Objetivos:

I. Desenvolver o FrontEnd da aplicação

i. Desenvolvendo o FrontEnd da aplicação

Passo 1 – O objetivo é criar o FrontEnd da aplicação da última aula. Verificar se está salvando os dados no pgAdmin4.

Vamos verificar os caminhos no arquivo **index.ts:**

```
import express from "express";
import dotenv from "dotenv";
import path from "path"; // para resolver caminhos
import { taskRoutes } from "./routes/taskRoutes";

dotenv.config();
const app = express();

app.use(express.urlencoded({ extended: true }));

// Middleware para ler JSON no corpo da requisição
app.use(express.json());

// Servir arquivos estáticos da pasta "views" (CSS, imagens, etc.)
app.use(express.static(path.join(__dirname, "views")));

// Rota para carregar o HTML principal
app.get("/", (req, res) => {
  res.sendFile(path.join(__dirname, "views", "index.html"));
});

// Rotas da API
app.use("/tasks", taskRoutes);

app.listen(3000, () => {
  console.log("Servidor rodando em http://localhost:3000");
});
```

Passo 2 – Vamos ajustar o nosso taskController.ts

src/controllers/taskController.ts

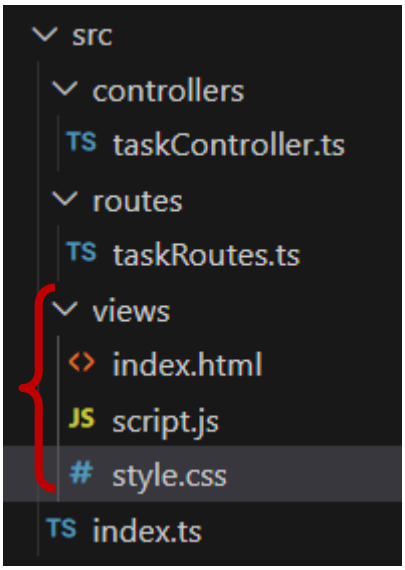
```
import { PrismaClient } from "@prisma/client";
import { Request, Response } from "express";

const prisma = new PrismaClient();

// 📌 Buscar todas as tarefas
export const getTasks = async (req: Request, res: Response) => {
  try {
    const tasks = await prisma.task.findMany();
    res.json(tasks);
  } catch (error) {
    res.status(500).json({ error: "Erro ao buscar tarefas" });
  }
};


// 📌 Criar uma nova tarefa
export const createTask = async (req: Request, res: Response) => {
  try {
    const { title, description } = req.body;

    if (!title || !description) {
      return res.status(400).json({ error: "Título e descrição são
obrigatórios" });
    }

    const newTask = await prisma.task.create({
      data: {
        title,
        description
      }
    });

    res.status(201).json(newTask);
  } catch (error) {
    res.status(500).json({ error: "Erro ao criar tarefa" });
  }
```

```
};

// 📌 Atualizar tarefa
export const updateTask = async (req: Request, res: Response) => {
  try {
    const { id } = req.params;
    const { title, description } = req.body;

    const updatedTask = await prisma.task.update({
      where: { id: Number(id) },
      data: { title, description }
    });

    res.json(updatedTask);
  } catch (error) {
    res.status(500).json({ error: "Erro ao atualizar tarefa" });
  }
};

// 📌 Deletar tarefa
export const deleteTask = async (req: Request, res: Response) => {
  try {
    const { id } = req.params;
    await prisma.task.delete({
      where: { id: Number(id) }
    });

    res.status(204).send();
  } catch (error) {
    res.status(500).json({ error: "Erro ao deletar tarefa" });
  }
};
```

Passo 3 – Criar a pasta views para realizar o nosso FrontEnd:

Para criar o nosso FrontEnd, teremos a seguinte estrutura:

Sobre as pastas:



- `index.html`: Exibir os campos de Título e Descrição; fazer o fetch para criar a tarefa; atualizar a lista automaticamente depois de criar;
- `style.css`: este arquivo define o estilo do nosso aplicativo;
- script.js: para realizar as atualizações das tarefas utilizando um ambiente agradável de edição.

Com isso, ao abrir http://localhost:3000, o HTML será carregado. O JavaScript vai buscar todas as tarefas no banco e exibir. Ao enviar o formulário, ele envia via fetch para o backend (POST /tasks). Será possível editar e excluir sem recarregar a página.

src/views/index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gerenciador de Tarefas</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>📋 Lista de Tarefas</h1>

  <form id="taskForm">
    <input type="text" id="title" placeholder="Título" required>
    <input type="text" id="description" placeholder="Descrição" required>
    <button type="submit">Adicionar</button>
  </form>

  <ul id="taskList"></ul>

  <script src="script.js"></script>
</body>
```

```
</html>
```

src/views/style.css

```
/* ======= Estilo Geral ======= */
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  max-width: 700px;
  margin: auto;
  padding: 20px;
  background: linear-gradient(135deg, #f0f4ff, #d9e6ff);
  color: #333;
}

h1 {
  text-align: center;
  color: #2b4eff;
  margin-bottom: 20px;
}

/* ======= Formulário ======= */
form {
  display: flex;
  gap: 10px;
  margin-bottom: 25px;
  background: white;
  padding: 15px;
  border-radius: 12px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.08);
}

input {
  flex: 1;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 8px;
  outline: none;
  font-size: 14px;
```

```
  transition: 0.3s;
}

input:focus {
  border-color: #2b4eff;
  box-shadow: 0 0 5px rgba(43, 78, 255, 0.4);
}

button {
  padding: 10px 15px;
  cursor: pointer;
  border: none;
  border-radius: 8px;
  background: #2b4eff;
  color: white;
  font-size: 14px;
  font-weight: bold;
  transition: 0.3s;
}

button:hover {
  background: #1d37cc;
  transform: scale(1.05);
}

/* ======= Lista de Tarefas ======= */
ul {
  list-style: none;
  padding: 0;
}

li {
  background: white;
  padding: 15px;
  margin-bottom: 12px;
  display: flex;
  justify-content: space-between;
  align-items: center;
```

```
  border-radius: 10px;
  box-shadow: 0 3px 8px rgba(0,0,0,0.05);
  transition: 0.3s;
}

li:hover {
  transform: scale(1.02);
}

/* ======= Ações ======= */
.actions button {
  margin-left: 5px;
  padding: 6px 10px;
  font-size: 12px;
  border-radius: 6px;
  background: #eaeaea;
  color: #333;
  border: none;
  transition: 0.3s;
}

.actions button:hover {
  background: #ccc;
}

.actions button:first-child {
  background: #ffc107;
  color: #fff;
}

.actions button:first-child:hover {
  background: #e0a800;
}

.actions button:last-child {
  background: #dc3545;
  color: #fff;
}
```

```
.actions button:last-child:hover {
  background: #b52a37;
}
```

src/views/script.js

```
const API_URL = "/tasks";
const taskForm = document.getElementById("taskForm");
const taskList = document.getElementById("taskList");

// 📌 Carregar tarefas ao iniciar
async function loadTasks() {
  const res = await fetch(API_URL);
  const tasks = await res.json();

  taskList.innerHTML = "";
  tasks.forEach(task => {
    const li = document.createElement("li");
    li.innerHTML = `
      <div>
        <strong>${task.title}</strong> - ${task.description}
      </div>
      <div class="actions">
        <button onclick="editTask(${task.id})">✏️ Editar</button>
        <button onclick="deleteTask(${task.id})">🗑️ Excluir</button>
      </div>
    `;
    taskList.appendChild(li);
  });
}

// 📌 Criar nova tarefa
taskForm.addEventListener("submit", async (e) => {
  e.preventDefault();

  const title = document.getElementById("title").value;
```

```
  const description = document.getElementById("description").value;

  await fetch(API_URL, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ title, description })
  });

  taskForm.reset();
  loadTasks();
});

// 📌 Excluir tarefa
async function deleteTask(id) {
  await fetch(`${API_URL}/${id}`, { method: "DELETE" });
  loadTasks();
}

// 📌 Editar tarefa
async function editTask(id) {
  const newTitle = prompt("Novo título:");
  const newDescription = prompt("Nova descrição:");

  if (newTitle && newDescription) {
    await fetch(`${API_URL}/${id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ title: newTitle, description: newDescription })
    });
    loadTasks();
  }
}

// Inicializa
loadTasks();
```

Passo 4 – Executar a aplicação através do `npm run dev`

- Abrir http://localhost:3000

- Cadastrar tarefas
- Verificar se o registro cadastrado está no pgAdmin 4