**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:**

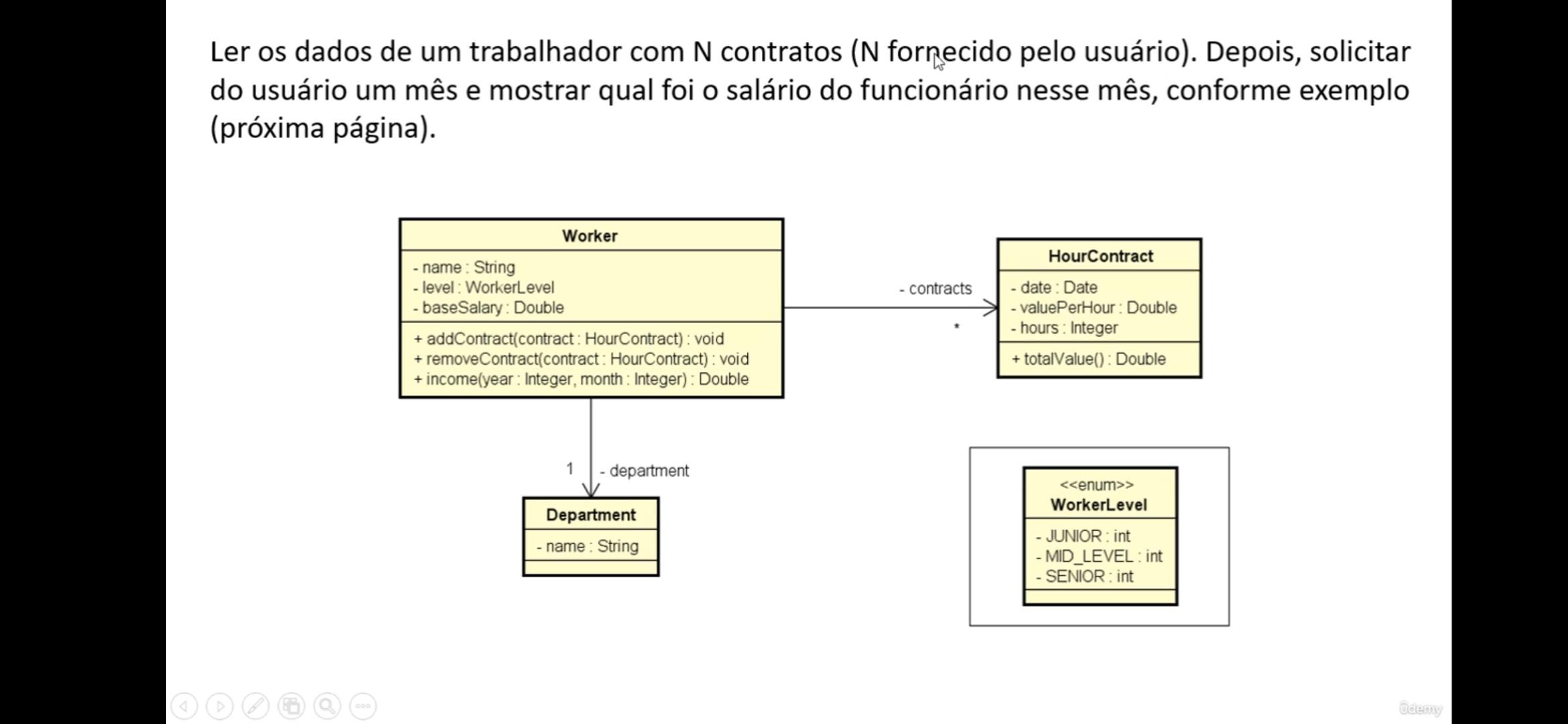
* Após o refinamento final do diagrama de classes, será iniciado o desenvolvimento das entidades no backend com base na modelagem definida.
* Planejamento de reunião de acompanhamento para revisar a implementação inicial e alinhar possíveis ajustes necessários.

**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
   1. Elaborar com base no modelo a seguir:



1. Tabela de referência de tipos primitivos entre C# e SQL Server
   1. Modelar os tipos primitivos de SQL Server para C#:
   2. 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). |

1. Script SQL Server utilizado
   1. 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)

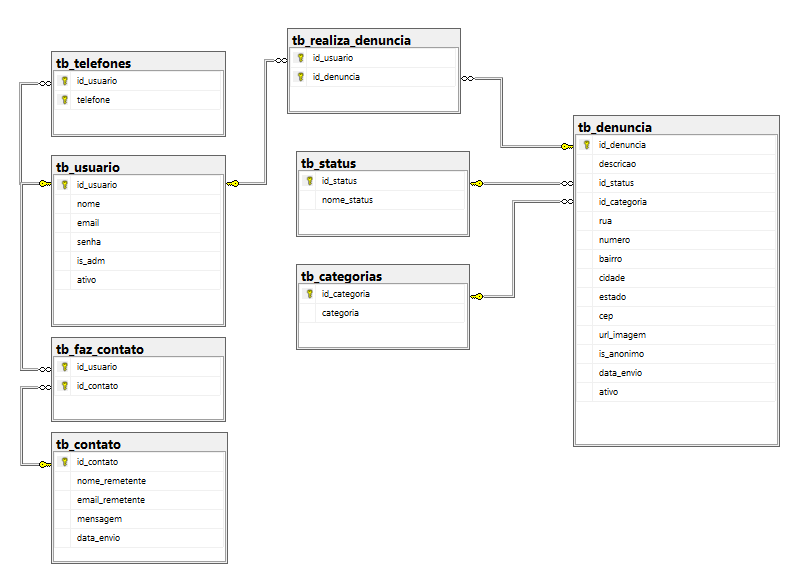
REFERENCES tb\_usuario (id\_usuario),

CONSTRAINT fk\_faz\_contato\_contato FOREIGN KEY (id\_contato)

REFERENCES tb\_contato (id\_contato)

);

1. Diagrama do Banco de Dados



**Resultado:**

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

1. Diagrama de Classes Elaborado

