

CID - Criminal Incident Database

Nathaly Camargo do Nascimento¹

Luan Felipe Tenroller²

Luiz Gustavo da Silva Przygoda³

Maria Cecília Schneider de Oliveira⁴

Emanuel Previatti⁵

Vinícius Andrei Wille⁶

Otilia Donato Barbosa⁷

Roberson Junior Fernandes Alves⁸

Trabalho apresentado ao curso de Ciência da Computação como parte dos requisitos para avaliação nos componentes curriculares: Banco de Dados II, Engenharia de Software I e Programação II.

¹Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
nathaly.n@unoesc.edu.br

² Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
luanf.tenroller@gmail.com

³ Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
luizprzygoda24@gmail.com

⁴Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
cicaschneider@gmail.com

⁵Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste

Rua Oiapoc, 2011. São Miguel do Oeste-SC
emanuel.previatti@unoesc.edu.br

⁶Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
vinicius.wille@unoesc.edu.br

⁷Mestre em Engenharia Biomédia e Informática Industrial
Docente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
otilia.barbosa@unoesc.edu.br

⁸Mestre em Computação Aplicada
Docente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
roberson.alves@unoesc.edu.br

SUMÁRIO

1. INTRODUÇÃO.....	5
2. CID - Criminal Incident Database.....	6
2.1 SISTEMA REQUISITADO.....	6
2.2 FUNCIONALIDADES.....	7
3. DIAGRAMAS.....	8
3.1 CASOS DE USO.....	8
3.2 SEQUÊNCIA.....	10
3.3 ATIVIDADE.....	11
3.4 ESTADO.....	12
3.5 CLASSES.....	12
4. CODIFICAÇÃO.....	13
5. MODELO RELACIONAL.....	19
6. CRIAÇÃO DO BANCO DE DADOS.....	20
6.1 SCRIPTS DE CRIAÇÃO DAS TABELAS DO BANCO.....	20
6.2 RELATÓRIOS.....	27
8. REFERÊNCIAS.....	31

FIGURAS

Figura 1, Diagrama de casos de uso - CID	8
Figura 2, Diagrama de casos de uso geral - CID	9
Figura 3, Diagrama de sequência - CID	10
Figura 4, Diagrama de atividade - CID	11
Figura 5, Diagrama de estado - CID	12
Figura 6, Diagrama de Classes - CID	13
Figura 7, Home - CID	16
Figura 8, Login cidadão - CID	16
Figura 9, Cadastro Policial - CID	17
Figura 10, Menu - CID	17
Figura 11, Registro BO - CID	18
Figura 12, Modelo relacional - CID	19

RESUMO

O **CID - Criminal Incident Database** é um sistema desenvolvido para o registro e gerenciamento de boletins de ocorrência (BOs), visando otimizar e modernizar o processo de documentação e consulta de incidentes. A plataforma oferece uma interface acessível e intuitiva, permitindo que cidadãos e agentes da segurança pública registrem e acompanhem ocorrências de maneira prática e eficiente.

Com foco na segurança pública, o CID centraliza informações essenciais, contribuindo para a organização e padronização dos dados, além de oferecer maior agilidade no acesso às informações. O sistema também reforça a importância de um ambiente digital seguro, garantindo a integridade e confidencialidade dos dados registrados.

Palavras-chave: **Registro de BOs, boletins de ocorrência, gerenciamento de ocorrências, CID, tecnologias.**

1. INTRODUÇÃO

O **CID - Criminal Incident Database** foi desenvolvido para registrar e gerenciar boletins de ocorrência (BOs), e com a necessidade de aprimorar a segurança. Diante da crescente demanda por soluções tecnológicas que otimizem processos e garantam acesso rápido e seguro a informações essenciais, o CID surge como uma ferramenta robusta e acessível, tanto para cidadãos quanto para agentes de segurança.

O texto apresenta informações essenciais para o desenvolvimento de um projeto que tem como objetivo a criação e gerenciamento de boletins de ocorrência, onde os conhecimentos abordados são essenciais para a realização de qualquer sistema; Apresentaremos ao leitor principais objetivos e funções presentes no sistema, o artigo traz imagens, código e diagramas que podem auxiliar o leitor no entendimento do conteúdo, mostraremos na prática a criação e planejamento de projeto

2. CID - Criminal Incident Database

O sistema é feito para facilitar o registro e a consulta de boletins de ocorrência (BOs). Ele ajuda tanto cidadãos quanto agentes da segurança pública a acessar e organizar informações de forma rápida e segura, centralizando tudo em um único lugar. O objetivo é tornar o processo mais prático e melhorar a segurança pública.

2.1 SISTEMA REQUISITADO

O sistema foi desenvolvido para atender as necessidades identificadas por profissionais da segurança pública, como o policial Eduardo Vargas, que trabalhou na Delegacia de Polícia de São José do Cedro – SC. Durante sua experiência no registro de boletins de ocorrência (BOs), Eduardo apontou diversos problemas nos sistemas utilizados, como um layout inadequado, com fontes pequenas e cores pouco agradáveis, tornando o uso do sistema difícil e pouco intuitivo. Além disso, a otimização do sistema era deficiente, com poucos recursos de atualização, o que tornava o processo ainda mais demorado.

Outro ponto crítico mencionado foi a classificação dos boletins de ocorrência, que muitas vezes exigia que os registros fossem classificados como "Outros", forçando os policiais a descreverem detalhadamente os incidentes. O documento gerado após o registro do boletim também não atendia às expectativas, sendo considerado pouco atrativo e difícil de entender, especialmente para leigos. Com isso, o CID busca solucionar esses problemas, oferecendo um sistema com layout mais agradável, otimização de recursos e uma classificação mais detalhada e intuitiva, além de gerar boletins mais claros e acessíveis, tanto para profissionais quanto para o público geral.

2.2 FUNCIONALIDADES

O sistema oferece uma série de funcionalidades que visam melhorar a gestão e o registro de boletins de ocorrência (BOs), atendendo tanto cidadãos quanto policiais. As principais funcionalidades do sistema incluem:

- Registro de Boletins de Ocorrência (BOs): O sistema permite que tanto cidadãos quanto policiais registrem boletins de ocorrência de forma rápida e eficiente. O processo é simplificado, com campos de fácil preenchimento e uma interface intuitiva.
- Validação de BOs pelo Policial: Após o registro do boletim, o policial tem a capacidade de validar e revisar os registros, garantindo que todas as informações estejam corretas e completas antes de serem arquivadas.
- Acesso aos Boletins de Ocorrência pelo Cidadão: Os cidadãos podem acessar e consultar seus próprios boletins de ocorrência a qualquer momento, facilitando o acompanhamento do status do registro.
- Pesquisa de BOs pelo Policial: O policial tem acesso a uma funcionalidade de pesquisa avançada, que permite buscar qualquer boletim registrado.

Além dessas funcionalidades, o CID também resolve problemas apontados por profissionais de segurança pública, como a interface melhorada, com layout mais agradável e otimizado, classificação detalhada dos incidentes, e geração de boletins de ocorrência mais claros e acessíveis, que podem ser facilmente compreendidos por leigos.

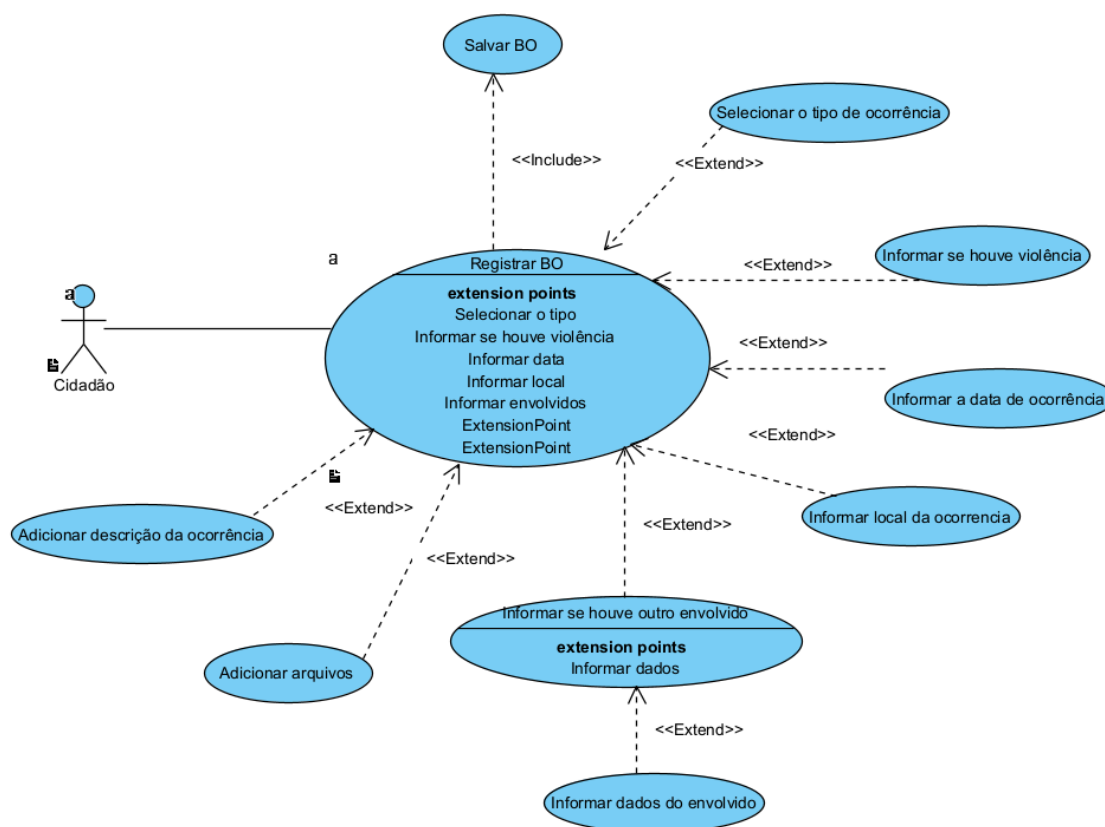
3. DIAGRAMAS

Os diagramas são ferramentas essenciais na modelagem e documentação de sistemas, permitindo representar graficamente os componentes, interações e fluxos de informações. Nesta seção, serão apresentados diferentes tipos de diagramas utilizados no desenvolvimento do CID, detalhando suas funcionalidades e a estrutura do sistema. Esses diagramas fornecem uma visão abrangente das interações entre os usuários e o sistema, facilitando a compreensão de seus processos e componentes.

3.1 CASOS DE USO

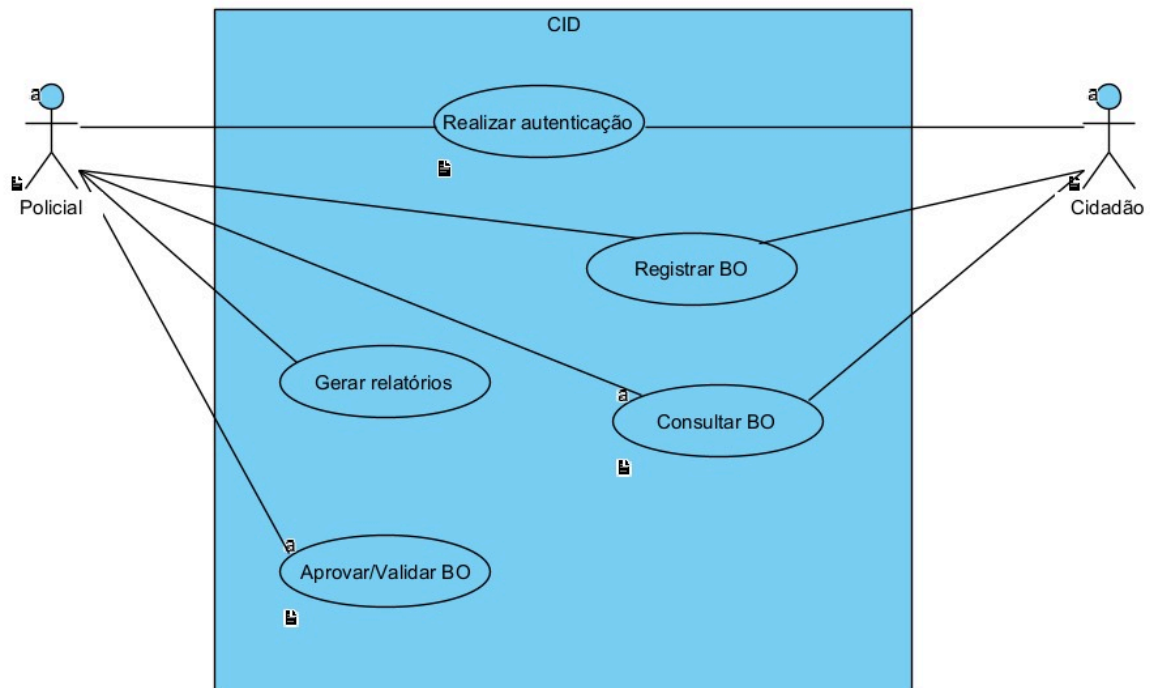
Na linguagem de modelagem unificada (UML), o diagrama de caso de uso resume os detalhes dos usuários do seu sistema (também conhecidos como atores) e as interações deles com o sistema. Na criação de um, use-se um conjunto de símbolos e conectores especializados. O diagrama de caso de uso tem como principal finalidade ajudar sua equipe a representar e discutir cenários e interações entre usuário e sistema.

Figura 1, Diagrama de casos de uso - CID



Fonte: Os autores

Figura 2, Diagrama de casos de uso geral - CID



Fonte: Os autores

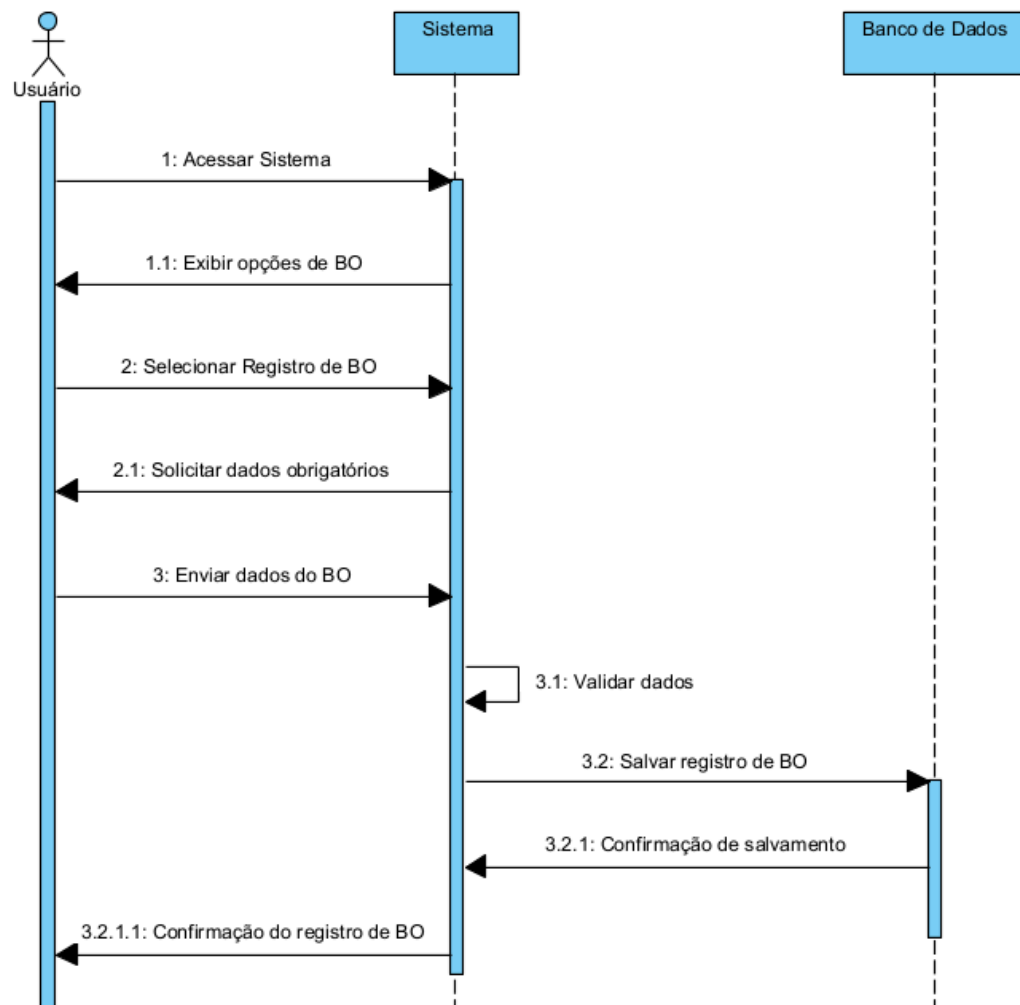
A figura 2 ilustra como o sistema CID centraliza o gerenciamento de boletins de ocorrência (BOs) ao permitir interações distintas para cada tipo de ator. Enquanto cidadãos podem registrar e consultar boletins, os policiais desempenham funções adicionais, como aprovar/validar BOs e fazer consultas específicas.

Este modelo é crucial para demonstrar os requisitos funcionais do sistema, evidenciando as permissões e responsabilidade de cada usuário.

3.2 SEQUÊNCIA

O diagrama de sequência descreve como, e em qual ordem, um grupo de objetos trabalha em conjunto. Estes diagramas são usados para entender as necessidades de um novo sistema, seus cenários e eventos ou para documentar um processo existente. O seguinte diagrama de sequência descreve o trecho da parte de registro de boletins de ocorrência.

Figura 3, Diagrama de sequência - CID



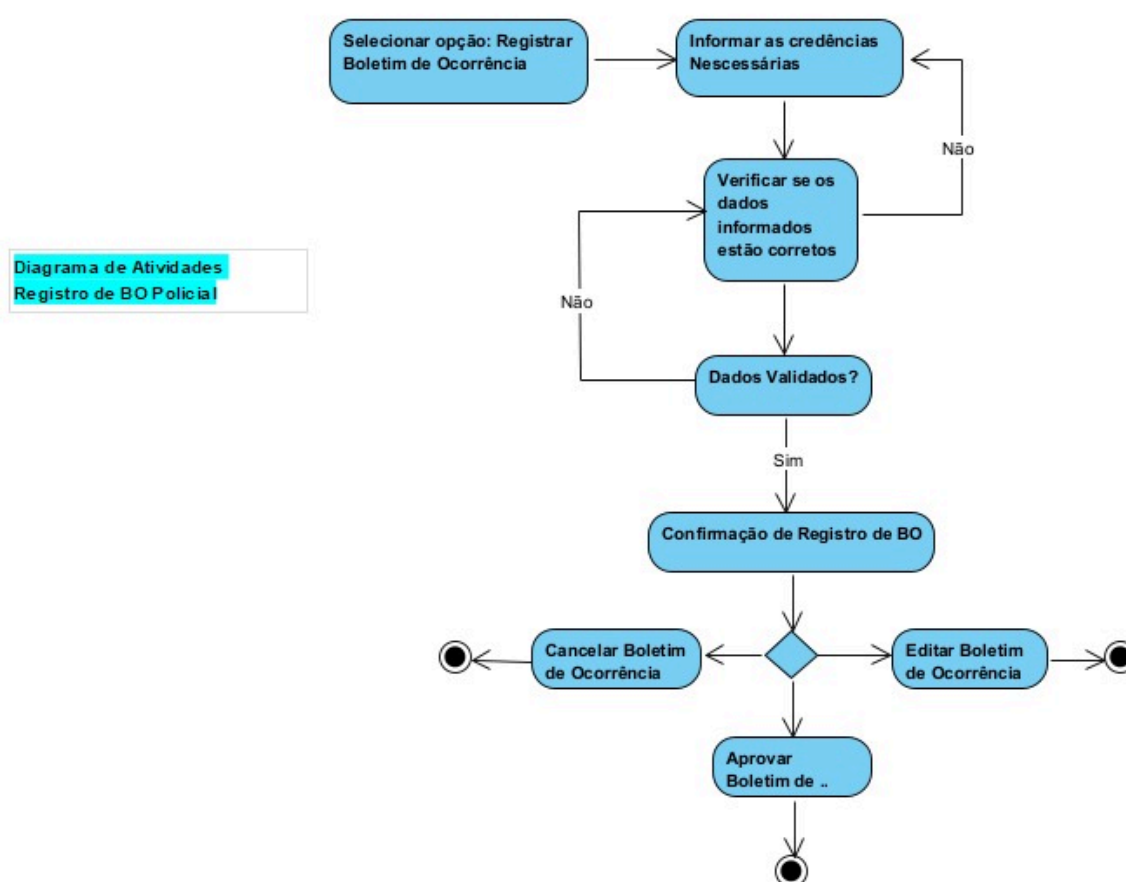
Fonte: Os autores

3.3 ATIVIDADE

Diagramas de atividade ajudam a entender o processo e comportamento do sistema. Para criar um diagrama de atividade, é necessário um conjunto de símbolos especiais, incluindo aqueles para dar partida, encerrar, fundir ou receber etapas no fluxo.

No seguinte diagrama pode ser identificado como funciona o sistema de registro de boletins de ocorrência do policial.

Figura 4, Diagrama de Atividade - CID



Fonte: Os autores

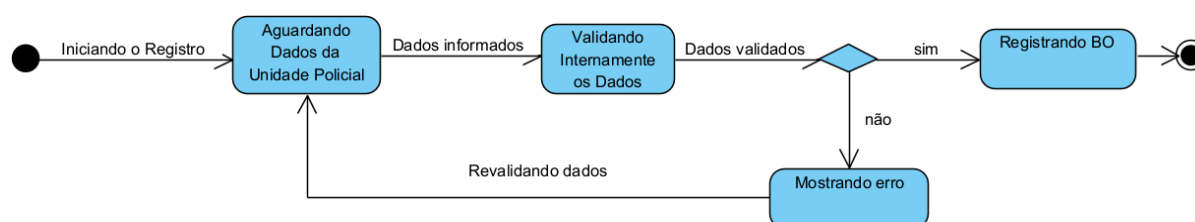
3.4 ESTADO

Em um diagrama de estado, um objeto possui um comportamento e um estado, o estado de um objeto depende da atividade na qual ele está processando. Um diagrama de estado mostra os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças de estado.

O seguinte diagrama foi baseado nos possíveis estados quando o policial registra um boletim de ocorrência.

Figura 5, diagrama de estado - CID

stm [Diagrama de Estado Registro de BO - Policial]



Fonte: Os autores

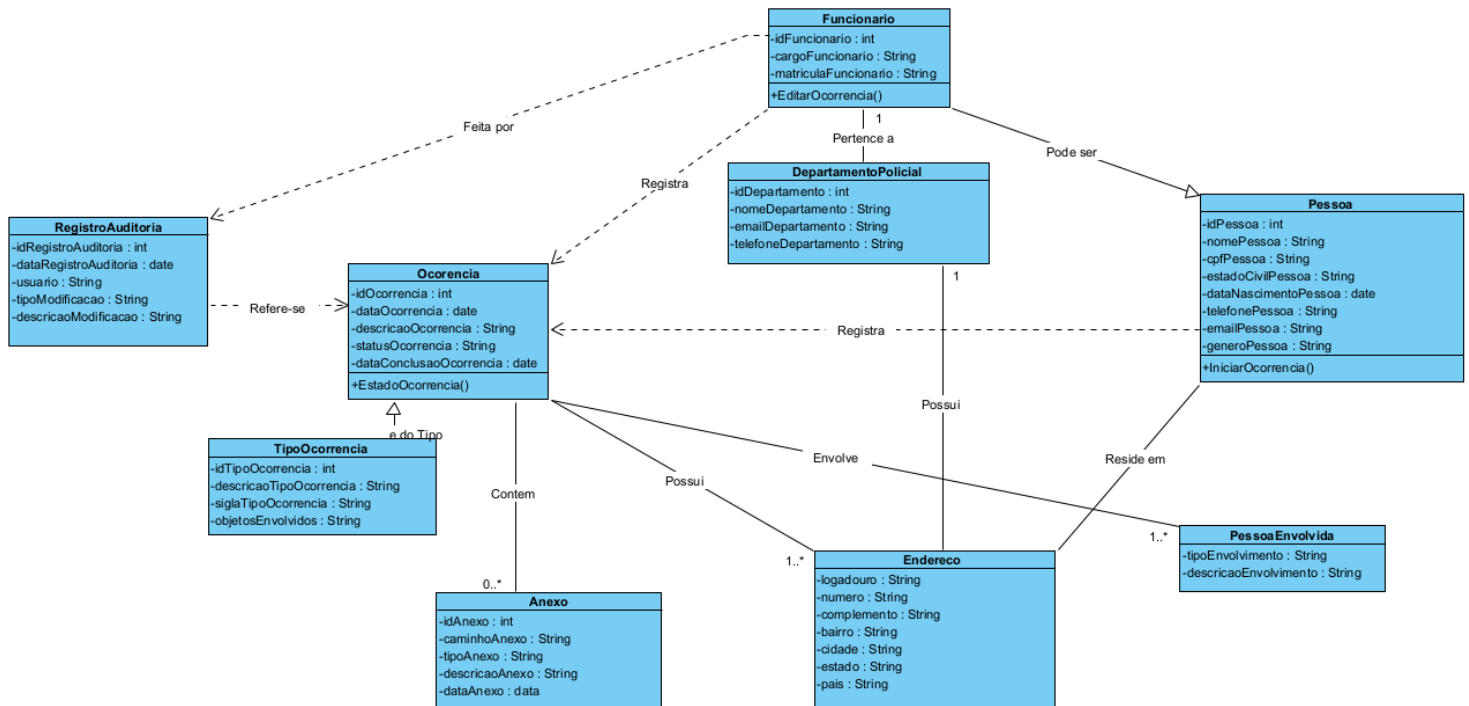
3.5 CLASSES

Diagramas de classes servem para mapear de forma clara a estrutura de um determinado sistema ao modelar suas classes, seus atributos, operações e relações entre objetos.

Os diagramas de classes são um tipo de diagrama da estrutura porque descrevem o que deve estar presente no sistema a ser modelado. Podemos afirmar de maneira mais simples que seria um conjunto de objetos com as mesmas características, assim saberemos identificar objetos e agrupá-los, de forma a encontrar suas respectivas classes.

Na imagem logo abaixo pode ser identificada todas as classes, métodos e atributos que foram necessários para o desenvolvimento do projeto.

Figura 6, diagrama de Classes - CID



Fonte: Os autores

4. CODIFICAÇÃO

O sistema foi desenvolvido utilizando Java com Spring Boot no backend, Next.js e CSS para o front-end. Na prática, o Spring Boot oferece uma série de vantagens, como a redução da complexidade na configuração inicial, permitindo que os desenvolvedores "se concentrem no código em vez de configurar a infraestrutura" (SPRING Boot, 2024). Isso contribui para projetos mais ágeis e organizados.

Ou seja, com o Spring Boot, não precisamos nos preocupar em construir uma aplicação do zero, pois, conforme Deitel e Deitel (2016), "o framework já fornece uma infraestrutura sólida e extensível", permitindo que o foco esteja na implementação das funcionalidades e regras de negócio.

Para o front-end, foi escolhido o Next.js, um framework React que oferece "uma experiência de usuário altamente eficiente" por meio de renderização no lado do servidor e

geração de sites estáticos (Next.js, 2024). Já o CSS3 foi usado para criar um design responsivo e acessível, promovendo uma experiência otimizada para os usuários.

Essas tecnologias foram escolhidas para garantir a escalabilidade, segurança e manutenção eficiente do sistema CID. Para começarmos, criamos a classe CidApplication no package br.edu.unoesc.CID. Essa será nossa classe principal, responsável por inicializar e gerenciar as chamadas para outras classes do package Model.

```
package br.edu.unoesc.CID;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CidApplication {

    public static void main(String[] args) {
        SpringApplication.run(CidApplication.class, args);
    }

}
```

Foram criados os arquivos entidades como classes utilizando os conceitos de POO e utilizando a anotação @Entity e a anotação @Table do Spring Data para a vinculação da entidade com a tabela do banco de dados como nos exemplos a seguir:

```
package br.edu.unoesc.CID.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.HashSet;
import java.util.Set;

@Entity
@Getter
@Setter
@Table(name = "bairro")
public class Bairro{

    @Id
```

```

@GeneratedValue(strategy = GenerationType.IDENTITY)

protected long idBairro;

@Column(name = "nombai", nullable = false)
private String nomeBairro;

@OneToMany(mappedBy = "bairro", cascade = CascadeType.ALL,
orphanRemoval = true)
private Set<EnderecoPessoa> enderecoPessoa = new HashSet<>();

@OneToOne(mappedBy = "bairro")
private DepartamentoPolicial departamentoPolicial;
}

```

E para a organização do Front-end, Todas as telas (como o menu, login e formulário de registro) estão organizadas na pasta components. Cada componente é separado em seu próprio arquivo .tsx, o que facilita a manutenção e a reutilização do código.

Na pasta pages o arquivo index.tsx é o ponto de entrada da aplicação. Ele gerencia as transições entre as telas (componentes) e organiza a navegação do sistema. Essa separação ajuda a manter o código limpo e a lógica centralizada.

O gerenciamento de navegação foi implementado com o hook useRouter do Next.js, permitindo transições eficientes entre páginas. Para estilização, utilizou-se o padrão de arquivos module.css, garantindo organização e encapsulamento dos estilos para cada componente.

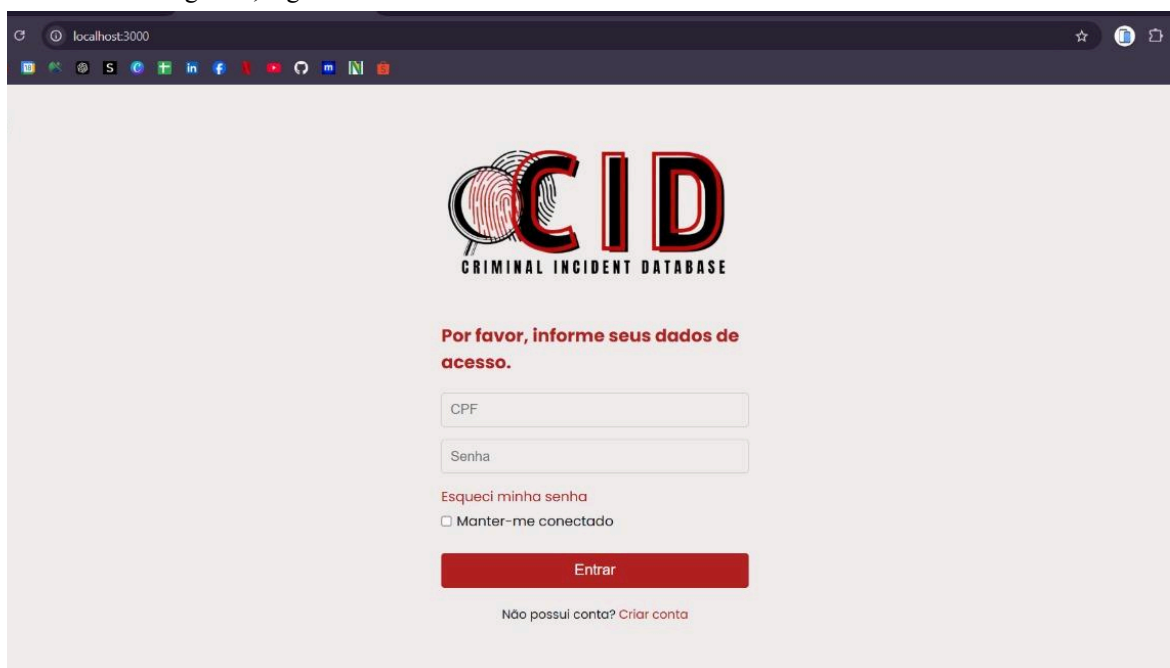
A seguir apresentamos os projetos das telas do projeto:

Figura 7, home - CID



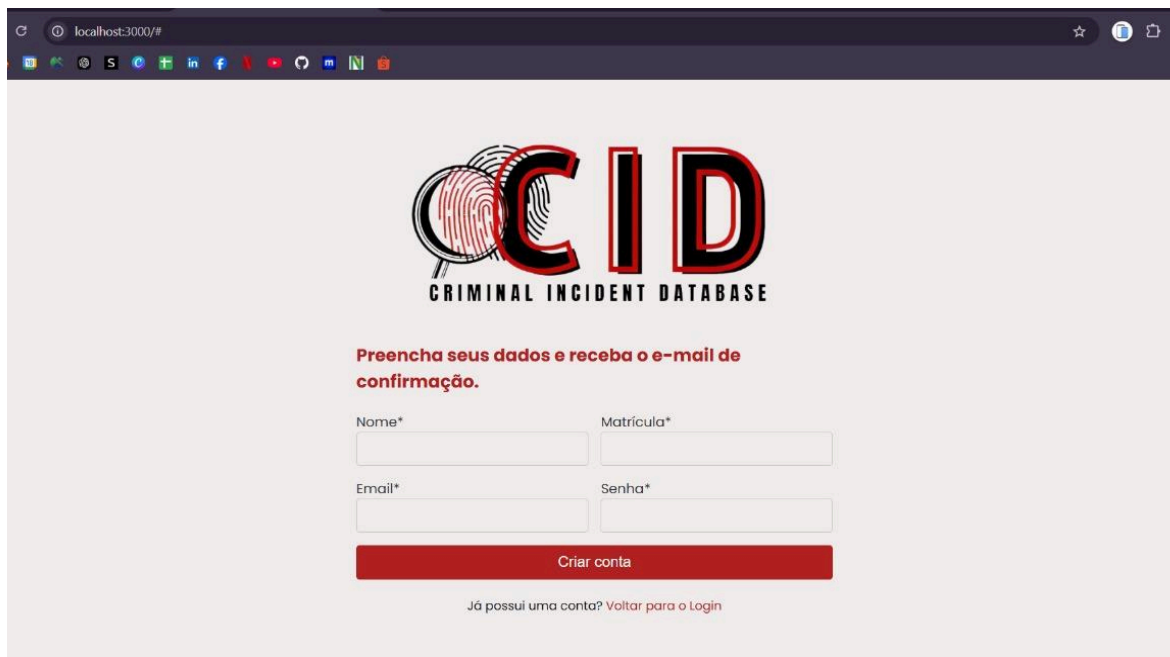
Fonte: Os autores

Figura 8, login cidadão - CID



Fonte: Os autores

Figura 9 - Cadastro policial



localhost:3000/#

CID
CRIMINAL INCIDENT DATABASE

Preencha seus dados e receba o e-mail de confirmação.

Nome* Matrícula*

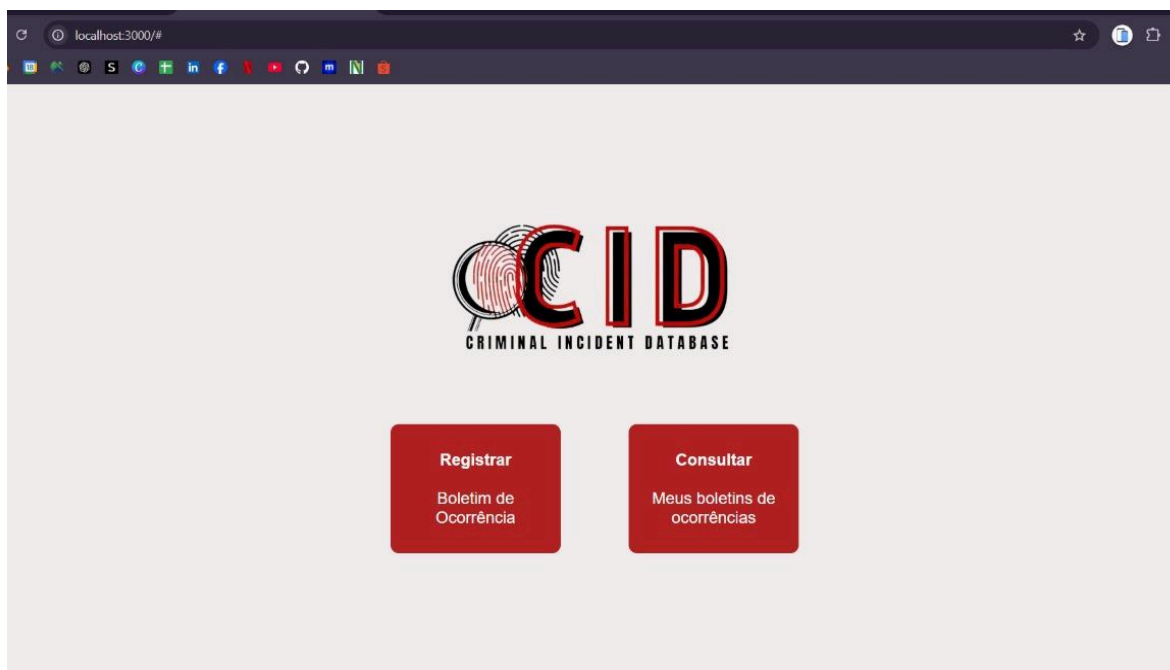
Email* Senha*

Criar conta

Já possui uma conta? [Voltar para o Login](#)

Fonte: Os autores

Figura 10 - Menu



Fonte: Os autores

Figura 11 - Registro BO

localhost:3000/#

CID
INCIDENT DATABASES

Seleção o Tipo de Ocorrência:*
Selecione

Houve Violência?* ☐ SIM ☒ NÃO

Data e Hora da Ocorrência:*
dd/mm/aaaa
--:--

Local da Ocorrência:
Cidade
Tipo de Local

Inserir um novo envolvido

Participação do envolvido*

☐ Comunicante
☐ Vítima
☐ Testemunha
☐ Suspeito

Nome Completo:
Descrição:
CPF:

Data de Nascimento:
dd/mm/aaaa

Telefone:
E-mail:

Salvar Envolvido

Descrição da Ocorrência:

Anexar Arquivos:
Escolher ficheiro Nenhum fi...elecionado

Registrar

Fonte: Os autores

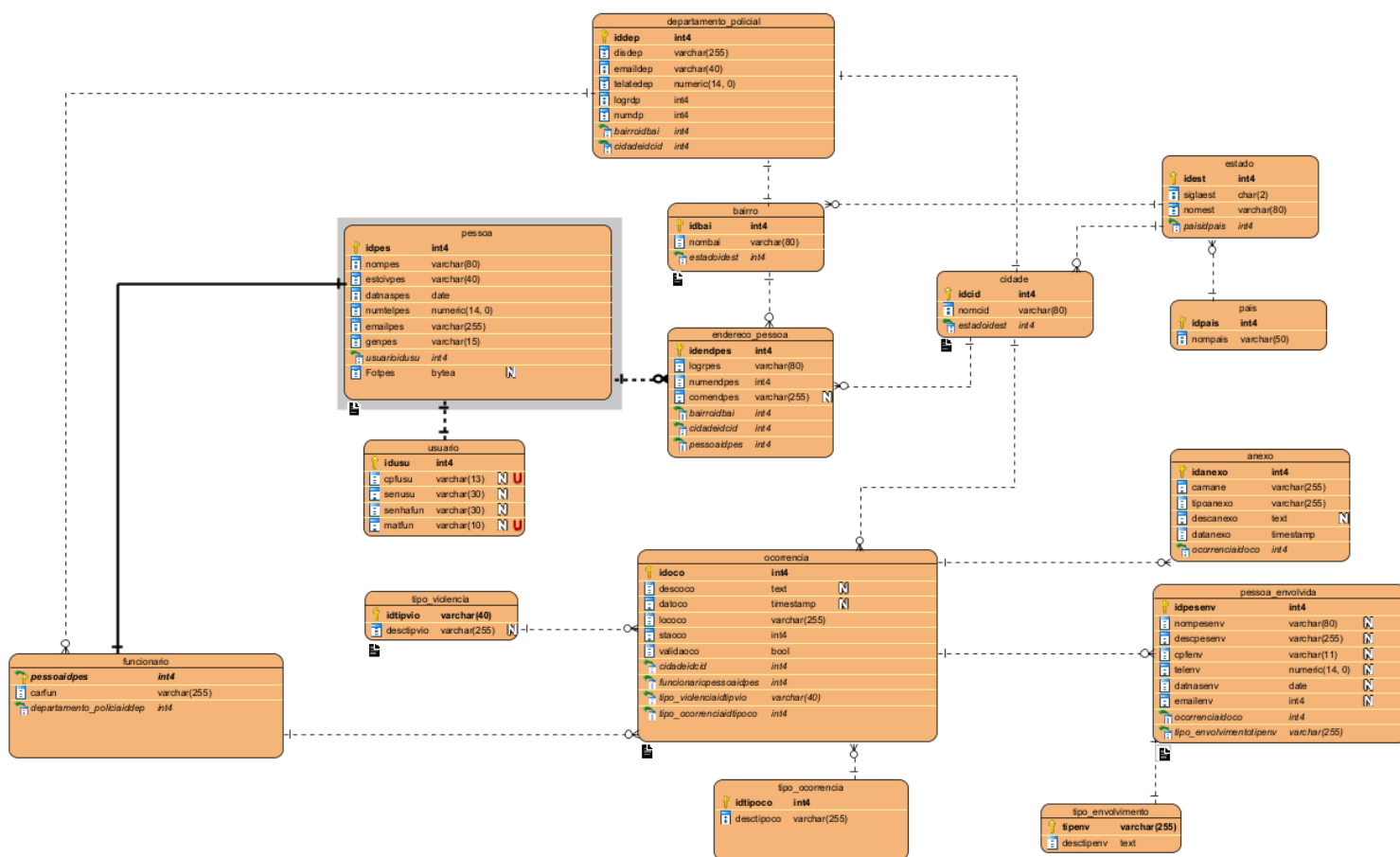
O sistema CID - Criminal Incident Database apresenta uma interface projetada para atender tanto às necessidades dos usuários cidadãos quanto dos policiais. Conforme ilustrado nas Figuras 7 e 8, a tela inicial e a tela de login oferecem uma navegação intuitiva e acessível. A interface foi desenvolvida com foco na usabilidade, permitindo que os cidadãos realizem o login para registrar ou consultar boletins de ocorrência. Já os policiais possuem acesso a funcionalidades adicionais, como a validação e a consulta detalhada de registros, garantindo maior controle e eficiência no uso do sistema.

A tela de registro de boletins de ocorrência, apresentada na Figura 11, exemplifica a simplicidade e funcionalidade do sistema. Por meio de um formulário estruturado, é possível inserir informações detalhadas sobre os incidentes, promovendo um registro rápido e completo. O projeto, incluindo sua implementação no front-end e back-end, encontra-se documentado e acessível no repositório oficial, disponível no endereço [GitHub](#).

5. MODELO RELACIONAL

O Modelo Entidade Relacionamento, descreve todas as entidades (tabelas) necessárias para o desenvolvimento do projeto, no modelo ER são descritos também todos os atributos de cada tabela, mostra as relações entre as entidades, chaves primárias, estrangeiras e o tipo do atributo

Figura 12, Modelo relacional - CID



Fonte: Os autores

6. CRIAÇÃO DO BANCO DE DADOS

O banco de dados é a base para o funcionamento do sistema CID, sendo responsável por armazenar, organizar e gerenciar as informações de forma eficiente e segura. A estrutura do banco de dados foi projetada para atender aos requisitos do sistema, garantindo escalabilidade e suporte a operações complexas. Nesta seção, são apresentados os scripts de criação das tabelas que compõem o banco de dados, detalhando suas características e relações.

6.1 SCRIPTS DE CRIAÇÃO DAS TABELAS DO BANCO

-- Tabela pais

```
CREATE TABLE pais (  
    idpais SERIAL NOT NULL,  
    nompais VARCHAR(50) NOT NULL,  
    PRIMARY KEY (idpais)  
);  
COMMENT ON COLUMN pais.idpais IS 'Id do pais';  
COMMENT ON COLUMN pais.nompais IS 'Nome do pais';
```

-- Tabela tipo de envolvimento

```
CREATE TABLE tipo_envolvimento (  
    tipenv VARCHAR(80) NOT NULL,  
    desctipenv TEXT NOT NULL,  
    PRIMARY KEY (tipenv)  
);  
COMMENT ON COLUMN tipo_envolvimento.tipenv IS 'Tipo do envolvimento';  
COMMENT ON COLUMN tipo_envolvimento.desctipenv IS 'Descrição do tipo do envolvimento';
```

-- Tabela tipo de ocorrência

```
CREATE TABLE tipo_ocorrencia (  
    idtipoco SERIAL NOT NULL,  
    desctipoco TEXT NOT NULL,  
    PRIMARY KEY (idtipoco)
```

```

);
COMMENT ON COLUMN tipo_ocorrencia.idtipoco IS 'Código do tipo de ocorrência';
COMMENT ON COLUMN tipo_ocorrencia.desctipoco IS 'Descrição do acontecimento';

-- Tabela tipo de violência
CREATE TABLE tipo_violencia (
    idtipvio VARCHAR(40) NOT NULL,
    desctipvio TEXT,
    PRIMARY KEY (idtipvio)
);
COMMENT ON TABLE tipo_violencia IS 'Tabela dos tipos de violência';
COMMENT ON COLUMN tipo_violencia.desctipvio IS 'Descrição da violência';

-- Tabela principal de usuários
CREATE TABLE usuario (
    idusu SERIAL PRIMARY KEY,
    tipo_usu VARCHAR(20) NOT NULL CHECK (tipo_usu IN ('Cidadão', 'Policial')),
    status BOOLEAN DEFAULT TRUE, -- Indica se o usuário está ativo
    criado_em TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    atualizado_em TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE usuario IS 'Tabela principal de usuários';
COMMENT ON COLUMN usuario.tipo_usu IS 'Tipo do usuário (Cidado ou Policial)';
COMMENT ON COLUMN usuario.status IS 'Status do usuário (ativo ou inativo)';

-- Tabela de acesso para cidadãos
CREATE TABLE acesso_cidado (
    idaccessoc SERIAL PRIMARY KEY,
    usuarioidusu INT NOT NULL UNIQUE,
    cpfusu VARCHAR(14) NOT NULL UNIQUE, -- CPF formatado
    senusu VARCHAR(255) NOT NULL,      -- Senha do cidadão
    FOREIGN KEY (usuarioidusu) REFERENCES usuario (idusu) ON DELETE CASCADE
);
COMMENT ON TABLE acesso_cidado IS 'Tabela de credenciais de acesso para cidadãos';

```

```
COMMENT ON COLUMN acesso_cidadao.cpfusu IS 'CPF do cidadão';
COMMENT ON COLUMN acesso_cidadao.senusu IS 'Senha do cidadão';
```

-- Tabela de acesso para policiais

```
CREATE TABLE acesso_policial (
    idaccessop SERIAL PRIMARY KEY,
    usuarioidusu INT NOT NULL UNIQUE,
    matfun VARCHAR(10) NOT NULL UNIQUE, -- Matrícula do policial
    senhafun VARCHAR(255) NOT NULL,    -- Senha do policial
    FOREIGN KEY (usuarioidusu) REFERENCES usuario (idusu) ON DELETE CASCADE
);
COMMENT ON TABLE acesso_policial IS 'Tabela de credenciais de acesso para policiais';
COMMENT ON COLUMN acesso_policial.matfun IS 'Matrícula do policial';
COMMENT ON COLUMN acesso_policial.senhafun IS 'Senha do policial';
```

-- Tabela estado

```
CREATE TABLE estado (
    idest SERIAL NOT NULL,
    siglaest CHAR(2) NOT NULL,
    nomest VARCHAR(80) NOT NULL,
    paisidpais INT4 NOT NULL,
    PRIMARY KEY (idest),
    FOREIGN KEY (paisidpais) REFERENCES pais (idpais)
);
COMMENT ON COLUMN estado.siglaest IS 'Sigla do estado';
COMMENT ON COLUMN estado.nomest IS 'Nome do estado';
```

-- Tabela cidade

```
CREATE TABLE cidade (
    idcid SERIAL NOT NULL,
    nomcid VARCHAR(80) NOT NULL,
    estadoideest INT4 NOT NULL,
    PRIMARY KEY (idcid),
    FOREIGN KEY (estadoideest) REFERENCES estado (ideest)
```

```

);
COMMENT ON TABLE cidade IS 'Tabela de cidades';
COMMENT ON COLUMN cidade.nomcid IS 'Nome da cidade';

-- Tabela bairro
CREATE TABLE bairro (
    idbai SERIAL NOT NULL,
    nombai VARCHAR(80) NOT NULL,
    estadoidest INT4 NOT NULL,
    PRIMARY KEY (idbai),
    FOREIGN KEY (estadoidest) REFERENCES estado (idest)
);
COMMENT ON TABLE bairro IS 'Tabela de bairros';
COMMENT ON COLUMN bairro.nombai IS 'Nome do bairro';

-- Tabela departamento policial
CREATE TABLE departamento_policial (
    iddep SERIAL NOT NULL,
    disdep VARCHAR(80) NOT NULL,
    emaildep VARCHAR(40) NOT NULL,
    telatedep NUMERIC(14, 0) NOT NULL,
    logrdp VARCHAR(255) NOT NULL,
    numdp INT4 NOT NULL,
    bairroidbai INT4 NOT NULL,
    cidadeidcid INT4 NOT NULL,
    PRIMARY KEY (iddep),
    FOREIGN KEY (bairroidbai) REFERENCES bairro (idbai) ON DELETE CASCADE,
    FOREIGN KEY (cidadeidcid) REFERENCES cidade (idcid) ON DELETE CASCADE
);
COMMENT ON COLUMN departamento_policial.disdep IS 'Distrito do departamento
policial';
COMMENT ON COLUMN departamento_policial.emaildep IS 'Email do departamento
policial';

```

-- Tabela pessoa

```
CREATE TABLE pessoa (  
    idpes SERIAL NOT NULL,  
    nompes VARCHAR(80) NOT NULL,  
    estciwpes VARCHAR(40) NOT NULL,  
    datnaspes DATE NOT NULL CHECK (datnaspes <= CURRENT_DATE),  
    numtelpes NUMERIC(14, 0) NOT NULL CHECK (numtelpes > 0),  
    emailpes VARCHAR(80) NOT NULL CHECK (emailpes LIKE '%@%'),  
    genpes VARCHAR(15) NOT NULL CHECK (genpes IN ('Masculino', 'Feminino',  
'Outro')),  
    usuarioidusu INT4 NOT NULL,  
    Fotpes BYTEA,  
    acesso_cidadao int4,  
    acesso_policial int4,  
    PRIMARY KEY (idpes),  
    FOREIGN KEY (acesso_cidadao) REFERENCES acesso_cidadao (idaccessoc),  
    FOREIGN KEY (acesso_policial) REFERENCES acesso_policial (idaccessop)  
);  
COMMENT ON TABLE pessoa IS 'Tabela de pessoas para registro';
```

-- Tabela funcionário

```
CREATE TABLE funcionario (  
    pessoaidpes INT4 NOT NULL,  
    carfun VARCHAR(80) NOT NULL,  
    departamento_policiaiddep INT4 NOT NULL,  
    PRIMARY KEY (pessoaidpes),  
    FOREIGN KEY (pessoaidpes) REFERENCES pessoa (idpes),  
    FOREIGN KEY (departamento_policiaiddep) REFERENCES departamento_policial  
(iddep)  
);  
COMMENT ON COLUMN funcionario.carfun IS 'Cargo do funcionário';
```

```
CREATE TABLE endereco_pessoa (  

```



```

idendpes SERIAL NOT NULL,
logrpes VARCHAR(255) NOT NULL,
numendpes INT4 NOT NULL,
comendpes VARCHAR(255),
bairroidbai INT4 NOT NULL,
cidadeidcid INT4 NOT NULL,
pessoaidpes INT4 NOT NULL,
PRIMARY KEY (idendpes),
FOREIGN KEY (pessoaidpes) REFERENCES pessoa (idpes),
FOREIGN KEY (bairroidbai) REFERENCES bairro (idbai) ON DELETE CASCADE,
FOREIGN KEY (cidadeidcid) REFERENCES cidade (idcid) ON DELETE CASCADE
);
COMMENT ON COLUMN endereco_pessoa.logrpes IS 'Rua ou avenida';
COMMENT ON COLUMN endereco_pessoa.numendpes IS 'Numero do imovel da pessoa';
COMMENT ON COLUMN endereco_pessoa.comendpes IS 'Complemento';

CREATE TABLE ocorrencia (
    idoco SERIAL NOT NULL,
    descoco TEXT,
    datoco TIMESTAMP,
    lococo VARCHAR(255) NOT NULL,
    staoco Varchar(40) NOT NULL,
    cidadeidcid INT4 NOT NULL,
    funcionario pessoaidpes INT4 NOT NULL,
    tipo_violencia idtipoco VARCHAR(40) NOT NULL,
    tipo_ocorrencia idtipoco INT4 NOT NULL,
    validaoco BOOLEAN DEFAULT FALSE,
    PRIMARY KEY (idoco),
    FOREIGN KEY (cidadeidcid) REFERENCES cidade (idcid),
    FOREIGN KEY (funcionario pessoaidpes) REFERENCES funcionario (pessoaidpes),
    FOREIGN KEY (tipo_violencia idtipoco) REFERENCES tipo_violencia (idtipvio),
    FOREIGN KEY (tipo_ocorrencia idtipoco) REFERENCES tipo_ocorrencia (idtipoco)
);
COMMENT ON TABLE ocorrencia IS 'Tabela de Ocorrências';

```

```

COMMENT ON COLUMN ocorrencia.idoco IS 'Codigo da ocorrencia';
COMMENT ON COLUMN ocorrencia.descoco IS 'Descrição da ocorrencia';
COMMENT ON COLUMN ocorrencia.datoco IS 'Data que aconteceu da ocorrencia';
COMMENT ON COLUMN ocorrencia.lococo IS 'Local da ocorrencia';
COMMENT ON COLUMN ocorrencia.staoco IS 'Status da ocorrencia';

```

```

CREATE TABLE pessoa_envolvida (
    idpesenv SERIAL NOT NULL,
    nompesenv VARCHAR(80),
    descpesenv text,
    cpfenv VARCHAR(11),
    telenv NUMERIC(14, 0) CHECK (telenv > 0),
    datnasenv DATE CHECK (datnasenv <= CURRENT_DATE),
    emailenv varchar(80),
    ocorrenciaidoco INT4 NOT NULL,
    tipo_envolvimentotipenv VARCHAR(255) NOT NULL,
    PRIMARY KEY (idpesenv),
    FOREIGN KEY (ocorrenciaidoco) REFERENCES ocorrencia (idoco) ON DELETE
    CASCADE,
    FOREIGN KEY (tipo_envolvimentotipenv) REFERENCES tipo_envolvimento (tipenv)
    ON DELETE CASCADE
);
COMMENT ON TABLE pessoa_envolvida IS 'Tabela de pessoas envolvidas';
COMMENT ON COLUMN pessoa_envolvida.nompesenv IS 'Nome da pessoa envolvida';
COMMENT ON COLUMN pessoa_envolvida.descpesenv IS 'Descrição da pessoa
envolvida';
COMMENT ON COLUMN pessoa_envolvida.cpfenv IS 'Cpf da pessoa envolvida';
COMMENT ON COLUMN pessoa_envolvida.telenv IS 'Telefone da pessoa envolvida';
COMMENT ON COLUMN pessoa_envolvida.datnasenv IS 'Data de nascimento da pessoa
envolvida';
COMMENT ON COLUMN pessoa_envolvida.emailenv IS 'Email da pessoa envolvida';

```

```

CREATE TABLE anexo (
    idanexo SERIAL NOT NULL,

```

```

camane VARCHAR(255) NOT NULL,
tipoanexo VARCHAR(255) NOT NULL CHECK (tipoanexo IN ('imagem', 'video',
'documento')),
descanexo TEXT,
datanexo TIMESTAMP NOT NULL,
ocorrenciaidoco INT4 NOT NULL,
PRIMARY KEY (idanexo),
FOREIGN KEY (ocorrenciaidoco) REFERENCES ocorrencia (idoco)
);
COMMENT ON COLUMN anexo.camane IS 'Caminho do arquivo anexado';
COMMENT ON COLUMN anexo.tipoanexo IS 'Tipo do anexo (video, imagem,
documento...)';
COMMENT ON COLUMN anexo.descanexo IS 'Descrição do anexo';
COMMENT ON COLUMN anexo.datanexo IS 'Data que o anexo foi carregado';

```

6.2 RELATÓRIOS

--- View que mostra informações detalhadas sobre cada ocorrência, incluindo o tipo de ocorrência, o local e o funcionário responsável.

```

CREATE VIEW vw_ocorrencias_detalhadas AS
SELECT
    o.idoco AS id_ocorrencia,
    o.datoco AS data_ocorrencia,
    o.lococo AS local_ocorrencia,
    t.desctipoco AS tipo_ocorrencia,
    f.carfun AS cargo_funcionario,
    p.nompes AS nome_funcionario
FROM ocorrencia o
JOIN tipo_ocorrencia t ON o.tipo_ocorrenciaidtipoco = t.idtipoco
JOIN funcionario f ON o.funcionarioidpes = f.pessoaidpes
JOIN pessoa p ON f.pessoaidpes = p.idpes;

```

-- 1. Pessoas com idades entre 20 e 30 anos e do sexo feminino, ordenadas pelo nome em ordem descendente.

```

CREATE VIEW vw_pessoas_20_30_feminino AS
SELECT
    idpes,
    nompes,

```

```

    EXTRACT(YEAR FROM AGE(datnaspes)) AS idade,
    genpes
FROM
    pessoa
WHERE
    genpes = 'Feminino'
    AND EXTRACT(YEAR FROM AGE(datnaspes)) BETWEEN 20 AND 30
ORDER BY
    nompes DESC;

```

-- 2. Ocorrências registradas em meses pares de 2023 nas cidades de São Miguel do Oeste e Descanso, ordenadas pelo tipo de ocorrência.

```

CREATE VIEW vw_ocorrencias_meses_pares_2023 AS
SELECT
    o.idoco,
    o.descoco,
    o.datoco,
    o.lococo,
    o.staoco,
    t.desctipoco AS tipo_ocorrencia,
    c.nomcid AS cidade
FROM
    ocorrencia o
JOIN
    tipo_ocorrencia t ON o.tipo_ocorrenciaidtipoco = t.idtipoco
JOIN
    cidade c ON o.cidadeidcid = c.idcid
WHERE
    EXTRACT(YEAR FROM o.datoco) = 2023
    AND EXTRACT(MONTH FROM o.datoco) IN (2, 4, 6, 8, 10, 12)
    AND c.nomcid IN ('São Miguel do Oeste', 'Descanso')
ORDER BY
    t.desctipoco ASC;

```

-- 3. Departamentos de polícia que registraram casos de roubo em 2024 nas cidades especificadas, ordenados por número de ocorrências.

```

CREATE VIEW vw_departamentos_roubo_2024 AS
SELECT
    dp.iddep,
    dp.disdep AS distrito,
    c.nomcid AS cidade,
    COUNT(o.idoco) AS total_ocorrencias

```

```

FROM
    departamento_policial dp
JOIN
    ocorrencia o ON dp.iddep = o.funcionariopeessoaidpes
JOIN
    tipo_ocorrencia t ON o.tipo_ocorrenciaidtipoco = t.idtipoco
JOIN
    cidade c ON dp.cidadeidcid = c.idcid
WHERE
    t.desctipoco = 'Roubo'
    AND EXTRACT(YEAR FROM o.datoco) = 2024
    AND c.nomcid IN ('Maravilha', 'Descanso', 'Itapiranga', 'Guaraciaba')
GROUP BY
    dp.iddep, dp.disdep, c.nomcid
ORDER BY
    total_ocorrencias DESC;
-----

-- 4. Total de ocorrências por tipo e cidade, ordenado da cidade com mais para menos
ocorrências.
CREATE VIEW vw_ocorrencias_por_tipo_cidade AS
SELECT
    t.desctipoco AS tipo_ocorrencia,
    c.nomcid AS cidade,
    COUNT(o.idoco) AS total_ocorrencias
FROM
    ocorrencia o
JOIN
    tipo_ocorrencia t ON o.tipo_ocorrenciaidtipoco = t.idtipoco
JOIN
    cidade c ON o.cidadeidcid = c.idcid
GROUP BY
    t.desctipoco, c.nomcid
ORDER BY
    total_ocorrencias DESC;
-----

```

7. CONCLUSÃO

O CID - Criminal Incident Database é um sistema inovador que se destaca por modernizar e simplificar o processo de registro e consulta de boletins de ocorrência (BOs). Desenvolvido com base em necessidades reais apontadas por profissionais da segurança pública, o CID apresenta uma solução robusta e intuitiva, que beneficia tanto cidadãos quanto agentes da segurança pública.

Com funcionalidades como registro e validação de BOs, consulta simplificada e interface amigável, o sistema não apenas resolve os problemas identificados em plataformas anteriores, mas também oferece uma experiência otimizada para seus usuários. O uso de tecnologias modernas, como Java com Spring Boot no backend e Next.js no frontend, reforça a eficiência, segurança e escalabilidade do sistema, garantindo que ele atenda às demandas atuais e futuras.

Ao centralizar e padronizar informações, o CID contribui diretamente para a organização e agilidade no gerenciamento de dados de ocorrências, tornando-se uma ferramenta essencial no contexto da segurança pública. Assim, o sistema não apenas melhora processos operacionais, mas também fortalece a confiança dos cidadãos e profissionais na utilização de soluções tecnológicas no dia a dia.

8. REFERÊNCIAS

CHEN, Peter. **Modelagem de dados: a abordagem Entidade-Relacionamento para Projeto Lógico**. São Paulo: Makron Books, 1990. 80 p.

DEITEL, Paul; DEITEL, Harvey. **Java: Como Programar**. 10. ed. São Paulo: Pearson, 2016.

DOUGLAS, Korry; DOUGLAS, Susan. **PostgreSQL: Introduction and Concepts**. Boston: Addison-Wesley, 2002. 480 p.

GONZAGA, Jorge Luiz. **Dominando o PostgreSQL**. Rio de Janeiro: Ciência Moderna, 2007. 228 p.

JOHNSON, Rod; HOELLER, Juergen. **Expert One-on-One J2EE Development without EJB**. Indianapolis: Wiley, 2004. 576 p.

MEYER, Eric. **CSS: The Definitive Guide**. 4. ed. Sebastopol: O'Reilly Media, 2017. 951 p.

Next.js. **The React Framework for Production**. Disponível em: <https://nextjs.org/>. Acesso em: 13 out. 2024.

PostgreSQL Global Development Group. **PostgreSQL Documentation**. Disponível em: <https://www.postgresql.org/docs/>. Acesso em: 2 nov. 2024.

SHAH, Tushar. "Bridging the Gap Between Next.js and Spring Boot: A Full-Stack Perspective". *Web Development Journal*. v. 4, n. 2, p. 78-90, 2023.

SPRING Boot. Disponível em: <https://spring.io/projects/spring-boot>. Acesso em: 2 out. 2024.

VASCONCELOS, Estevão; OLIVEIRA, Jomar. **PostgreSQL 14: Banco de Dados para Aplicações Modernas**. São Paulo: Novatec Editora, 2022.