ATA DE REUNIÃO

Data: 11 de novembro de 2024 às 14 Horas
Local: Discord
Pauta: Realizar uma revisão dos requisitos e elaborar o diagrama de classes, utilizando o Diagrama Entidade-Relacionamento (DER) como base para identificar entidades, atributos e suas relações.

Objetivos:

- Revisão dos requisitos principais do projeto para assegurar alinhamento com o escopo.
- Análise do DER existente, identificação das entidades e atributos necessários para o diagrama de classes.
- Elaboração do diagrama de classes, especificando atributos e relações entre as classes com base no DER.
- Discussão sobre padrões de nomenclatura e estrutura para garantir consistência.

Participantes:

- Jerônimo Barbieri Junior
- Pedro Ferreira Leite
- Miguel Miranda Morandini
- Rafael Romano Silva

Ações a Serem Tomadas:

- Todos os participantes revisarão a consistência entre o DER e o diagrama de classes e fará ajustes, se necessário.
- A equipe revisará o diagrama completo para feedback antes do início da implementação do backend.
- Todos os participantes finalizarão o diagrama de classes.

Próximos Passos para o Backend:

- Após o refinamento final do diagrama de classes, será iniciado o desenvolvimento das entidades no backend com base na modelagem definida.
- Planejamento de reunião de acompanhamento para revisar a implementação inicial e alinhar possíveis ajustes necessários.

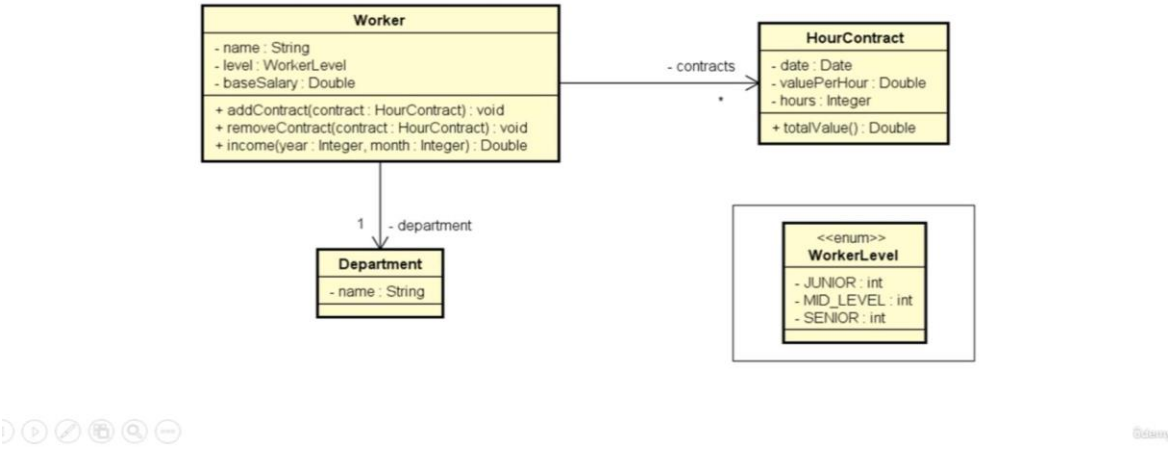
Encerramento:

- A previsão de encerramento da reunião é às 16H30.
- A próxima reunião foi agendada para 13/11/2024.

Material de Apoio:

1. Modelo de Diagrama de Classes para Padronização
    a. Elaborar com base no modelo a seguir:

Ler os dados de um trabalhador com N contratos (N fornecido pelo usuário). Depois, solicitar do usuário um mês e mostrar qual foi o salário do funcionário nesse mês, conforme exemplo (próxima página).



2. Tabela de referência de tipos primitivos entre C# e SQL Server
    a. Modelar os tipos primitivos de SQL Server para C#:
    b. Site apoio: https://cbsa.com.br/post/tipos-de-dados-equivalentes-do-c-e-sql-valores-maximos-e-minimos-suportados.aspx

C#	SQL Server	Descrição
`int`	`INT`	Número inteiro de 4 bytes.
`short`	`SMALLINT`	Número inteiro de 2 bytes.
`long`	`BIGINT`	Número inteiro de 8 bytes.
`byte`	`TINYINT`	Número inteiro de 1 byte.
`bool`	`BIT`	Representa valores booleanos (0 ou 1).
`float`	`REAL`	Número de ponto flutuante de precisão simples.
`double`	`FLOAT`	Número de ponto flutuante de precisão dupla.
`decimal`	`DECIMAL(p,s)`	Número decimal, `p` para precisão e `s` para escala.
`char`	`CHAR(1)`	Caracter único (texto de 1 caractere).
`string`	`VARCHAR(n)` / `NVARCHAR(n)`	Cadeia de caracteres variável, `n` define o tamanho máximo.
`DateTime`	`DATETIME`	Representa data e hora até milissegundos.
`DateTimeOffset`	`DATETIMEOFFSET`	Representa data e hora com fuso horário.
`TimeSpan`	`TIME`	Representa intervalo de tempo.
`Guid`	`UNIQUEIDENTIFIER`	Identificador único global (UUID/GUID).

3. Script SQL Server utilizado
    a. Embasar nesse Script para realização do diagrama de classes:

```
CREATE DATABASE db_cidade_unida

USE db_cidade_unida;

-- Tabela Usuario
CREATE TABLE tb_usuario (
   id_usuario INT NOT NULL IDENTITY(1,1),
   nome VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL, -- Garante que o e-mail é único
   senha VARCHAR(255) NOT NULL,
   is_adm BIT DEFAULT 0 NOT NULL,
   ativo BIT DEFAULT 1 NOT NULL,
CONSTRAINT pk_usuario PRIMARY KEY (id_usuario)
);

-- Tabela Telefones
CREATE TABLE tb_telefones (
   id_usuario INT NOT NULL,
   telefone VARCHAR(15) NOT NULL,
   CONSTRAINT pk_telefones PRIMARY KEY (id_usuario, telefone),
CONSTRAINT fk_telefones_usuario FOREIGN KEY (id_usuario)
REFERENCES tb_usuario (id_usuario)
);

-- Tabela Status
CREATE TABLE tb_status (
   id_status INT NOT NULL IDENTITY(1,1),
   nome_status VARCHAR(50) NOT NULL,
CONSTRAINT pk_status PRIMARY KEY (id_status)
);

-- Tabela Categoria
CREATE TABLE tb_categorias (
   id_categoria INT NOT NULL IDENTITY(1,1),
   categoria VARCHAR(100) NOT NULL,
CONSTRAINT pk_categoria PRIMARY KEY (id_categoria)
);

-- Tabela Denuncia
CREATE TABLE tb_denuncia (
   id_denuncia INT NOT NULL IDENTITY(1,1),
   descricao VARCHAR(MAX) NOT NULL,
   id_status INT NOT NULL,
   id_categoria INT NOT NULL,
   rua VARCHAR(100) NOT NULL,
   numero VARCHAR(10) NOT NULL,
   bairro VARCHAR(50) NOT NULL,
   cidade VARCHAR(50) NOT NULL,
   estado CHAR(2) NOT NULL,
   cep VARCHAR(10) NOT NULL,
   url_imagem VARCHAR(255) NULL,
   is_anonimo BIT DEFAULT 0 NOT NULL,
   data_envio DATETIME DEFAULT GETDATE() NOT NULL,
   ativo BIT DEFAULT 1 NOT NULL,
CONSTRAINT pk_denuncia PRIMARY KEY (id_denuncia),
CONSTRAINT fk_denuncia_status FOREIGN KEY (id_status)
```

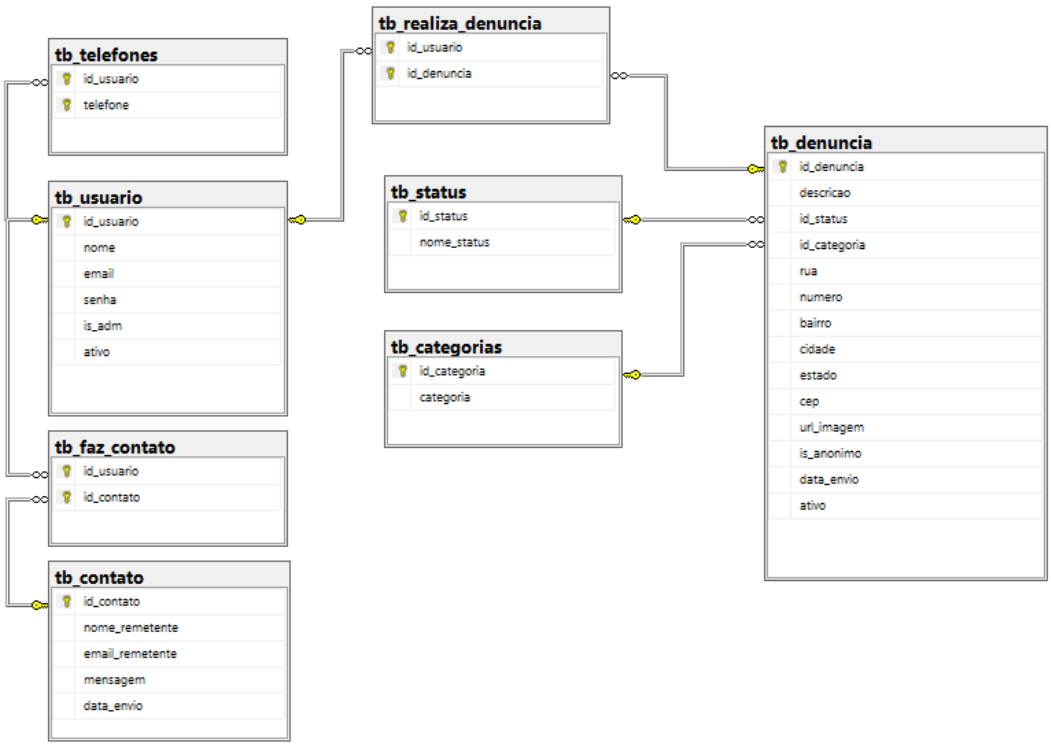
```
REFERENCES tb_status (id_status),
CONSTRAINT fk_denuncia_categoria FOREIGN KEY (id_categoria)
REFERENCES tb_categorias (id_categoria)
);

-- Tabela Realiza Denuncia (relacionamento entre Usuario e Denuncia)
CREATE TABLE tb_realiza_denuncia (
   id_usuario INT NOT NULL,
   id_denuncia INT NOT NULL,
CONSTRAINT pk_realiza_denuncia PRIMARY KEY (id_usuario, id_denuncia),
CONSTRAINT fk_realiza_denuncia_usuario FOREIGN KEY (id_usuario)
REFERENCES tb_usuario (id_usuario),
CONSTRAINT fk_realiza_denuncia_denuncia FOREIGN KEY (id_denuncia)
REFERENCES tb_denuncia (id_denuncia)
);

-- Tabela Contato
CREATE TABLE tb_contato (
   id_contato INT NOT NULL IDENTITY(1,1),
   nome_remetente VARCHAR(100) NOT NULL,
   email_remetente VARCHAR(100) NOT NULL,
   mensagem VARCHAR(MAX) NOT NULL,
   data_envio DATETIME DEFAULT GETDATE(),
CONSTRAINT pk_contato PRIMARY KEY (id_contato)
);

-- Tabela Faz Contato (relacionamento entre Usuario e Contato)
CREATE TABLE tb_faz_contato (
   id_usuario INT NOT NULL,
   id_contato INT NOT NULL,
CONSTRAINT pk_faz_contato PRIMARY KEY (id_usuario, id_contato),
CONSTRAINT fk_faz_contato_usuario FOREIGN KEY (id_usuario)
REFERENCES tb_usuario (id_usuario),
CONSTRAINT fk_faz_contato_contato FOREIGN KEY (id_contato)
REFERENCES tb_contato (id_contato)
);
```

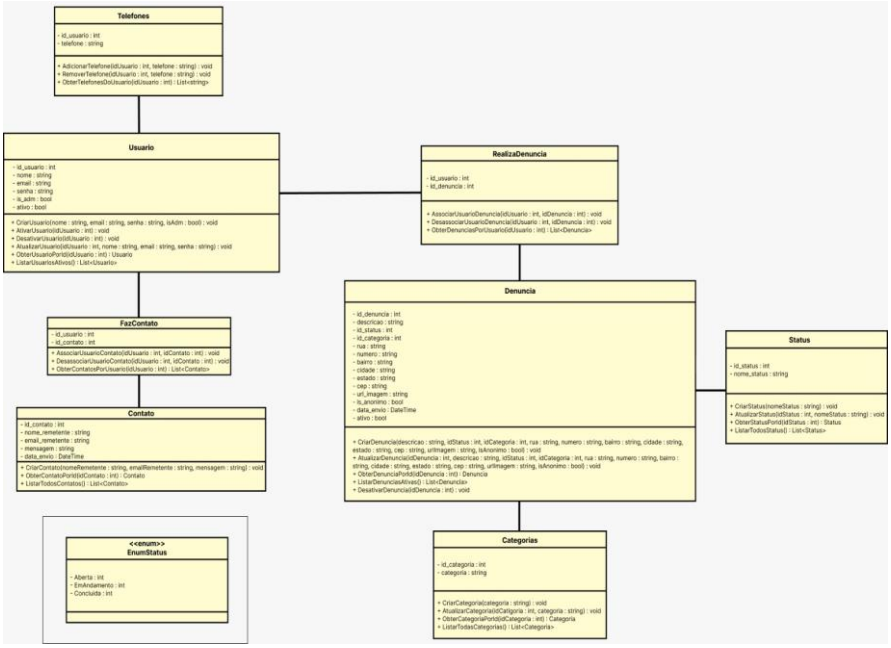
4. Diagrama do Banco de Dados

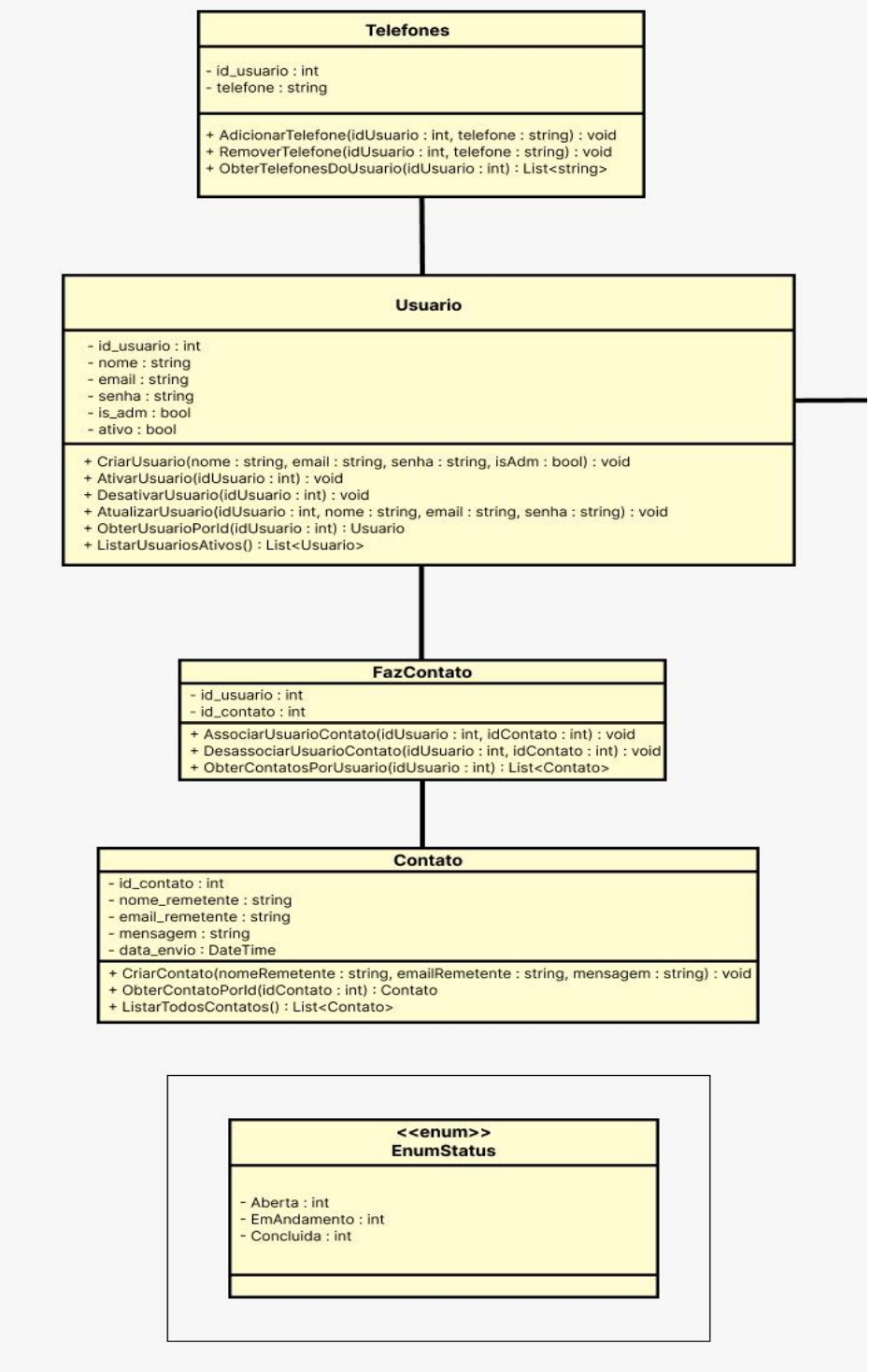


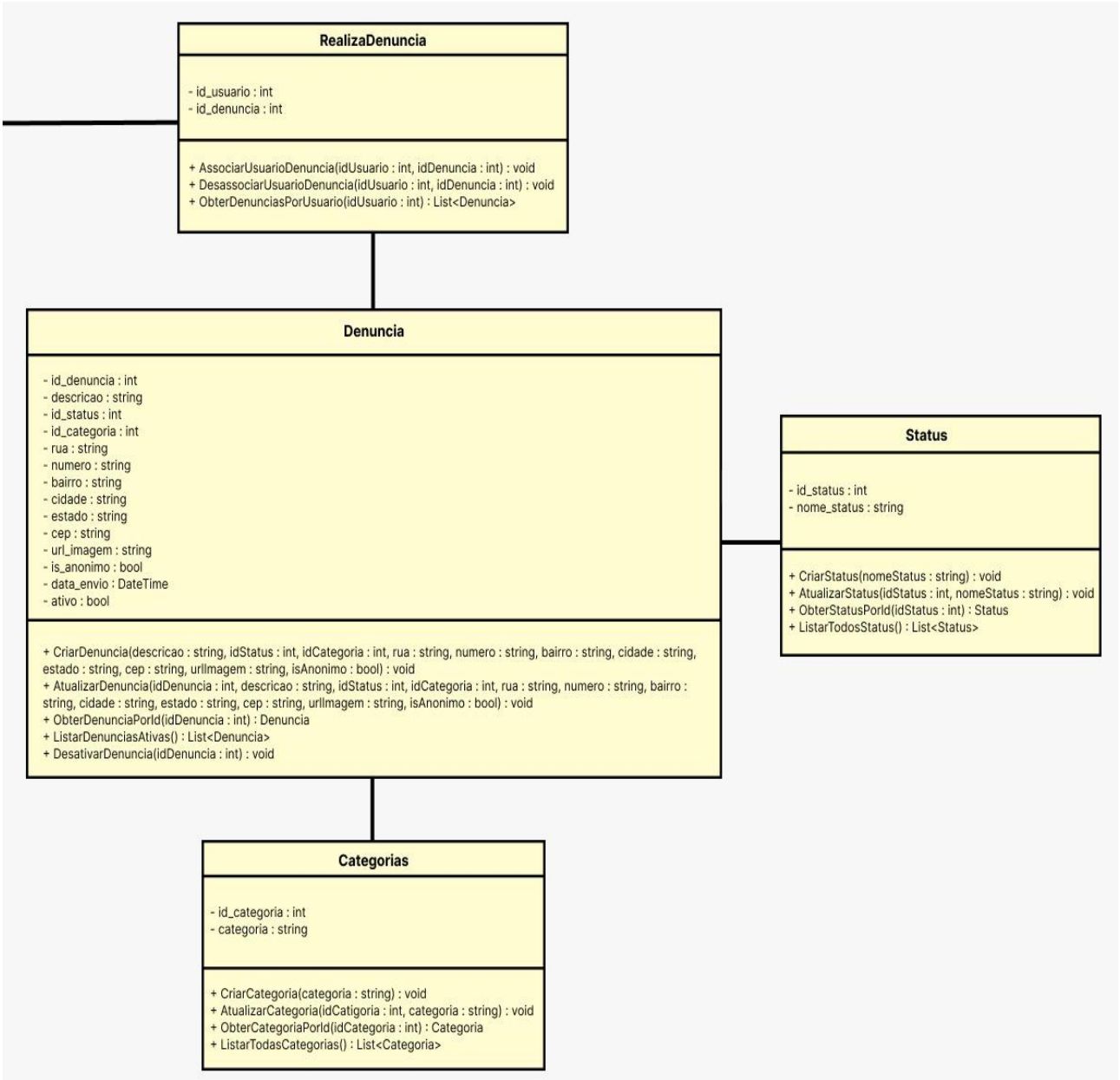
Resultado:

- Todos os objetivos da reunião foram atingidos como o esperado.

1. Diagrama de Classes Elaborado




Telefones
- id_usuario : int
- telefone : string
+ AdicionarTelefone(idUsuario : int, telefone : string) : void
+ RemoverTelefone(idUsuario : int, telefone : string) : void
+ ObterTelefonesDoUsuario(idUsuario : int) : List<string>
Usuario
- id_usuario : int
- nome : string
- email : string
- senha : string
- is_adm : bool
- ativo : bool
+ CriarUsuario(nome : string, email : string, senha : string, isAdm : bool) : void
+ AtivarUsuario(idUsuario : int) : void
+ DesativarUsuario(idUsuario : int) : void
+ AtualizarUsuario(idUsuario : int, nome : string, email : string, senha : string) : void
+ ObterUsuarioPorId(idUsuario : int) : Usuario
+ ListarUsuariosAtivos() : List<Usuario>
FazContato
- id_usuario : int
- id_contato : int
+ AssociarUsuarioContato(idUsuario : int, idContato : int) : void
+ DesassociarUsuarioContato(idUsuario : int, idContato : int) : void
+ ObterContatosPorUsuario(idUsuario : int) : List<Contato>
Contato
- id_contato : int
- nome_remetente : string
- email_remetente : string
- mensagem : string
- data_envio : DateTime
+ CriarContato(nomeRemetente : string, emailRemetente : string, mensagem : string) : void
+ ObterContatoPorId(idContato : int) : Contato
+ ListarTodosContatos() : List<Contato>
<<enum>>
EnumStatus
- Aberta : int
- EmAndamento : int
- Concluida : int


RealizaDenuncia
- id_usuario : int
- id_denuncia : int
+ AssociarUsuarioDenuncia(idUsuario : int, idDenuncia : int) : void
+ DesassociarUsuarioDenuncia(idUsuario : int, idDenuncia : int) : void
+ ObterDenunciasPorUsuario(idUsuario : int) : List<Denuncia>
Denuncia
- id_denuncia : int
- descricao : string
- id_status : int
- id_categoria : int
- rua : string
- numero : string
- bairro : string
- cidade : string
- estado : string
- cep : string
- url_imagem : string
- is_anonimo : bool
- data_envio : DateTime
- ativo : bool
+ CriarDenuncia(descricao : string, idStatus : int, idCategoria : int, rua : string, numero : string, bairro : string, cidade : string, estado : string, cep : string, urlImagem : string, isAnonimo : bool) : void
+ AtualizarDenuncia(idDenuncia : int, descricao : string, idStatus : int, idCategoria : int, rua : string, numero : string, bairro : string, cidade : string, estado : string, cep : string, urlImagem : string, isAnonimo : bool) : void
+ ObterDenunciaPorId(idDenuncia : int) : Denuncia
+ ListarDenunciasAtivas() : List<Denuncia>
+ DesativarDenuncia(idDenuncia : int) : void
Status
- id_status : int
- nome_status : string
+ CriarStatus(nomeStatus : string) : void
+ AtualizarStatus(idStatus : int, nomeStatus : string) : void
+ ObterStatusPorId(idStatus : int) : Status
+ ListarTodosStatus() : List<Status>
Categorias
- id_categoria : int
- categoria : string
+ CriarCategoria(categoria : string) : void
+ AtualizarCategoria(idCatigoria : int, categoria : string) : void
+ ObterCategoriaPorId(idCategoria : int) : Categoria
+ ListarTodasCategorias() : List<Categoria>