## **Configuração Inicial e Deploy no Render**

### **Pré-requisitos**

**Conta Render:** Uma conta ativa no Render (Conectada ao Github)

**Configuração do Banco de Dados (PostgreSQL)**

Serviços criados manualmente.

Acesse o painel do Render e clique em **"New"** → **"PostgreSQL"**.

**Name:** financas-db-prod  
**Database:** financas\_db  
**User:** financas\_user  
**Plan:** Selecione **"Free"**.

Clique em **"Create Database"**.

Após a criação, vá para a aba **"Info"** e **COPIE o Internal Connection String (URL Interno)**. Guarde este link.

**Configuração e Deploy do Backend**

No Painel do Render, clique em **"New"** → **"Web Service"**.  
Conecte-se ao repositório GitHub e selecione o projeto.  
Configure os detalhes de deploy:

**Name:** financas-pessoais-backend   
**Root Directory:** backend  
**Environment:** Node  
**Branch:** Sua branch principal  
**Build Command:** npm install  
**Start Command:** npm start   
**Plan:** Selecione **"Free"**.

**Variáveis de Ambiente (Env Vars):** Adicione as variáveis:

**Key:** DATABASE\_URL

**Value:** Cole o Internal Connection String. **ADICIONE o sufixo:** ?schema=public ao final.  
**Key:** FRONTEND\_URL

**Value:** https://financas-pessoais-frontend.onrender.com

Clique em **"Create Web Service"**.

### **Deploy do Frontend (Static Site)**

O Frontend é simples, pois não requer variáveis de ambiente complexas.

No Painel do Render, clique em **"New"** → **"Static Site"**.

Conecte-se ao repositório GitHub e selecione o projeto.

Configure:

**Name:** financas-pessoais-frontend  
**Root Directory:** frontend  
**Build Command:** Deixe vazio.  
**Publish Directory:** . (Ponto).

Clique em **"Create Static Site"**.

## **Substituição da Backend URL no Projeto**

Após o deploy do Backend, o Render fornece a URL final do serviço (ex: https://financas-pessoais-backend-xyz.onrender.com). Devemos usar essa URL para atualizar o código do Frontend.

### **Obter a URL do Backend**

No Painel do Render, vá para o serviço **financas-pessoais-backend**.

**Copie a URL pública completa** (ex: https://financas-pessoais-backend-0dbj.onrender.com).

### **Substituir no script.js**

Abra o seu arquivo **frontend/script.js** localmente.

Localize a constante BACKEND\_URL no topo do arquivo:  
JavaScript  
const BACKEND\_URL = 'https://financas-pessoais-backend-0dbj.onrender.com';

**Substitua o valor** pela URL exata que você copiou do Render.  
Faça um **commit** e **push** dessa alteração para o GitHub.  
Vá ao Render e **force um novo Deploy** no serviço **financas-pessoais-frontend** para que ele use o script.js corrigido.

## **Forçar as Tabelas do PostgreSQL (Migração Manual)**

### **Pré-requisito: Cliente PSQL**

Instale o **cliente de linha de comando psql** na sua máquina

**Gere o Código SQL**

Abra o terminal na pasta **/backend** do seu projeto.

Execute o comando para gerar o código SQL de criação das tabelas  
  
npx prisma migrate diff --from-empty --to-schema-datamodel prisma/schema.prisma --script

**Copie TODO o código SQL** que aparecerá no terminal (começando com CREATE TABLE...).

### **Conectar ao Banco de Dados e Executar o SQL**

Usaremos o comando PSQL fornecido pelo Render para acesso externo.  
No Render, vá para o serviço de banco de dados (financas-db-prod).  
Copie o **PSQL Command** completo (ele contém o host, usuário e senha para conexão externa).  
No seu terminal (PowerShell ou CMD):

**Defina a senha** (A senha está no PSQL Command.  
$env:PGPASSWORD="[SUA SENHA DO RENDER AQUI]"

**Execute o Comando PSQL** (Substitua [CAMINHO\_PSQL] pelo caminho completo, como C:\Program Files\PostgreSQL\16\bin\psql.exe):  
  
& "[CAMINHO\_PSQL]\psql.exe" -h [HOST\_EXTERNO] -U [USUÁRIO] [NOME\_DB]

*Exemplo:*  
& "[C:\Program Files\PostgreSQL\16\bin\psql.exe]" -h dpg-d3cac737mgec73abc00g-a.oregon-postgres.render.com -U financas\_user financas\_db

Ao ser conectado (financas\_db=>):  
**Cole o código SQL** que foi gerado.  
Pressione **ENTER** para executar.  
Você verá as mensagens CREATE TABLE e ALTER TABLE para todas as entidades.

Digite \q e ENTER para sair.

Vale ressaltar que precisa usar a versão 17 do postg.

Caso perguntem porque não foi criado normalmente no render as tabelas como deveria ser através do schema.prisma, foi pq o serviço não carregava e por isso não conseguia criar as tabelas de forma automática. A solução foi forçar a criação das tabelas por um acesso externo com senha para que o banco fosse carregado.

Poderia ter usado o próprio arquivo render para criar tudo de forma automática, mas pelo blueprint ele solicitava uma conta paga para isso. Por isso, cada parte foi criada manualmente.