Front-end

Estrutura

exemplo/

```
├── main.py
├── models.py
├── schemas.py
├── database.py

├── templates/
    ├── index.html

├── static/
    ├── style.css
    ├── script.js
```

main.py

```python
from fastapi import FastAPI, status, Depends, Request
from schemas import Mensagem
import models
from database import engine, get_db
from sqlalchemy.orm import Session
from fastapi.responses import HTMLResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates

models.Base.metadata.create_all(bind=engine)

app = FastAPI()

app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="templates")

@app.get("/", response_class=HTMLResponse)
def home(request: Request, db: Session = Depends(get_db)):
    mensagens = db.query(models.Model_Mensagem).all()
    return templates.TemplateResponse("index.html",{"request": request, "mensagens": mensagens})

@app.post("/mensagens", status_code=status.HTTP_201_CREATED)
def criar_mensagem(nova_mensagem: Mensagem, db_session: Session = Depends(get_db)):
    mensagem_criada = models.Model_Mensagem(**nova_mensagem.model_dump())
    db_session.add(mensagem_criada)
    db_session.commit()
    db_session.refresh(mensagem_criada)
    return {"message": mensagem_criada}
```

main.py

```python
from fastapi import FastAPI, status, Depends, Request
from schemas import Mensagem
import models
from database import engine, get_db
from sqlalchemy.orm import Session
from fastapi.responses import HTMLResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates

models.Base.metadata.create_all(bind=engine)

app = FastAPI()

app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="templates")

@app.get("/", response_class=HTMLResponse)
def home(request: Request, db: Session = Depends(get_db)):
    mensagens = db.query(models.Model_Mensagem).all()
    return templates.TemplateResponse("index.html",{"request": request, "mensagens": mensagens})

@app.post("/mensagens", status_code=status.HTTP_201_CREATED)
def criar_mensagem(nova_mensagem: Mensagem, db_session: Session = Depends(get_db)):
    mensagem_criada = models.Model_Mensagem(**nova_mensagem.model_dump())
    db_session.add(mensagem_criada)
    db_session.commit()
    db_session.refresh(mensagem_criada)
    return {"message": mensagem_criada}
```

Por padrão, o FastAPI entende que toda rota retorna JSON. Mas, quando você quer devolver uma página HTML (como um arquivo index.html renderizado com o Jinja2), é necessário avisar isso explicitamente.

style.css

```css
body {
    font-family: Arial, sans-serif;
    margin: 20px;
}

input, textarea, button {
    font-size: 16px;
}

ul {
    list-style-type: none;
    padding-left: 0;
}
```

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Mensagens</title>
    <link rel="stylesheet" href="/static/style.css">
</head>
<body>
    <h2>Mensagens cadastradas:</h2>
    <ul id="listaMensagens">
        {% for msg in mensagens %}
            <li>
                <strong>{{ msg.titulo }}</strong>: {{ msg.conteudo }}
                {% if msg.publicada %}[Publicada]{% else %}[Não publicada]{%
endif %}
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

Browser

Mensagens cadastradas:

Primeira mensagem: Olá banco de dados! [Publicada]

Vamos criar um formulário para enviar uma mensagem!

Index.html (Adicionar em cima de <h2>Mensagens cadastradas:</h2>)

```
<h1>Enviar Mensagem</h1>

    <form id="mensagemForm">
        <label>Título:</label><br>
        <input type="text" name="titulo" required><br><br>

        <label>Conteúdo:</label><br>
        <textarea name="conteudo" rows="4" cols="50" required></textarea><br><br>

        <label>Publicada:</label>
        <input type="checkbox" name="publicada" checked><br><br>

        <button type="submit">Enviar</button>
    </form>

    <p id="mensagemSucesso" style="color: green;"></p>

    <hr>
```

Index.html (Adicionar no fim!)

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

script.js (Parte 1)

```
const form = document.getElementById("mensagemForm");
const mensagemSucesso = document.getElementById("mensagemSucesso");
const lista = document.getElementById("listaMensagens");


// Função para adicionar uma mensagem na lista HTML
function adicionarMensagemNaLista(msg) {
    const li = document.createElement("li");
    li.innerHTML = `<strong>${msg.titulo}</strong>: ${msg.conteudo} ${msg.publicada ?
'[Publicada]' : '[Não publicada]'}`;
    lista.appendChild(li);
}
```

- O FastAPI recebe a mensagem, salva no banco e devolve uma resposta JSON.
- Mas o HTML que está na tela não muda sozinho, porque o navegador não foi recarregado.
- O JavaScript precisa atualizar o DOM (a estrutura HTML da página) manualmente, é aí que entra a função adicionarMensagemNaLista().

script.js (Parte 2)

```
// Evento de envio do formulário
form.addEventListener("submit", async (e) => {
    e.preventDefault(); // evita recarregar a página


    // Captura os dados do formulário
    const formData = new FormData(form);
    const data = Object.fromEntries(formData.entries());
    data.publicada = formData.has("publicada"); // checkbox


    // Envia para a API via POST
    const response = await fetch("/mensagens", {
        method: "POST",
        headers: {"Content-Type": "application/json"},
        body: JSON.stringify(data)
    });


    if (response.ok) {
        // Apenas adiciona a nova mensagem na lista, sem limpar o restante
        adicionarMensagemNaLista(data);
        // Exibe mensagem de sucesso
        mensagemSucesso.textContent = `Mensagem '${data.titulo}' criada com sucesso!`;
        // Limpa o formulário
        form.reset();
    } else {
        mensagemSucesso.textContent = "Erro ao criar a mensagem!";
        mensagemSucesso.style.color = "red";
    }
});
```

Browser


Enviar Mensagem
Título:
Conteúdo:
Publicada:
Enviar
Mensagens cadastradas:
Primeira mensagem: Olá banco de dados! [Publicada]
Segunda mensagem: Olá front-end [Publicada]