Development Environment*) *Spring Tool Suite*. Esta ferramenta permite a criação de uma aplicação Java com *Spring Boot* de maneira fácil e intuitiva.

No momento da criação (Figura 16) é possível selecionar a versão do Java, o gerenciador de dependências e as dependências iniciais que serão utilizadas no projeto, além de outras configurações básicas.

Figura 16 - Criação de um novo projeto Java no Spring Tool Suite



Fonte: Produzido pelos autores

Neste caso, foram escolhidos a versão 11 do Java e o *Maven* como gerenciador de dependências.

Após a criação do projeto é gerado um arquivo no formato xml chamado pom (Figura 17), que é o responsável pelas configurações que serão interpretadas pelo *Maven*. Neste arquivo constam informações como nome do projeto, versões de cada ferramenta, *plugins* externos e todas as dependências utilizadas.

Figura 17 - Arquivo pom.xml

adda/pom.xml

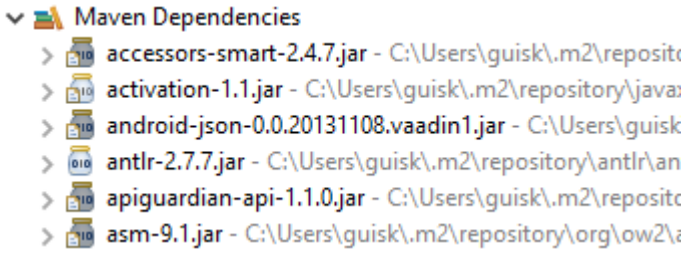
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.4</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>
    <groupId>br.com</groupId>
    <artifactId>adda</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>adda</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
```

Fonte: Produzido pelos autores

Ao executar a aplicação, o *Maven* utiliza estas informações para baixar os arquivos no formato jar de todas as dependências e adicioná-las ao projeto (Figura 18), para que sejam utilizadas. Algumas dependências utilizadas no projeto foram:

- *Spring Boot Starter JPA* - API padrão que descreve uma interface comum para persistência de dados em Java;
- *MySQL Connector Java* - permite a comunicação de uma aplicação Java com um banco de dados MySQL;
- *Spring Boot Starter Security* - sistema de autenticação e autorização para aplicações Java;
- *Javax Mail* - pacote para envio de e-mails em aplicações Java.

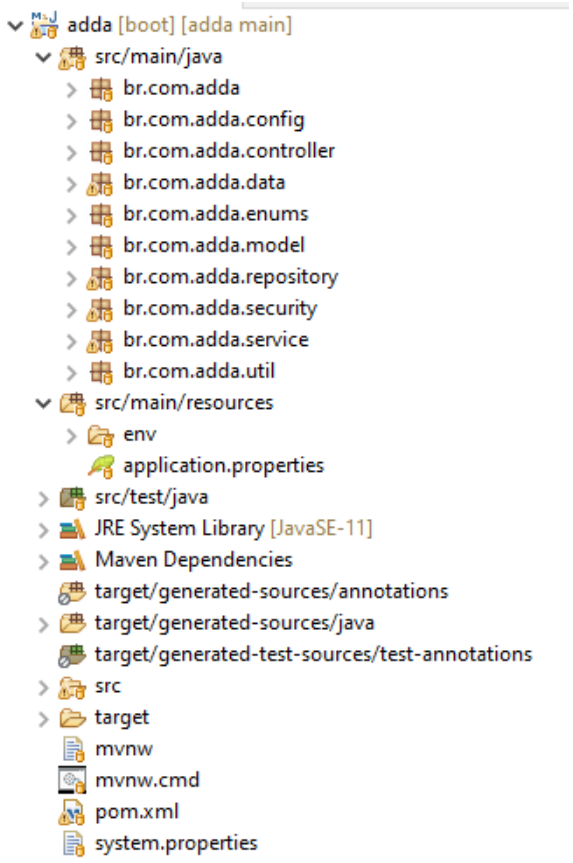
Figura 18 - Dependências do projeto no formato jar



Fonte: Produzido pelos autores

Após sua criação, foram geradas as classes iniciais da aplicação, seguindo o padrão MVC *(Model-View-Controller*). A estrutura do projeto pode ser conferida na Figura 19.

Figura 19 - Estrutura do projeto Java



Fonte: Produzido pelos autores

Os principais diretórios do projeto são: java e *resources*, ambos localizados no caminho *src/main*. O diretório java engloba toda a estrutura principal e os pacotes referentes à arquitetura utilizada; como a *controller*, *model*, *service* e *repository*. Já o diretório *resources* contém os arquivos de configuração da aplicação, como os dados para acesso ao servidor de *e-mail* e credenciais para conexão com o banco de dados.

Em cada pacote do diretório *main*, as classes são separadas de acordo com suas responsabilidades. Neste caso, como o projeto aborda três segmentos diferentes (usuário, contato de emergência e ocorrências), a maioria dos pacotes possuem uma classe para cada um destes; exceto em alguns pacotes como *model* e *security*, onde existem classes adicionais para tratamento de envio de *e-mail* ou autenticação.

O primeiro passo foi construir o sistema de autenticação, que opera da seguinte maneira: Usuário insere o seu *e-mail* e senha; o sistema verifica se existe uma combinação semelhante na tabela de usuários do banco de dados e, em caso positivo, é gerado um *token* jwt (*Json Web Token*) que será trafegado no *header* de todas as requisições realizadas pelo usuário enquanto logado, em caso negativo, é exibida uma mensagem de erro no momento do *login*.

Para isto, foi adicionado o pacote *security*, no qual tem-se os filtros de Validação e Autenticação, além de uma classe de configuração onde é possível definir as características do *token* jwt que será gerado (Figura 20), além de definir rotas que não necessitam de validação; como por exemplo a própria rota de *login* ou a rota de cadastro de um novo usuário (Figura 21).

Figura 20 - Configurações do token

```
@Override
protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain,
        Authentication authResult) throws IOException, ServletException {

    UserDetailData userData = (UserDetailData) authResult.getPrincipal();

    String token = JWT.create().withSubject(userData.getUsername())
            .sign(Algorithm.HMAC512(TOKEN_SENHA));
```

Fonte: Produzido pelos autores

Figura 21 - Habilitações de rotas e adição de filtros

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable().authorizeRequests().antMatchers(HttpMethod.POST, "/login").permitAll()
            .antMatchers(HttpMethod.POST, "/user").permitAll().antMatchers(HttpMethod.PUT, "/user/password/reset")
            .permitAll().anyRequest().authenticated().and()
            .addFilter(new JWTAuthenticationFilter(authenticationManager()))
            .addFilter(new JWTValidationFilter(authenticationManager())).sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
}
```

Fonte: Produzido pelos autores

Após a configuração de autenticação, foram desenvolvidas as operações destinadas aos usuários, como: cadastro, listagem, busca por id, alteração e exclusão de um usuário do sistema. Os três segmentos principais da aplicação (usuário, contato de emergência e ocorrências) seguem o mesmo fluxo: *Controller* - *Service* - *Repository*. A classe de tipo *controller* atua como a porta de entrada da aplicação, é onde são definidas as rotas e o método para cada funcionalidade. Neste projeto, a comunicação é realizada através de requisições HTTP (*GET*, *POST*, *PUT*, *PATCH* e *DELETE*). Além das funcionalidades principais, o segmento de usuários ainda possui um método para a redefinição de senha, onde o usuário informa o *e-mail* cadastrado, uma nova senha é gerada automaticamente e enviada para tal *e-mail*. A Figura 22 apresenta um trecho da *controller* de usuários.

Figura 22 - Controller de operações relacionadas ao usuário

```
@RestController
@Api(value = "Usuário")
public class UserController {

    @Autowired
    private UserService userService;

    @ApiOperation(value = "Lista todos usuários")
    @GetMapping("/user")
    public ResponseEntity<?> get() {
        return userService.listUsers();
    }

    @ApiOperation(value = "Cria um novo usuário")
    @PostMapping("/user")
    public ResponseEntity<?> add(@RequestBody User user) {
        return userService.addUser(user);
    }

    @ApiOperation(value = "Remove um usuário")
    @DeleteMapping("/user/{id}")
    public ResponseEntity<?> remove(@PathVariable Long id) {
        return userService.removeUser(id);
    }

    @ApiOperation(value = "Atualiza um usuário já cadastrado")
    @PutMapping("/user")
    public ResponseEntity<?> update(@RequestBody User user) {
        return userService.updateUser(user);
    }

    @ApiOperation(value = "Redefine a senha")
    @PutMapping("/user/password/reset")
    public ResponseEntity<?> resetPassword(@RequestParam String email) {
        return userService.resetPassword(email);
    }
```

Fonte: Produzido pelos autores

Após receber a requisição, a *controller* aciona o método correspondente na classe *service*. Vale ressaltar que a *service* é referenciada na *controller* através da Injeção de Dependência, mencionada no Item 5.1.2 deste documento.

As classes *service* são responsáveis pela lógica de negócio da aplicação, tratamento de dados e comunicação com camadas mais internas, como uma classe de dados, por exemplo.

Na Figura 23, é possível analisar o método na classe *UserService* responsável pela redefinição de senha. Este método verifica, através do *e-mail* informado, se existe um usuário correspondente no banco de dados e, em caso positivo, gera uma nova senha aleatoriamente, salva essa senha codificada no banco de dados e a envia para o *e-mail* do usuário.

Figura 23 - Método para redefinição de senha na classe UserService

```
public ResponseEntity<?> resetPassword(String email) {
    try {

        Optional<User> user = repository.findByEmail(email);

        if (user.isEmpty()) {
            return new ResponseEntity<>("Usuário com e-mail " + email + " não cadastrado!", HttpStatus.BAD_REQUEST);
        }

        String random = generateRandomString();

        String newPass = encoder.encode(random);

        user.get().setPassword(newPass);

        repository.save(user.get());

        sendEmail(email, random);

        return new ResponseEntity<>("Senha temporária gerada! Verifique seu e-mail.", HttpStatus.OK);

    } catch (Exception e) {
        return new ResponseEntity<>("Usuário com e-mail " + email + " não cadastrado!", HttpStatus.BAD_REQUEST);
    }
}
```

Fonte: Produzido pelos autores

Nota-se que para as operações no banco de dados, é chamada uma outra classe, denominada como *repository*. Tal camada é a responsável pela comunicação direta com o banco de dados.

Neste caso, como foi utilizado a dependência JPA no projeto, a interface *repository* estende da interface *JpaRepository*, que proporciona vários métodos prontos para utilização, como *findAll* (retorna todos os dados da tabela referenciada) e *save* (salva um novo dado na tabela referenciada). Além dos métodos padrões, também é possível inserir instruções personalizadas conforme a necessidade do

projeto. A figura 24 apresenta a interface *UserRepository* com os métodos personalizados: *findByEmailAndPhone* e *findByEmail*.

Figura 24 - Interface UserRepository

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {

    @Query("SELECT new User(u.id, u.name, u.email, u.password, u.gender, u.age, u.phone, u.photoUrl,"
            + " u.isAdmin) FROM User u WHERE u.email = :email OR u.phone = :phone")
    User findByEmailAndPhone(@Param("email") String email, @Param("phone") String phone);

    public Optional<User> findByEmail(String email);
}
```

Fonte: Produzido pelos autores

Ainda nas camadas do fluxo principal, é importante mencionar a camada *model*. Nas classes do tipo *model*, estão estruturadas as entidades do projeto e que geralmente fazem referência às tabelas do banco de dados.

Tais classes também são instanciadas como objetos, sendo populados a partir de dados enviados pelo usuário ou por consultas realizadas no banco de dados. Na Figura 25 é possível observar a estrutura inicial da classe *Occurrences*.

Figura 25 - Estrutura inicial da classe Occurrences

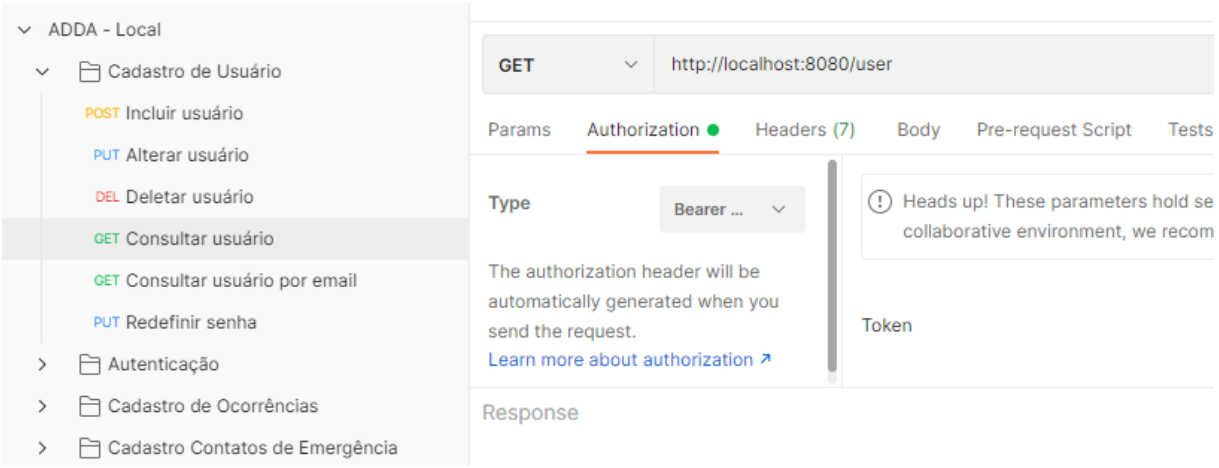
```
@Entity
@Table(name="occurrences")
public class Occurrences implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = -2070535942817794052L;


    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;
    private int categoryId;
    private Long userId;

    @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss", shape = JsonFormat.Shape.STRING)
    private LocalDateTime occurrenceDate;

    private String latitude;
    private String longitude;
    private String description;

    @Transient
    private String categoryName;
```

Fonte: Produzido pelos autores

Como já mencionado anteriormente, os três segmentos principais da aplicação seguem os mesmos fluxos e métodos de desenvolvimento.

Todos os métodos desenvolvidos no projeto do *backend* foram validados utilizando a ferramenta *Postman*, conforme Figura 26.

Figura 26 - Interface do Postman com as requisições implementadas na API

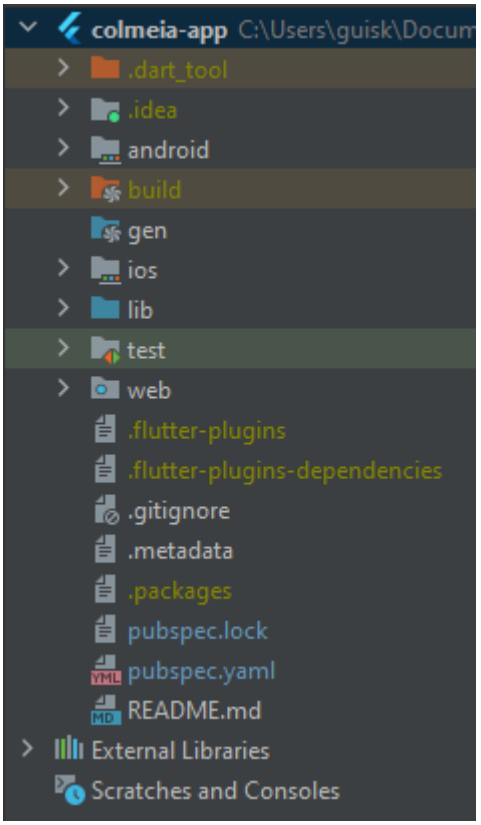


Fonte: Produzido pelos autores

Após a validação da API, deu-se início ao desenvolvimento do aplicativo *mobile*, este que foi desenvolvido com o *framework Flutter* em sua versão 2.2.3. Como o *Flutter* é voltado ao desenvolvimento de aplicativos híbridos, na estrutura de sua aplicação (Figura 27) há um diretório específico para configurações do iOS, outro para configurações do Android e um terceiro, chamado lib, para o desenvolvimento do código em si, que será comum para ambos os sistemas operacionais. Além dos diretórios, é gerado um arquivo no formato yaml chamado pubspec (Figura 28); neste arquivo, de modo semelhante ao pom.xml do Java, são inseridas as configurações e dependências utilizadas no projeto. As principais dependências utilizadas foram:

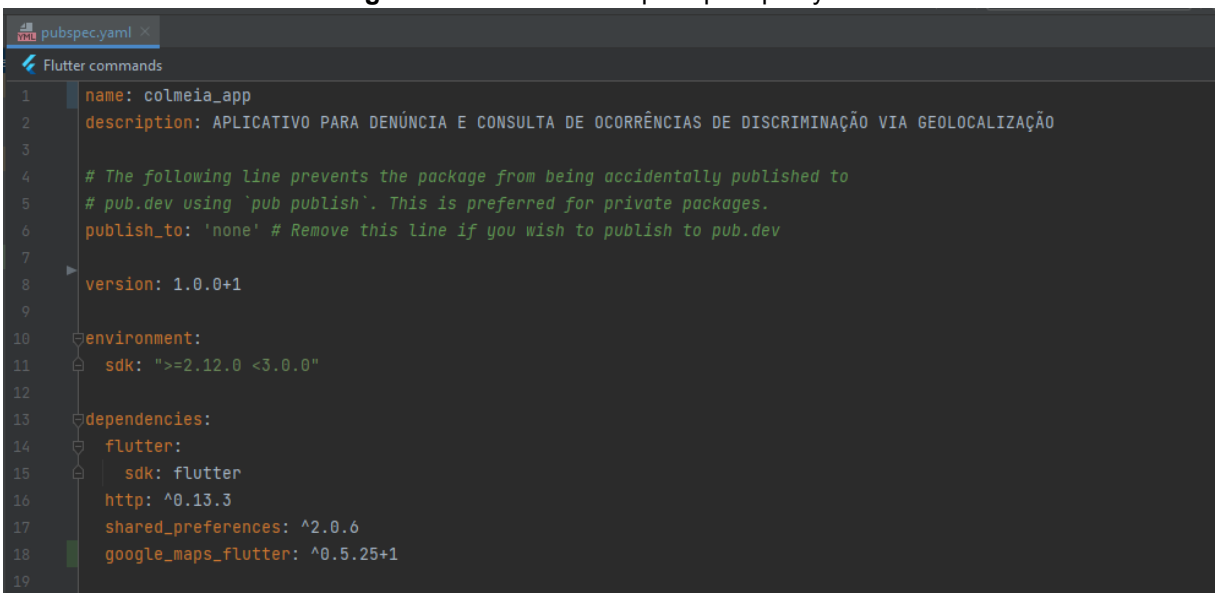
- *http* - permite a efetuação de chamadas http para serviços externos (neste caso, requisições à API desenvolvida em Java);
- *shared_preferences* - utilizada para o armazenamento tipo chave-valor de informações localmente, como *token* e o id do usuário logado;
- *google_maps_flutter* - necessária para a utilização da API de mapas disponibilizada pela Google.

Figura 27 - Estrutura do projeto em Flutter



Fonte: Produzido pelos autores

Figura 28 - Trecho do arquivo pubspec.yaml

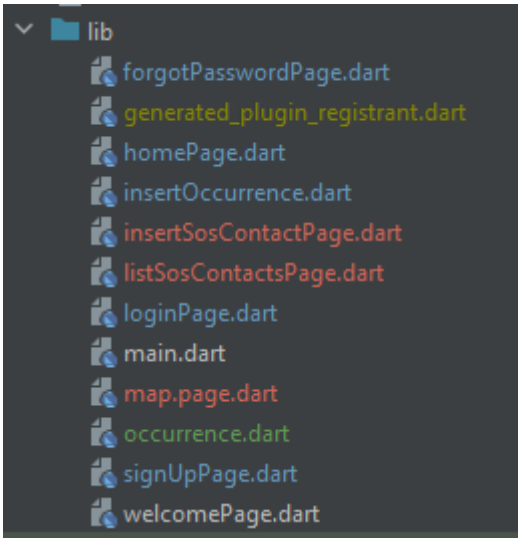


Fonte: Produzido pelos autores

No diretório lib, cada arquivo se refere a uma página ou classe de entidades do aplicativo, conforme apresentado na Figura 29. Cada classe, por sua vez, possui

toda a estrutura, lógica e métodos de requisição necessários para o seu funcionamento.

Figura 29 - Estrutura do diretório lib



Fonte: Produzido pelos autores

O arquivo *main*.dart (Figura 30) contém a configuração inicial que será executada no carregamento do aplicativo. Neste caso, há um *widget* chamado "*MaterialApp*" que é responsável pelo redirecionamento à classe principal "*WelcomePage*". Vale ressaltar que a construção e estruturação de telas no *Flutter* é realizada através de *widgets*.

Figura 30 - Classe principal da aplicação

```dart
main.dart
/.../

import ...

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'ColmeiaApp',
      theme: ThemeData(
        primarySwatch: Colors.orange,
      ),  // ThemeData
      home: WelcomePage(),
    );  // MaterialApp
  }
}
```

Fonte: Produzido pelos autores

Na classe *WelcomePage* é realizada uma verificação, através de uma instância de *SharedPreferences*, se existe um *token* salvo na sessão. Caso exista, significa que há um usuário logado na aplicação e, neste cenário, o usuário é redirecionado para a *HomePage*. Em caso negativo, o mesmo é redirecionado para a página de Login.

Após a estruturação da *WelcomePage*, foram desenvolvidas as páginas responsáveis pelo cadastro e autenticação do usuário: *LoginPage*, *SignUpPage* e *ForgotPasswordPage*. Na classe *LoginPage*, há um *widget* de Formulário que contém dois campos de entrada de texto (um para o *e-mail* e outro para a senha), além de três *widgets* de botão:

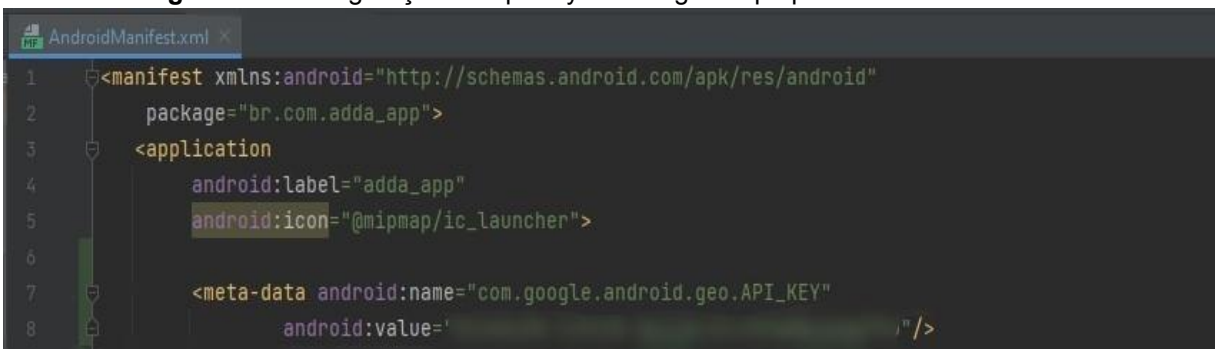
- Entrar - chama a função *login*, que efetua uma requisição http de tipo *POST* à api, especificamente ao endpoint /*login*, onde é verificada a combinação *e-mail*/senha na tabela de usuários no banco de dados e, em caso positivo, retorna o *token* gerado e o armazena em uma instância de *SharedPreferences*, além das informações do usuário. Após essa ação o usuário é redirecionado para a *HomePage*;
- Cadastrar - redireciona o usuário para a *SignUpPage*, que contém um formulário onde o mesmo pode inserir as informações requeridas e efetuar o cadastro no aplicativo, através da função *signUp*, que efetua uma requisição http de tipo *POST* ao endpoint /*user* da API;
- Esqueci minha senha - redireciona o usuário para a *ForgotPasswordPage*, onde há um *widget* de formulário com um campo de entrada de texto, para que o mesmo digite o seu e-mail cadastrado no aplicativo; e um botão “Redefinir Senha” que, ao ser clicado, efetua uma requisição http de tipo *PUT* ao endpoint *user/password/reset* e envia o *e-mail* digitado via *query param* para a API. Neste método a nova senha, gerada automaticamente pelo sistema, é encaminhada para o *e-mail* cadastrado.

A tela principal do aplicativo, *HomePage*, possui um *widget* para o mapa, além de um botão de menu, que abre um menu lateral para as outras opções do sistema: “Cadastrar ocorrência”, “Contatos de Emergência”, “Alterar Perfil” e “Sair”. Tais opções redirecionam o usuário para as suas telas correspondentes, todas possuem um formulário com campos de entrada onde é possível inserir as

informações requeridas e efetuar as ações designadas, através de requisições http à API. Vale ressaltar que para cada ação, foram considerados os cenários de sucesso e falha; sendo que, para o segundo, são exibidas notificações com o motivo que levou ao insucesso da ação. Para o *widget* do mapa, foi necessário criar uma conta na Plataforma de Nuvem da *Google* (*Google Cloud Platform*), para a geração de uma chave de API única que deve ser adicionada nas configurações do projeto.

Tal chave deve ser configurada nos diretórios dedicados aos sistemas Android e iOS. Na Figura 31 tem-se a configuração efetuada no arquivo *AndroidManifest*, do diretório android.

Figura 31 - Configuração da Api Key do Google Maps para o sistema Android



Fonte: Produzido pelos autores

Vale ressaltar que os serviços de nuvem da *Google* são pagos. Ao efetuar o cadastro, é disponibilizado um crédito de U$200,00 durante o primeiro mês de uso e, após esse período, o serviço é cobrado de acordo com o número de requisições. Atualmente, para a utilização do mapa estático, o valor cobrado é de U$2,00 por cada 1.000 requisições, segundo o site da própria plataforma.

Com o *Google Maps* configurado no projeto, foi utilizado o *widget GoogleMap* (Figura 32); no qual é possível definir a posição inicial da câmera, através dos parâmetros latitude e longitude, o *zoom* aplicado no mapa e os marcadores, que neste projeto representam as incidências de ocorrências.

Figura 32 - Widget GoogleMap

```
body: GoogleMap(
  onMapCreated: _onMapCreated,
    initialCameraPosition: CameraPosition(
      target: LatLng(lat, long),
      zoom: 15.0,
),  // CameraPosition
  markers: markers.map((e) => e).toSet(),
),  // GoogleMap
```

Fonte: Produzido pelos autores

Para a inclusão dos marcadores no mapa, é realizada uma requisição http de tipo *GET* ao endpoint */occurrences* da API, que retorna uma lista com as ocorrências cadastradas, de acordo com os parâmetros especificados na consulta. Com o retorno da requisição, é realizada uma iteração em cada item da lista e, para cada, é adicionado um *widget Marker* que contém a localização (latitude e longitude), além do tipo e data/hora de cada ocorrência. Cada *Marker* é adicionado a uma lista de marcadores, que então é inserida no mapa, conforme pode ser observado na Figura 33.

Figura 33 -Inclusão das ocorrências no mapa através dos marcadores

```
LatLng position = LatLng(lat, long);
    mapController!.moveCamera(CameraUpdate.newLatLng(position));

    List<Occurrence> occurrences = await getOccurrences(http.Client());

    List<Marker> markersList = [];

    for (int i = 0; i < occurrences.length; i++) {
      Marker marker = Marker(
        markerId: new MarkerId(occurrences[i].id.toString()),
        position: LatLng(double.parse(occurrences[i].latitude), double.parse(occurrences[i].longitude)),
        infoWindow: InfoWindow(
            title: convertCategoryName(occurrences[i].categoryId),
            snippet: occurrences[i].occurrenceDate
        ),  // InfoWindow
      );  // Marker

      markersList.add(marker);
    }

    setState(() {
      markers.addAll(markersList);
    });
```

Fonte: Produzido pelos autores

A cada nova inclusão de uma ocorrência, a classe responsável pelo mapa é novamente carregada e, consequentemente, atualiza automaticamente as informações no mesmo.

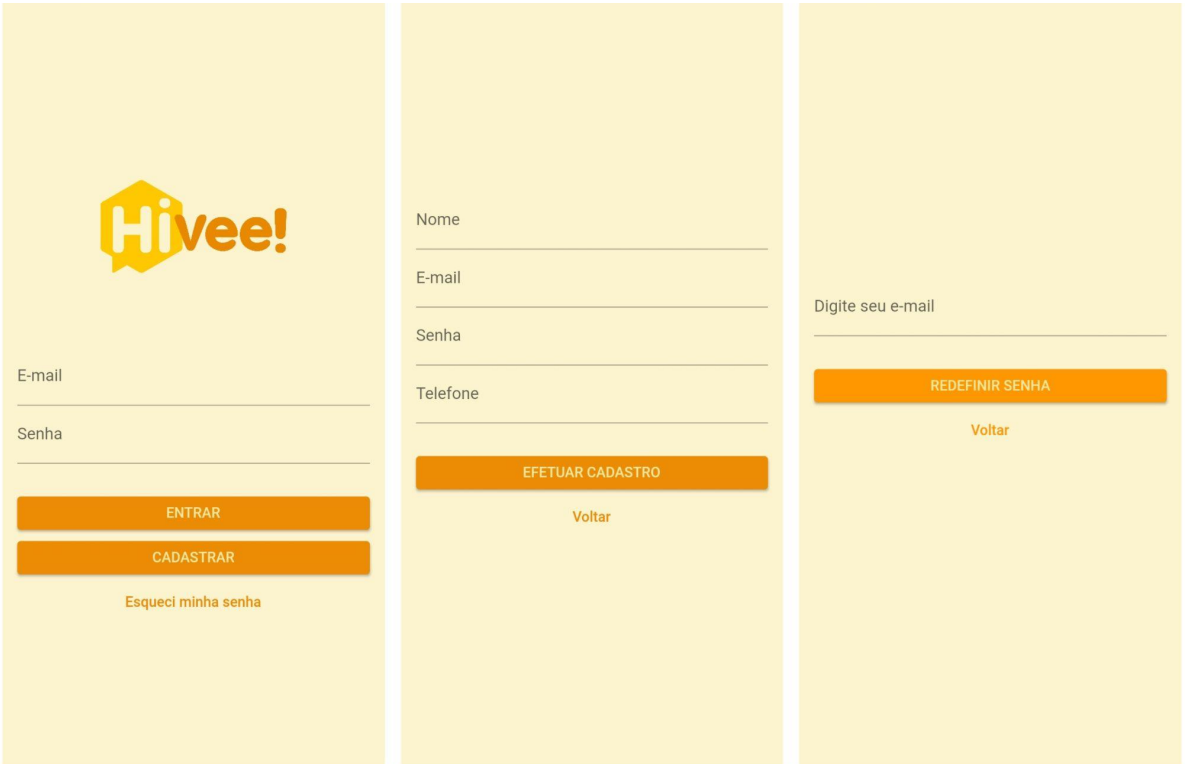
Após o desenvolvimento de todas as telas, o aplicativo foi testado localmente, considerando os cenários de sucesso e falha, com a utilização de um dispositivo físico.

6 Resultados e Discussão

O aplicativo, denominado *Hivee* (fazendo alusão à Colmeia), devido à proposta apresentada neste projeto, que consiste em criar uma rede compartilhada de informações em prol de uma comunidade, foi concluído com êxito e obteve resultado satisfatório nos testes iniciais, realizados pela equipe de desenvolvedores.

A Figura 34 apresenta *prints* do sistema em seu fluxo de cadastro e autenticação de usuários (*login*, cadastro de um novo usuário e redefinição de senha).

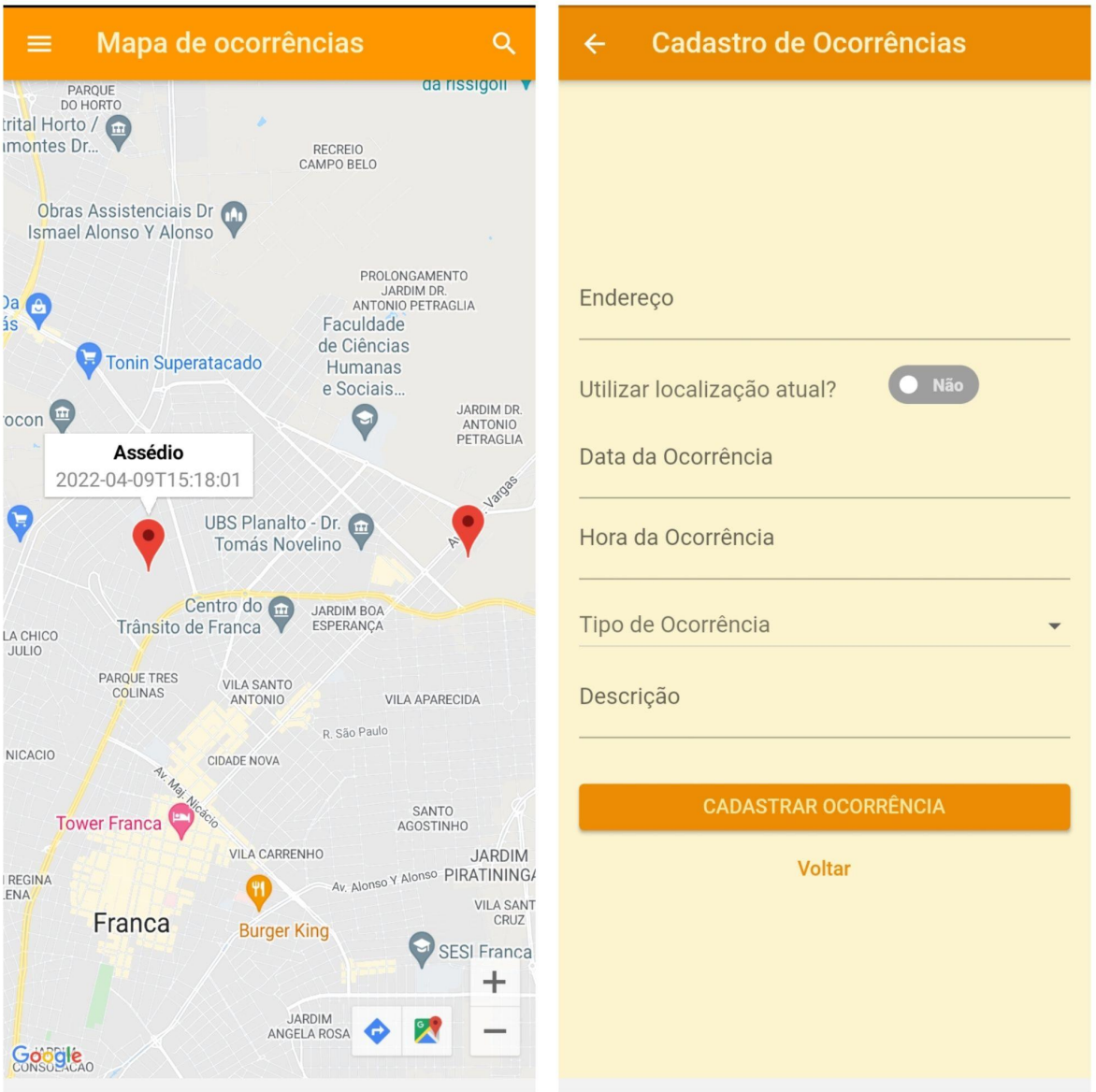
Figura 34 -Telas referentes ao fluxo de cadastro e autenticação de usuários



Fonte: Produzido pelos autores

Já na Figura 35, são apresentadas as telas referentes à consulta de ocorrências no mapa e o cadastro de uma nova ocorrência.

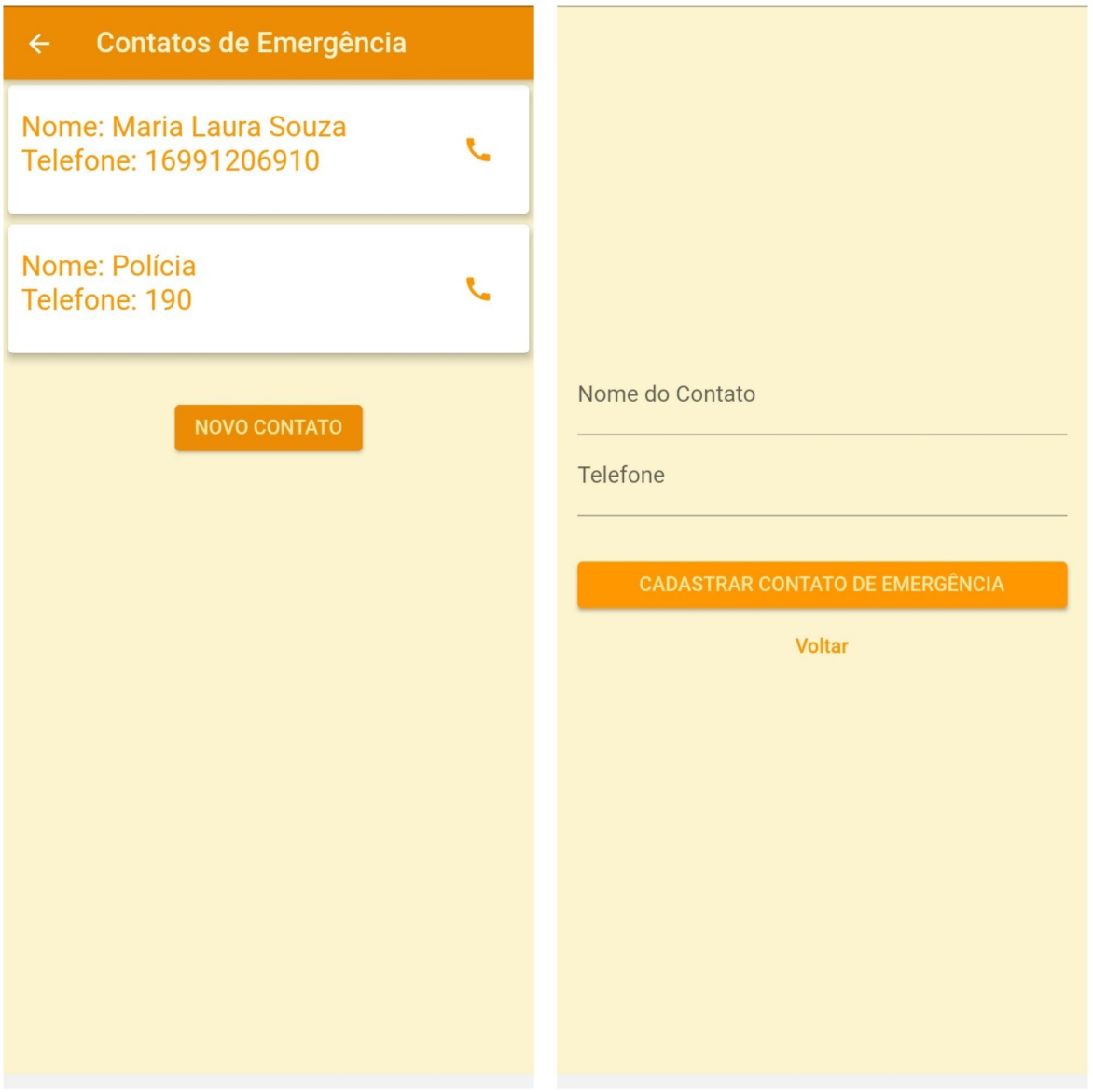
Figura 35 -Telas referentes à consulta e cadastro de ocorrências



Fonte: Produzido pelos autores

Por último, na Figura 36 tem-se as telas referentes à listagem dos contatos de emergência e cadastro de um novo contato.

Figura 36 -Telas referentes à listagem e cadastro de contatos de emergência



Fonte: Produzido pelos autores

Considerações finais

Neste trabalho foi apresentada a estrutura do desenvolvimento de um aplicativo para cadastro e consulta de ocorrências de discriminação via geolocalização. O propósito do aplicativo é tornar visível para os usuários e autoridades responsáveis os locais que tais incidências ocorrem com mais frequência e dessa forma tornar a cidade um lugar mais seguro.

Para validar a ideia e mapear os requisitos do sistema, foi desenvolvida uma pesquisa, em formato de formulário para obter informações sobre segurança e

discriminação. Com base nos resultados, constatou-se que casos de assédio, preconceito e discriminação são frequentes e que a denúncia não ocorre por medo, vergonha, insegurança, entre outros. Esse cenário evidencia a situação de vulnerabilidade a que as pessoas são expostas e de que se agrava quando não ocorre a denúncia.

Tendo isso em vista, deu-se início ao projeto a partir da especificação dos requisitos, onde também foi feita a escolha das tecnologias utilizadas no desenvolvimento e posteriormente foram desenvolvidos fluxos do processo do sistema e diagramas para representar como as funcionalidades se relacionam umas com as outras e como serão utilizadas pelo usuário.

Com toda documentação estruturada, iniciou-se o desenvolvimento do sistema exclusivamente no formato de aplicativo *mobile*, por meio da linguagem Dart, com o *framework Flutter*; consumindo uma API desenvolvida em Java e com todos os dados armazenados no banco de dados MySQL.

A partir de um cadastro inicial, o aplicativo permite consultar índices de ocorrências de determinado local, registrar uma ocorrência, cadastrar contatos de emergência e realizar ligações para estes contatos ou para a polícia. Vale ressaltar que todas as telas foram desenvolvidas com soluções amigáveis e intuitivas para o usuário final.

Ao longo do ciclo do desenvolvimento do aplicativo, um dos principais desafios enfrentados foi a necessidade da alteração de alguns requisitos que só foram conhecidos durante a sua implementação, contudo as modificações foram tratadas de forma efetiva, garantindo a qualidade do sistema. Além disso, outro ponto a se destacar foi a compreensão mais aprofundada da linguagem Dart, junto ao *framework Flutter* por parte dos autores para que pudesse atender todas as necessidades do desenvolvimento.

Para projetos futuros, considera-se desenvolver, em suas próximas versões, melhorias pontuais e novas funcionalidades, como por exemplo, compartilhar a localização em tempo real com um dos contatos de emergência cadastrados.

Conclui-se que o objetivo do projeto foi atingido, levando em conta que o software foi implementado em conformidade com o que foi especificado, podendo ser evoluído futuramente, de acordo com os requisitos necessários.

Referências

ABREU, Jadson C.; PASTANA, Claudionor O. **Introdução à Estatística Educacional**. 1ª Ed. Curitiba: Editora CRV. 2022.

AFONSO, A. **Produtividade no Desenvolvimento de Aplicações Web com Spring Boot**. 3ª Ed. Algaworks. 2017.

Alura, O que é SQL? 14/07/2019. Disponível em: <https://www.alura.com.br/artigos/o-que-e-sql>. Acesso em: 06.mar.22.

Atlantic, O que é MySQL? [s.d]. Disponível em: <https://www.atlantic.net/dedicated-server-hosting/what-is-mysql/>. Acesso em: 19.abr.22.

Brasil de Fato, Racismo: taxa de assassinatos cresce para negros e cai para o resto da população. 27/08/2020. Disponível em: < https://www.brasildefato.com.br/2020/08/27/racismo-taxa-assassinatos-de-negros-cresce-e-cai-para-o-resto-da-populacao>. Acesso em: 22.nov.20.

BOND, L. Agência Brasil, Casos de feminicídio crescem 22% em 12 estados durante pandemia**.** 01/06/2020. Disponível em: <https://agenciabrasil.ebc.com.br/direitos-humanos/noticia/2020-06/casos-de-feminicidio-crescem-22-em-12-estados-durante-pandemia>. Acesso em: 22.nov.20.

Coodesh, O que é dart? [s.d]. Disponível em: <https://coodesh.com/blog/dicionario/o-que-e-dart/>. Acesso em: 20.abr.22.

DEITEL H.; DEITEL P.; **Java Como Programar**. 6ª Ed. Porto Alegre: Editora Pearson Education. 2005.

Devmedia, Entendendo a linguagem SQL. 2008. Disponível em: <https://www.devmedia.com.br/entedendo-a-linguagem-sql/7775#:~:text=A%20linguagem%20SQL%20surgiu%20em,adapta%2Dse%20ao%20modelo%20relacional.>. Acesso em: 06.mar.22.

Devmedia, Tecnologia Flutter. [s.d]. Disponível em <https://www.devmedia.com.br/guia/flutter/40713>. Acesso em: 20.abr.22

Euax, O que é BPMN e como aplicar essa notação na Modelagem de Processos. 08/02/2017. Disponível em: <https://www.euax.com.br/2017/02/o-que-e-bpmn-business-process-model-and-notation/>. Acesso em: 26.abr.22.

FARIA, Alexandre. Introdução a Linguagem Java. 2009. Disponível em: <http://aberto.univem.edu.br/bitstream/handle/11077/746/miniCursoJavaBasico.pdf?sequence=1>. Acesso em: 18.abr.22.

FILHO, Wilson P. P. **Engenharia de Software - Produtos**. 4ª Ed. Rio de Janeiro: Editora LTC. 2019

Folha Dirigida. Mais de 43% das mulheres sofrem assédio moral nas empresas. 12/03/2020. Disponível em: < https://folhadirigida.com.br/mais/noticias/empregos/mais-de-43-das-mulheres-sofrem-assedio-moral-nas-empresas>. Acesso em: 22.nov.20.

Google Cloud, Preços. Disponível em: <https://mapsplatform.google.com/pricing/>. Acesso em: 21.abr.22.

Hostgator, quais funcionalidades o Flutter oferece aos desenvolvedores? 29/06/2020. Disponível em: <https://www.hostgator.com.br/blog/flutter-para-desenvolvedores/>. Acesso em 20.abr.22.

Hostinger, O que é MySQL: MySQL explicado para iniciantes. 13/05/2022. Disponível em: <https://www.hostinger.com/tutorials/what-is-mysql>. Acesso em 19.abr.22

JavaPoint, MySQL workbench. [s.d]. Disponível em: <https://www.javatpoint.com/mysql-workbench>. Acesso em: 19.abr.22.

PEREIRA, Daniel. O que é o Business Model Canvas. 08/07/2016. Disponível em: <https://analistamodelosdenegocios.com.br/o-que-e-o-business-model-canvas/>. Acesso em: 26.abr.22.

PRESSMAN, Roger S. **Engenharia de Software: Uma Abordagem Profissional**. 7ª Ed. São Paulo: Editora Sênior Arysinha Jacques Affonso, 2011.

REIS, Glauco. S. **Modelagem de Processos de Negócios com BPMN - Curso Completo**. 1ª Ed. São Paulo: Editora PortalBPM, 2008.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª Ed. São Paulo: Editora Pearson Addison - Wesley, 2011.

Startecjobs, Linguagem Dart: o que é e por que aprender? 25/10/2021. Disponível em: <https://startecjobs.com/artigos/carreira/linguagem-dart-o-que-e/>. Acesso em: 20.abr.22.

TechMundo, MySQL: o que é e como usar o sistema? 03/09/2021. Disponível em: <https://www.tecmundo.com.br/software/223924-mysql-usar-o-sistema.htm>. Acesso em 19.abr.22

TechTudo, O que é e como usar o MySQL? 17/04/2012. Disponível em: <https://www.techtudo.com.br/noticias/2012/04/o-que-e-e-como-usar-o-mysql.ghtml> Acesso em: 19.abr.22.

TOTVS, Como funciona e quais as vantagens da notação BPMN? 30/01/2020. Disponível em: <https://www.totvs.com/blog/gestao-industrial/bpmn/>. Acesso em: 26.abr.22.