SENAC

Campus Santo Amaro

TADS - Análise Desenvolvimento de Sistemas

*PW - Programa¸c˜ao Web*

Logotipo, nome da empresa

Descrição gerada automaticamente

**Atividade Discente**

**Orientada ADO #3**

Professor: Veríıssimo - [carlos.hvpereira@sp.senac.br](mailto:carlos.hvpereira@sp.senac.br)

Discentes: Alexandre William - [william20alex01@gmail.com](mailto:william20alex01@gmail.com)

Ian Saggioratto Pereira – [ian.saggior.pereira@gmail.com](mailto:ian.saggior.pereira@gmail.com)

20/11/2022

• Codigo da API: Deverá ter nesta seção todos os códigos (API

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* + /prisma/***schema.prisma***

// This is your Prisma schema file,

// learn more about it in the docs: https://pris.ly/d/prisma-schema

generator client {

  provider = "prisma-client-js"

}

datasource db {

  provider = "mysql"

  url      = env("DATABASE\_URL")

}

model User {

  idUser Int @id @default(autoincrement())

  nmUser String

  password String

  profile String

  created\_at DateTime? @default(now())

  updated\_at DateTime? @updatedAt

  user Comanda[] @relation

  @@map("users")

}

model Comanda {

  user User @relation(fields: [idUser], references: [idUser])

  idComanda Int @id @default(autoincrement())

  numeroMesa String

  status String

  draft String

  idUser Int

  comanda ItemComanda[] @relation("comanda")

  @@map("comanda")

}

model ItemComanda {

  comanda Comanda @relation("comanda", fields: [idComanda], references: [idComanda])

  idComanda Int

  produto Produto @relation("produto", fields: [idProduto], references: [idProduto])

  idProduto Int

  quantidade Int

  @@id([idComanda, idProduto])

  @@map("itemComanda")

}

model Produto {

  categoria Categoria @relation("categoria", fields: [idCategoria], references: [idCategoria])

  idProduto Int @id @default(autoincrement())

  idCategoria Int

  nmProduto String

  preco String

  descProduto String

  imgProduto String

  created\_at DateTime? @default(now())

  updated\_at DateTime? @updatedAt

  produto ItemComanda[] @relation("produto")

  @@map("produto")

}

model Categoria {

  idCategoria Int @id @default(autoincrement())

  nmCategoria String

  created\_at DateTime? @default(now())

  updated\_at DateTime? @updatedAt

  categoria Produto[] @relation("categoria")

  @@map("categoria")

}

* + /prisma/***migrations/*** ***migration\_lock.toml***

# Please do not edit this file manually

# It should be added in your version-control system (i.e. Git)

provider = "mysql"

* + /prisma/***migrations/*** ***20221117180210\_rendezvous\_tables/*** ***migration.sql***

-- CreateTable

CREATE TABLE `users` (

    `idUser` INTEGER NOT NULL AUTO\_INCREMENT,

    `nmUser` VARCHAR(191) NOT NULL,

    `password` VARCHAR(191) NOT NULL,

    `profile` VARCHAR(191) NOT NULL,

    `created\_at` DATETIME(3) NULL DEFAULT CURRENT\_TIMESTAMP(3),

    `updated\_at` DATETIME(3) NULL,

    PRIMARY KEY (`idUser`)

) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- CreateTable

CREATE TABLE `comanda` (

    `idComanda` INTEGER NOT NULL AUTO\_INCREMENT,

    `numeroMesa` VARCHAR(191) NOT NULL,

    `status` VARCHAR(191) NOT NULL,

    `draft` VARCHAR(191) NOT NULL,

    `idUser` INTEGER NOT NULL,

    PRIMARY KEY (`idComanda`)

) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- CreateTable

CREATE TABLE `itemComanda` (

    `idComanda` INTEGER NOT NULL,

    `idProduto` INTEGER NOT NULL,

    `quantidade` INTEGER NOT NULL,

    PRIMARY KEY (`idComanda`, `idProduto`)

) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- CreateTable

CREATE TABLE `produto` (

    `idProduto` INTEGER NOT NULL AUTO\_INCREMENT,

    `idCategoria` INTEGER NOT NULL,

    `nmProduto` VARCHAR(191) NOT NULL,

    `preco` VARCHAR(191) NOT NULL,

    `descProduto` VARCHAR(191) NOT NULL,

    `imgProduto` VARCHAR(191) NOT NULL,

    `created\_at` DATETIME(3) NULL DEFAULT CURRENT\_TIMESTAMP(3),

    `updated\_at` DATETIME(3) NULL,

    PRIMARY KEY (`idProduto`)

) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- CreateTable

CREATE TABLE `categoria` (

    `idCategoria` INTEGER NOT NULL AUTO\_INCREMENT,

    `nmCategoria` VARCHAR(191) NOT NULL,

    `created\_at` DATETIME(3) NULL DEFAULT CURRENT\_TIMESTAMP(3),

    `updated\_at` DATETIME(3) NULL,

    PRIMARY KEY (`idCategoria`)

) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- AddForeignKey

ALTER TABLE `comanda` ADD CONSTRAINT `comanda\_idUser\_fkey` FOREIGN KEY (`idUser`) REFERENCES `users`(`idUser`) ON DELETE RESTRICT ON UPDATE CASCADE;

-- AddForeignKey

ALTER TABLE `itemComanda` ADD CONSTRAINT `itemComanda\_idComanda\_fkey` FOREIGN KEY (`idComanda`) REFERENCES `comanda`(`idComanda`) ON DELETE RESTRICT ON UPDATE CASCADE;

-- AddForeignKey

ALTER TABLE `itemComanda` ADD CONSTRAINT `itemComanda\_idProduto\_fkey` FOREIGN KEY (`idProduto`) REFERENCES `produto`(`idProduto`) ON DELETE RESTRICT ON UPDATE CASCADE;

-- AddForeignKey

ALTER TABLE `produto` ADD CONSTRAINT `produto\_idCategoria\_fkey` FOREIGN KEY (`idCategoria`) REFERENCES `categoria`(`idCategoria`) ON DELETE RESTRICT ON UPDATE CASCADE;

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* + /src/***routes.ts***

import {Router, Request, Response} from 'express';

import { CreateUserController } from './controllers/CreateUserController';

import { ReadUserController } from './controllers/ReadUserController';

import cors from 'cors';

const router = Router();

router.use(cors())

router.get(('/users/all'), new ReadUserController().handle)

router.post('/users', new CreateUserController().handle)

export {router};

* + /src/***server.ts***

import express, {Request, Response, NextFunction} from 'express';

import {router} from './routes';

import 'express-async-errors';

const app = express();

app.use(express.json());

app.use(router)

app.use((err: Error, req: Request, res: Response, next: NextFunction) => {

    if (err instanceof Error) {

        return res.status(400).json({

            error: err.message

        })

    }

    return res.status(500).json({

        status: 'error',

        message: 'Internal server Error.'

    })

})

app.listen(3333, () => console.log("servidor online!"));

* + /src/***controllers/CreateUserController.ts***

import { Request, response, Response } from "express";

import { CreateUserService } from '../services/CreateUserService'

class CreateUserController {

    async handle(req: Request, res: Response) {

        try {

            const { nmUser, password, profile } = req.body;

            const createUserService = new CreateUserService();

            const user = await createUserService.execute({nmUser, password, profile});

            return res.json(user);

        } catch (error) {

            return res.status(400).send("" + error);

        }

    }

}

export { CreateUserController }

* + /src/***controllers/*** ***ReadUserController.ts***

import {Request, Response} from 'express';

import { ReadUserService } from '../services/ReadUserService';

export class ReadUserController {

    async handle(req: Request, res: Response) {

        const readUserService = new ReadUserService();

        const user = await readUserService.execute();

        res.header({

            Headers: {

                "Access-Control-Allow-Origin": "http://127.0.0.1:5500/index.html"

            }

        })

        return res.status(200).json(user);

    }

}

* + /src/***services/*** ***CreateUserService.ts***

import prismaClient from '../prisma'

interface UserRequest {

    nmUser: string;

    password: string;

    profile: string;

}

class CreateUserService {

    async execute({ nmUser, password, profile }: UserRequest) {

        if (!nmUser) {

            throw new Error("O nome do usuário precisa ser informado!!")

        }

        const userAlreadyExists = await prismaClient.user.findFirst({

            where: {

                nmUser: nmUser

            }

        })

        if (userAlreadyExists){

            throw new Error("Usuário informado já é existente!!")

        }

        const user = await prismaClient.user.create({

            data:{

                nmUser: nmUser,

                password: password,

                profile: profile

            }

        })

        return user;

    }

}

export { CreateUserService }

* + /src/***services/*** ***ReadUserService.ts***

import prismaClient from '../prisma';

export class ReadUserService {

    async execute() {

        const users = await prismaClient.user.findMany();

        return users;

    }

}

* + /src/***prisma/index.ts***

import { PrismaClient } from "@prisma/client";

const prismaClient = new PrismaClient();

export default prismaClient;

• Codigo WEB: Deverá ter nesta seção todos os códigos da págima WEB - HTML; Javascript; CSS

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* + /fetchAPI/index.css

body{

    font-family: cursive;

    background: linear-gradient(to right, rgb(10, 22, 35),rgb(0, 107, 156));

    color:#e8f1f2;

}

#corzinha{

    text-align: center;

    position:absolute;

    top: 10%;

    left: 34%;

    padding-bottom: 8px;

    background: linear-gradient(to right, rgb(32, 109, 143),rgb(27, 152, 224));

    font-size:larger;

    border-radius: 50px;

    width: 500px;

}

* + /fetchAPI/index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ADO4 - API</title>

    <script src="index.js"></script>

    <link href="/index.css" rel="stylesheet">

</head>

<body>

    <div id="corzinha">

        <h1 id="users">Lista com os usuários</h1>

        <button id="button" onclick="getUsers()">Fetch</button>

    </div>

</body>

</html>

* + /fetchAPI/index.js

const url = "http://localhost:3333/users/all";

function getUsers(){

    fetch(url)

    .then(response => {

        response.json().then(data => {

            let button = document.getElementById("button");

            button.remove();

            data.forEach(element => {

                let h2 = document.getElementById('users');

                let firstH2 = document.createElement("p");

                let secondH2 = document.createElement("p");

                firstH2.textContent = "Nome: " + element.nmUser;

                h2.appendChild(firstH2);

                secondH2.textContent = "Profile: " + element.profile;

                h2.appendChild(secondH2);

            })

        })

    })

}

* Evidência dos testes (Print screen dos resultados) – Lembrar também do print do Insomnia ou Postman

Pag Web:

Macaco do xandao tem q arrumar o código de viadão dps

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Tela de computador

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Insomnia:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente