Manual de Implantação

Glossário Técnico em Libras — Modelagem do Vestuário

Versão do sistema: 3.0.8

Data de atualização: Maio de 2026

Pesquisadora: Eline Santos Silva

Produto Educacional: Mestrado ProfEPT/IFC Blumenau

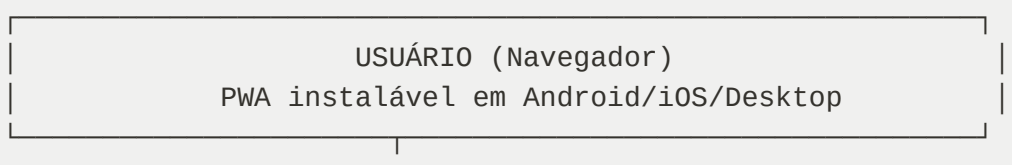
Sumário

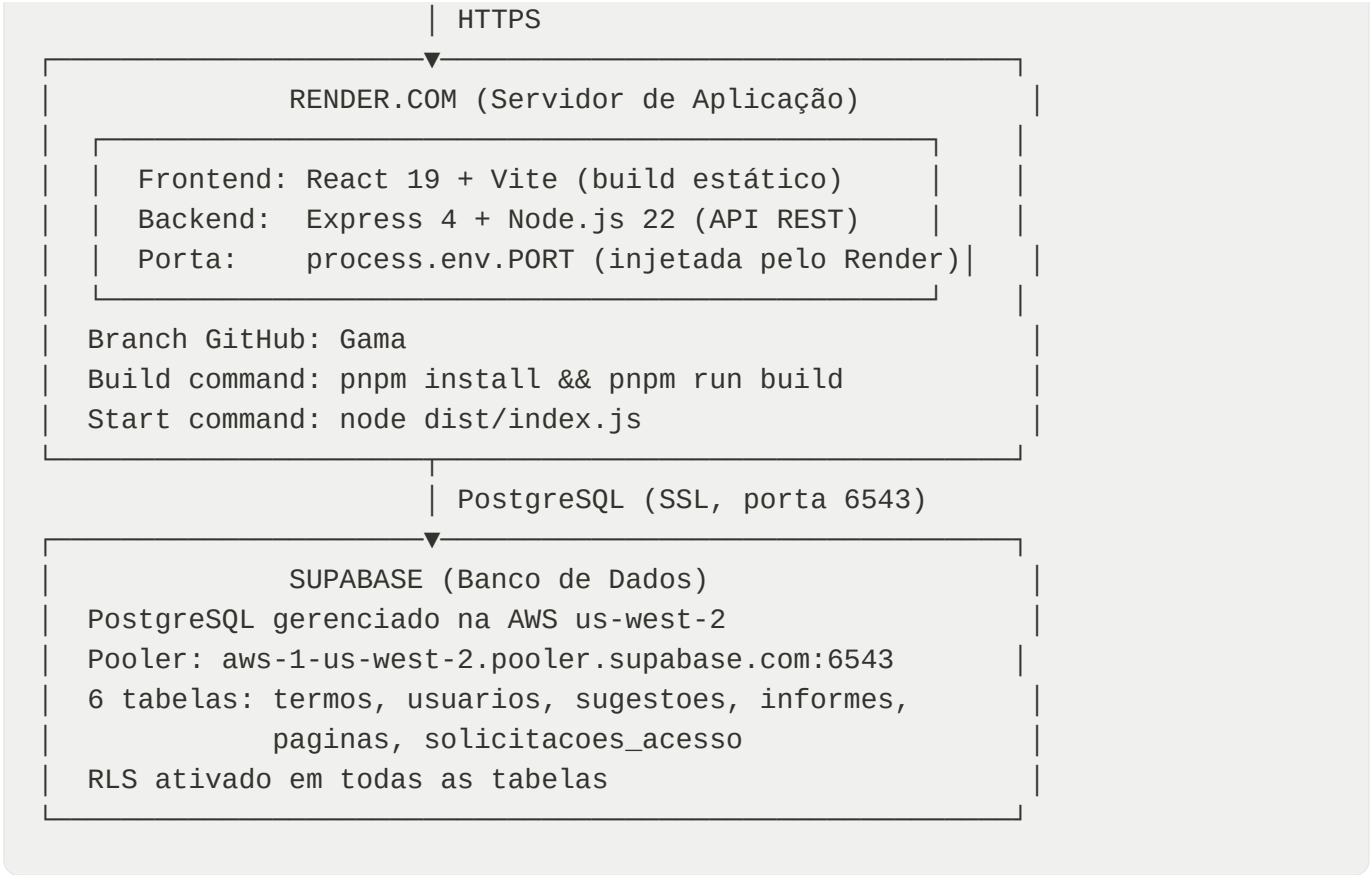
---

1. Visão Geral da Arquitetura

O sistema é uma aplicação web full-stack composta por três camadas independentes que se comunicam via HTTPS:

Plain Text





Tecnologias utilizadas:

Camada	Tecnologia	Versão
Frontend	React	19.x
Estilização	Tailwind CSS	4.x
Bundler	Vite	7.x
Roteamento	Wouter	3.x
Componentes UI	shadcn/ui + Radix UI	—
Backend	Express	4.x
Runtime	Node.js	22.x
Gerenciador de pacotes	pnpm	10.x
Banco de dados	PostgreSQL (Supabase)	15.x
Autenticação	JWT (jsonwebtoken) + bcryptjs	—
PWA	vite-plugin-pwa + Workbox	—

Linguagem	TypeScript	5.9.x

---

2. Pré-requisitos

2.1 Contas necessárias

Antes de iniciar a implantação, é necessário ter acesso às seguintes plataformas:

Plataforma	Finalidade	URL
GitHub	Repositório do código-fonte	https://github.com
Supabase	Banco de dados PostgreSQL gerenciado	https://supabase.com
Render.com	Hospedagem da aplicação (servidor + frontend)	https://render.com

Todas as plataformas oferecem plano gratuito suficiente para o uso educacional deste projeto.

2.2 Ferramentas locais (para desenvolvimento)

Para executar o projeto localmente ou realizar manutenção no código-fonte, são necessários:

- **Node.js 22.x** ou superior — disponível em https://nodejs.org
- **pnpm 10.x** — instalação: `npm install -g pnpm`
- **Git** — disponível em https://git-scm.com
- Editor de código recomendado: **Visual Studio Code** com extensões TypeScript e ESLint

2.3 Acesso ao repositório

O código-fonte está hospedado em:

Plain Text

```
https://github.com/Hedeson/glossario-libras
Branch de produção: Gama
```

Para clonar o repositório:

```bash
git clone https://github.com/Hedeson/glossario-libras.git
cd glossario-libras
git checkout Gama
```

---

3. Configuração do Banco de Dados (Supabase)

3.1 Criação do projeto no Supabase

1. Acesse https://supabase.com e faça login (ou crie uma conta gratuita).
2. Clique em **New Project**.
3. Preencha:
   - **Name:** `glossario-libras` (ou outro nome de sua preferência)
   - **Database Password:** escolha uma senha forte e **anote-a** — será necessária depois
   - **Region:** escolha a região mais próxima (ex.: `South America (São Paulo)` se disponível, ou `US West`)
4. Aguarde a criação do projeto (pode levar 1–2 minutos).

3.2 Obtenção da string de conexão

> **Atenção:** Use sempre a URL do **Transaction Pooler** (porta 6543), não a conexão direta (porta 5432). A conexão direta não funciona em ambientes serverless como o Render.

1. No painel do Supabase, acesse **Project Settings → Database**.
2. Role até a seção **Connection string**.
3. Selecione a aba **URI**.
4. Copie a URL que começa com `postgresql://postgres.[ID]:[SENHA]@aws-1-us-west-2.pooler.supabase.com:6543/postgres` .
5. Substitua `[YOUR-PASSWORD]` pela senha definida na criação do projeto.

A URL final terá o formato:

```
postgresql://postgres.qlaeiwytpbnrrouegpqn:SENHA@aws-1-us-west-2.pooler.supabas
```

3.3 Criação das tabelas

Acesse o **SQL Editor** no painel do Supabase e execute os seguintes scripts na ordem indicada:

Tabela `termos`

SQL

```sql
CREATE TABLE IF NOT EXISTS termos (
  id SERIAL PRIMARY KEY,
  termo VARCHAR(255) NOT NULL,
  significado TEXT NOT NULL,
  sinal_libras TEXT,
  configuracao_sinal TEXT,
  status VARCHAR(50) DEFAULT 'aprovado',
  curso VARCHAR(255),
  curso_sigla VARCHAR(50),
  imagem_url TEXT,
  imagem_alt TEXT,
  imagem_legenda TEXT,
  referencias TEXT[] DEFAULT '{}',
  observacoes TEXT,
  ativo BOOLEAN DEFAULT true,
  criado_por VARCHAR(255),
  explicacao_detalhada TEXT,
  exemplo_uso TEXT,
  explicacao_expandida TEXT,
  termos_relacionados TEXT[] DEFAULT '{}',
  data_adicao TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  ultima_edicao TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Tabela `usuarios`

SQL

```sql
CREATE TABLE IF NOT EXISTS usuarios (
  id SERIAL PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  senha_hash VARCHAR(255) NOT NULL,
  perfil VARCHAR(50) DEFAULT 'visualizador',
  instituicao VARCHAR(255),
  ativo BOOLEAN DEFAULT true,
```

```
  criado_em TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Tabela `sugestoes`

SQL

```
CREATE TABLE IF NOT EXISTS sugestoes (
  id SERIAL PRIMARY KEY,
  nome VARCHAR(255),
  email VARCHAR(255),
  termo VARCHAR(255) NOT NULL,
  video_url TEXT,
  descricao TEXT,
  instituicao VARCHAR(255),
  status VARCHAR(50) DEFAULT 'pendente',
  criado_em TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Tabela `informes`

SQL

```
CREATE TABLE IF NOT EXISTS informes (
  id SERIAL PRIMARY KEY,
  titulo VARCHAR(255) NOT NULL,
  conteudo TEXT NOT NULL,
  ativo BOOLEAN DEFAULT false,
  tipo VARCHAR(50) DEFAULT 'info',
  criado_em TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  atualizado_em TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Tabela `paginas`

SQL

```
CREATE TABLE IF NOT EXISTS paginas (
  id SERIAL PRIMARY KEY,
  slug VARCHAR(100) UNIQUE NOT NULL,
  titulo VARCHAR(255),
  conteudo TEXT,
  atualizado_em TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

```sql
-- Inserir conteúdo padrão das páginas
INSERT INTO paginas (slug, titulo, conteudo) VALUES
  ('sobre', 'Sobre o Projeto', '<p>Glossário Técnico em Libras desenvolvido com
  ('creditos', 'Créditos', '<p>Pesquisadora: Eline Santos Silva</p>')
ON CONFLICT (slug) DO NOTHING;
```

Tabela `solicitacoes_acesso`

SQL

```sql
CREATE TABLE IF NOT EXISTS solicitacoes_acesso (
  id SERIAL PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  instituicao VARCHAR(255),
  perfil VARCHAR(50) DEFAULT 'colaborador',
  justificativa TEXT,
  status VARCHAR(50) DEFAULT 'pendente',
  criado_em TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

3.4 Ativação do Row-Level Security (RLS)

O RLS é uma camada de segurança que impede acesso direto não autorizado às tabelas. Execute:

SQL

```sql
-- Ativar RLS em todas as tabelas
ALTER TABLE termos ENABLE ROW LEVEL SECURITY;
ALTER TABLE usuarios ENABLE ROW LEVEL SECURITY;
ALTER TABLE sugestoes ENABLE ROW LEVEL SECURITY;
ALTER TABLE informes ENABLE ROW LEVEL SECURITY;
ALTER TABLE paginas ENABLE ROW LEVEL SECURITY;
ALTER TABLE solicitacoes_acesso ENABLE ROW LEVEL SECURITY;

-- Políticas de leitura pública (termos, informes, páginas)
CREATE POLICY "Leitura pública de termos" ON termos FOR SELECT USING (true);
CREATE POLICY "Leitura pública de informes" ON informes FOR SELECT USING (true)
CREATE POLICY "Leitura pública de páginas" ON paginas FOR SELECT USING (true);

-- Políticas de escrita pelo service role (backend)
CREATE POLICY "Escrita service role termos" ON termos FOR ALL USING (auth.role(
CREATE POLICY "Escrita service role usuarios" ON usuarios FOR ALL USING (auth.r
CREATE POLICY "Escrita service role sugestoes" ON sugestoes FOR ALL USING (auth
```

```
CREATE POLICY "Escrita service role informes" ON informes FOR ALL USING (auth.r
CREATE POLICY "Escrita service role paginas" ON paginas FOR ALL USING (auth.rol
CREATE POLICY "Escrita service role solicitacoes" ON solicitacoes_acesso FOR AL
```

> **Nota:** O backend usa a `DATABASE_URL` com usuário `postgres` que tem permissão de `service_role`. As políticas acima garantem que apenas o backend autenticado possa escrever nos dados.

3.5 Criação dos usuários iniciais

Execute o script abaixo para criar os três usuários padrão do sistema. As senhas já estão em formato bcrypt (hash):

SQL

```sql
-- Usuário Administrador
INSERT INTO usuarios (nome, email, senha_hash, perfil, ativo) VALUES (
  'Administrador',
  'admin@glossario',
  '$2b$10$HASH_BCRYPT_AQUI',
  'administrador',
  true
);

-- Usuário Colaborador
INSERT INTO usuarios (nome, email, senha_hash, perfil, ativo) VALUES (
  'Colaborador',
  'colaborador@glossario',
  '$2b$10$HASH_BCRYPT_AQUI',
  'colaborador',
  true
);

-- Usuário Revisor
INSERT INTO usuarios (nome, email, senha_hash, perfil, ativo) VALUES (
  'Revisor',
  'revisor@glossario',
  '$2b$10$HASH_BCRYPT_AQUI',
  'revisor',
  true
);
```

> **Importante:** Para gerar os hashes bcrypt corretos das senhas, utilize o script Node.js a seguir (execute localmente após instalar as dependências):

JavaScript

```
// gerar_senhas.mjs
import bcrypt from 'bcryptjs';
const senhas = [
  { email: 'admin@glossario', senha: 'Admin@Glossario2025!' },
  { email: 'colaborador@glossario', senha: 'Colab@Libras2025!' },
  { email: 'revisor@glossario', senha: 'Revisor@Glossario2025!' },
];
for (const u of senhas) {
  const hash = await bcrypt.hash(u.senha, 10);
  console.log(`${u.email}: ${hash}`);
}
```

Execute com: `node gerar_senhas.mjs`

4. Configuração do Servidor de Aplicação (Render.com)

4.1 Criação do serviço no Render

1. Acesse https://render.com e faça login.
2. Clique em **New → Web Service**.
3. Conecte sua conta GitHub e selecione o repositório `Hedeson/glossario-libras`.
4. Preencha as configurações:

Campo	Valor
Name	`glossario-libras`
Branch	`Gama`
Runtime	`Node`
Build Command	`pnpm install --frozen-lockfile && pnpm run build`
Start Command	`node dist/index.js`
Instance Type	Free (ou superior para melhor desempenho)

1. Clique em **Advanced** e adicione as variáveis de ambiente (ver seção 6).
2. Clique em **Create Web Service**.

4.2 Configuração do auto-deploy

O Render detecta automaticamente novos commits no branch `Gama` e inicia um novo deploy. Para desativar o auto-deploy (deploy apenas manual), acesse **Settings → Auto-Deploy** e desative a opção.

4.3 URL da aplicação

Após o primeiro deploy bem-sucedido, a aplicação estará disponível em:

Plain Text

```
https://glossario-libras.onrender.com
```

> **Atenção:** No plano gratuito do Render, o serviço hiberna após 15 minutos de inatividade. O primeiro acesso após a hibernação pode levar até 60 segundos para responder. Para evitar isso, considere usar um serviço de ping periódico (ex.: UptimeRobot) ou fazer upgrade para o plano pago.

---

5. Configuração do Repositório GitHub

5.1 Fork ou clone do repositório

Se precisar criar uma nova cópia do projeto:

Bash

```bash
# Clonar o repositório original
git clone https://github.com/Hedeson/glossario-libras.git
cd glossario-libras
git checkout Gama

# Instalar dependências
pnpm install

# Executar em modo de desenvolvimento (local )
pnpm run dev
```

5.2 Estrutura de branches

Branch	Finalidade
`Gama`	**Produção** — branch conectado ao Render.com

`main`	Desenvolvimento / histórico anterior

> **Regra:** Sempre faça push para o branch `Gama` para acionar o deploy no Render.

5.3 Processo de atualização do código

Para atualizar o sistema em produção:

Bash

```
# 1. Faça as alterações no código
# 2. Teste localmente com pnpm run dev
# 3. Commit e push para o branch Gama
git add -A
git commit -m "feat: descrição da alteração"
git push origin Gama
# O Render iniciará o deploy automaticamente
```

---

6. Variáveis de Ambiente

As variáveis de ambiente devem ser configuradas no painel do Render em **Environment → Environment Variables**. **Nunca** as inclua no código-fonte ou em arquivos `.env` commitados.

Variável	Obrigatória	Descrição	Exemplo
`DATABASE_URL`	**Sim**	URL de conexão com o Supabase (pooler, porta 6543)	`postgresql://postgres.ID:SENHA@aws-1-us-west-2.pooler.supabase.com:6543/postgres?sslmode=require`
`JWT_SECRET`	**Sim**	Chave secreta para assinar os tokens JWT. Use uma string longa e aleatória	`minha-chave-secreta-muito-longa-e-aleatoria-2026`
`NODE_ENV`	**Sim**	Modo de execução	`production`
`PORT`	Não	Porta do servidor (injetada automaticamente pelo Render)	`3000`

6.1 Como gerar um JWT_SECRET seguro

Execute no terminal:

```bash
node -e "console.log(require('crypto').randomBytes(64).toString('hex'))"
```

Copie a saída (128 caracteres hexadecimais) e use como valor de `JWT_SECRET`.

6.2 Formato correto da DATABASE_URL

A URL deve obrigatoriamente:

- Usar o host do **pooler** (`pooler.supabase.com`), não o host direto
- Usar a **porta 6543** (Transaction Pooler), não 5432
- Incluir `?sslmode=require` ao final

Formato correto:

```
postgresql://postgres.SEU_PROJECT_ID:SUA_SENHA@aws-1-us-west-2.pooler.supabase.
```

Formato incorreto (não usar):

```
postgresql://postgres:SUA_SENHA@db.SEU_PROJECT_ID.supabase.co:5432/postgres
```

---

7. Estrutura do Banco de Dados

7.1 Diagrama de tabelas

```
termos
├── id (PK, SERIAL)
├── termo (VARCHAR 255, NOT NULL)
├── significado (TEXT, NOT NULL)
├── sinal_libras (TEXT) — URL do vídeo no YouTube
├── configuracao_sinal (TEXT)
```

```
├── status (VARCHAR 50) — 'aprovado'|'em_revisao'|'temporario'|'em_certificacao
├── curso (VARCHAR 255)
├── curso_sigla (VARCHAR 50)
├── imagem_url (TEXT)
├── imagem_alt (TEXT)
├── imagem_legenda (TEXT)
├── referencias (TEXT[])
├── observacoes (TEXT)
├── ativo (BOOLEAN, DEFAULT true)
├── criado_por (VARCHAR 255)
├── explicacao_detalhada (TEXT)
├── exemplo_uso (TEXT)
├── explicacao_expandida (TEXT)
├── termos_relacionados (TEXT[])
├── data_adicao (TIMESTAMPTZ, DEFAULT NOW())
└── ultima_edicao (TIMESTAMPTZ, DEFAULT NOW())

usuarios
├── id (PK, SERIAL)
├── nome (VARCHAR 255, NOT NULL)
├── email (VARCHAR 255, UNIQUE, NOT NULL)
├── senha_hash (VARCHAR 255, NOT NULL) — bcrypt hash, custo 10
├── perfil (VARCHAR 50) — 'administrador'|'colaborador'|'revisor'|'visualizador
├── instituicao (VARCHAR 255)
├── ativo (BOOLEAN, DEFAULT true)
└── criado_em (TIMESTAMPTZ, DEFAULT NOW())
```

7.2 Status dos termos

Status	Significado	Visível no glossário público
`aprovado`	Termo revisado e validado	Sim
`em_revisao`	Aguardando revisão da equipe	Sim (com indicador)
`temporario`	Sinal provisório, não validado	Sim (com indicador)
`em_certificacao`	Em processo de certificação pela comunidade surda	Sim (com indicador)
`pendente`	Sem sinal definido	Sim (sem link de Libras)

---

8. Usuários e Perfis de Acesso

8.1 Perfis disponíveis

Perfil	Criar termos	Editar termos	Excluir termos	Aprovar revisões	Gerenciar usuários
`administrador`	Sim	Sim	Sim	Sim	Sim
`colaborador`	Sim	Sim	Não	Não	Não
`revisor`	Não	Não	Não	Sim	Não
`visualizador`	Não	Não	Não	Não	Não

8.2 Credenciais padrão

> **Atenção:** Altere as senhas padrão após a primeira implantação em produção.

E-mail	Senha padrão	Perfil
`admin@glossario`	`Admin@Glossario2025!`	Administrador
`colaborador@glossario`	`Colab@Libras2025!`	Colaborador
`revisor@glossario`	`Revisor@Glossario2025!`	Revisor

8.3 Como alterar senhas

Para alterar a senha de um usuário, execute no SQL Editor do Supabase:

SQL

```sql
-- Substitua 'NOVO_HASH' pelo hash bcrypt da nova senha
UPDATE usuarios
SET senha_hash = 'NOVO_HASH'
WHERE email = 'admin@glossario';
```

Para gerar o hash bcrypt da nova senha:

JavaScript

```javascript
// Node.js (execute localmente)
import bcrypt from 'bcryptjs';
```

```
const hash = await bcrypt.hash('NovaSenha@2026!', 10);
console.log(hash);
```

8.4 Como criar novos usuários

Via painel Admin (recomendado):

1. Faça login com o perfil `administrador` em `/admin`
2. Acesse **Gerenciar Usuários**
3. Clique em **Novo Usuário** e preencha o formulário

Via SQL (alternativa):

```
SQL

INSERT INTO usuarios (nome, email, senha_hash, perfil, ativo)
VALUES ('Nome Completo', 'email@instituicao.edu.br', '$2b$10$HASH', 'colaborado
```

---

9. Processo de Deploy Completo (Passo a Passo)

Esta seção descreve o processo completo de implantação do zero, do início ao fim.

Etapa 1 — Configurar o Supabase

1. Crie o projeto no Supabase (seção 3.1).
2. Copie a URL do pooler (seção 3.2).
3. Execute os scripts SQL de criação das tabelas (seção 3.3).
4. Ative o RLS (seção 3.4).
5. Crie os usuários iniciais (seção 3.5).
6. Importe os termos do glossário (ver seção 10.1).

Etapa 2 — Configurar o GitHub

1. Faça fork do repositório ou clone-o para sua conta.
2. Certifique-se de que o branch `Gama` existe e está atualizado.

Etapa 3 — Configurar o Render

1. Crie o Web Service no Render (seção 4.1).
2. Adicione as variáveis de ambiente (seção 6):

- `DATABASE_URL` — URL do pooler do Supabase
- `JWT_SECRET` — chave secreta gerada aleatoriamente
- `NODE_ENV` — `production`

3. Aguarde o build e deploy (5–10 minutos no primeiro deploy).
4. Acesse a URL gerada pelo Render para verificar se o sistema está funcionando.

Etapa 4 — Verificação pós-deploy

Após o deploy, verifique os seguintes pontos:

1. **Página inicial carrega** — acesse `https://SEU-DOMINIO.onrender.com`
2. **Glossário exibe termos** — os termos devem aparecer na tela inicial
3. **Login funciona** — acesse `/admin` e faça login com `admin@glossario`
4. **Indicador de banco está verde** — no rodapé, o indicador deve mostrar "Banco de dados conectado"
5. **API de saúde responde** — acesse `https://SEU-DOMINIO.onrender.com/api/health` e verifique:

```JSON
{ "status": "ok", "version": "3.0.8", "database": "connected" }
```

---

10. Manutenção e Operações Comuns

10.1 Importar o glossário em lote

Para importar todos os termos de uma vez a partir do arquivo `server/data/glossary.json`:

1. Faça login no painel Admin com o perfil `administrador`.
2. Acesse a aba **Gestão de Termos**.
3. Clique em **Importar Glossário** (ícone de upload).
4. Selecione o arquivo `glossary.json` do seu computador.
5. Confirme a importação.

Alternativamente, via API REST (requer token JWT):

```Bash
curl -X POST https://SEU-DOMINIO.onrender.com/api/admin/glossary/import \
  -H "Authorization: Bearer SEU_TOKEN_JWT" \
```

```
  -H "Content-Type: application/json" \
  -d @server/data/glossary.json
```

10.2 Exportar o glossário

No painel Admin, clique em **Exportar JSON** para baixar todos os termos em formato JSON. O arquivo pode ser usado como backup ou para reimportação.

10.3 Adicionar imagens dos termos

As imagens dos termos seguem a nomenclatura `[Termo]_esc.jpg` (ex.: `Alinhavo_esc.jpg`). Para adicionar:

1. Gere a imagem seguindo o prompt em `docs/prompt_geracao_imagens.txt`.
2. Nomeie o arquivo conforme a convenção: `[Termo]_esc.jpg`.
3. Faça upload para `client/public/images/`.
4. No painel Admin, edite o termo e preencha o campo **URL da Imagem** com `/images/[Termo]_esc.jpg`.

10.4 Atualizar o conteúdo das páginas Sobre e Créditos

1. Faça login como `administrador`.
2. Acesse **Editar Conteúdo** no painel Admin.
3. Edite o conteúdo usando o editor HTML rico.
4. Clique em **Salvar**.

10.5 Gerenciar informes (avisos pop-up)

Os informes são mensagens exibidas em pop-up na página inicial. Para gerenciar:

1. Acesse **Informes** no painel Admin.
2. Crie, edite ou ative/desative informes conforme necessário.
3. Apenas um informe pode estar ativo por vez.

10.6 Backup do banco de dados

O Supabase realiza backups automáticos diários no plano gratuito (retenção de 7 dias). Para backup manual:

1. No painel do Supabase, acesse **Database → Backups**.
2. Clique em **Download** para baixar o backup mais recente.

Alternativamente, exporte os dados via SQL Editor:

```sql
-- Exportar todos os termos ativos
SELECT * FROM termos WHERE ativo = true ORDER BY termo;
```

---

11. Problemas Conhecidos e Soluções

11.1 Erro de SSL na conexão com o banco

Sintoma: Logs do servidor mostram `Error: There was an error establishing an SSL connection`.

Causa mais comum: A `DATABASE_URL` está usando a conexão direta (porta 5432) em vez do pooler (porta 6543), ou está faltando `?sslmode=require`.

Solução:

1. No Render, acesse **Environment → Environment Variables**.
2. Edite a variável `DATABASE_URL`.
3. Certifique-se de que a URL usa o host `pooler.supabase.com` e a porta `6543`.
4. Adicione `?sslmode=require` ao final da URL se não estiver presente.
5. Salve e aguarde o redeploy automático.

> **Nota:** Este erro é **esperado** no ambiente de desenvolvimento local da plataforma Manus, pois o `DATABASE_URL` local aponta para um banco MySQL interno. Não é um bug — o sistema funciona corretamente no Render com o Supabase.

11.2 Login retorna "Credenciais inválidas"

Sintoma: Ao tentar fazer login no painel Admin, a mensagem "Credenciais inválidas" é exibida mesmo com a senha correta.

Causas possíveis e soluções:

Causa	Solução
Hash bcrypt desatualizado no banco	Regenere o hash com `bcrypt.hash('senha', 10)` e atualize no Supabase via SQL
Usuário marcado como `ativo = false`	Execute `UPDATE usuarios SET ativo = true WHERE email = 'email@exemplo'`

Banco não conectado (DATABASE_URL errada)	Verifique o indicador de banco no rodapé — se estiver âmbar, corrija a DATABASE_URL
Tabela `usuarios` vazia	Execute o script de criação dos usuários iniciais (seção 3.5)

11.3 Glossário não carrega termos (tela em branco ou "0 termos")

Sintoma: A página inicial carrega, mas não exibe nenhum termo.

Causas possíveis e soluções:

Causa	Solução
Tabela `termos` vazia	Importe o glossário via painel Admin (seção 10.1)
Todos os termos com `ativo = false`	Execute `UPDATE termos SET ativo = true` no SQL Editor
Banco desconectado	Verifique e corrija a `DATABASE_URL` no Render
Erro de CORS	Verifique os logs do Render — o servidor deve estar rodando na mesma origem

11.4 Deploy falha no Render com erro de build

Sintoma: O build falha com erros de TypeScript ou dependências.

Solução:

1. Verifique os logs de build no Render (clique no deploy com erro → **Logs**).
2. Se o erro for `ERR_PNPM_OUTDATED_LOCKFILE`, execute localmente:

   Bash

   ```
   pnpm install --no-frozen-lockfile
   git add pnpm-lock.yaml
   git commit -m "fix: regenerar pnpm-lock.yaml"
   git push origin Gama
   ```

3. Se o erro for de TypeScript, execute `pnpm run check` localmente para identificar o problema.

11.5 PWA não instala ou não atualiza

Sintoma: O botão "Instalar App" não aparece, ou a versão instalada está desatualizada.

Solução:

1. Verifique se o arquivo `manifest.webmanifest` está sendo servido corretamente (acesse `https://SEU-DOMINIO.onrender.com/manifest.webmanifest`).
2. Para forçar atualização do Service Worker, acesse o site no navegador, abra as Ferramentas do Desenvolvedor (F12) → **Application → Service Workers** → clique em **Update**.
3. Se o ícone do app estiver faltando, verifique se os arquivos em `client/public/icons/` estão presentes no repositório.

11.6 Página inicial lenta no primeiro acesso

Sintoma: O primeiro acesso ao site demora 30–60 segundos para responder.

Causa: O plano gratuito do Render hiberna o serviço após 15 minutos de inatividade. O primeiro acesso acorda o serviço.

Soluções:

- Configure um serviço de monitoramento gratuito como UptimeRobot para fazer ping no endpoint `/api/health` a cada 14 minutos, mantendo o serviço ativo.
- Faça upgrade para o plano pago do Render (a partir de US$ 7/mês), que não hiberna.

11.7 Indicador de banco mostra "Modo offline" no Render

Sintoma: O rodapé mostra o indicador âmbar "Modo offline (dados locais)" mesmo no Render.

Causa: A `DATABASE_URL` está incorreta ou o Supabase está temporariamente indisponível.

Solução:

1. Acesse `https://SEU-DOMINIO.onrender.com/api/health` e verifique o campo `database`.
2. Se retornar `"error"`, verifique a `DATABASE_URL` no painel do Render.
3. Verifique o status do Supabase em https://status.supabase.com.

11.8 Erro "Cannot find module" ao iniciar o servidor

Sintoma: O servidor não inicia com erro `Cannot find module './routes/...'`.

Causa: O build não foi executado corretamente ou o diretório `dist/` está incompleto.

Solução:

1. No Render, acesse **Manual Deploy → Clear build cache & deploy**.
2. Aguarde o build completo.

11.9 RLS bloqueando operações do backend

Sintoma: Operações de escrita (criar/editar/excluir termos) retornam erro 403 ou não persistem.

Causa: As políticas RLS estão bloqueando o acesso do backend.

Solução: Execute no SQL Editor do Supabase:

```sql
-- Verificar políticas existentes
SELECT tablename, policyname, cmd FROM pg_policies WHERE schemaname = 'public';

-- Se necessário, recriar as políticas de escrita
DROP POLICY IF EXISTS "Escrita service role termos" ON termos;
CREATE POLICY "Escrita service role termos" ON termos FOR ALL USING (true);
```

> **Nota:** A política `USING (true)` permite acesso total. Isso é seguro porque o acesso ao banco é feito exclusivamente pelo backend, que já valida a autenticação JWT antes de qualquer operação de escrita.

---

12. Referências Técnicas

Recurso	URL
Repositório GitHub	https://github.com/Hedeson/glossario-libras/tree/Gama
Aplicação em produção	https://glossario-libras.onrender.com
Painel Supabase	https://supabase.com/dashboard
Painel Render	https://dashboard.render.com
Documentação Supabase	https://supabase.com/docs
Documentação Render	https://render.com/docs
Documentação Vite PWA	https://vite-pwa-org.netlify.app
Documentação Express	https://expressjs.com

Documentação React	https://react.dev

Manual gerado em 22 de maio de 2026 — Versão do sistema: 3.0.8