

# BookShelf (Deploy)

A sugestão para deploy é:

- Aplicação na vercel
- Database no Turso

Lembrando que é apenas uma sugestão - cada grupo/aluno pode fazer o deploy da forma que melhor entender.

## Instruções para Deploy na Vercel

Este documento contém instruções genéricas para fazer deploy da aplicação BookShelf v2 na Vercel.

### Pré-requisitos

1. Conta na [Vercel](#)
2. Projeto configurado no GitHub, GitLab ou Bitbucket
3. Node.js 18+ configurado localmente

## Deploy Automático via Git

### 1. Conectar Repositório

1. Acesse o [dashboard da Vercel](#)
2. Clique em "New Project"
3. Selecione seu provedor Git (GitHub, GitLab, Bitbucket)
4. Escolha o repositório do BookShelf v2
5. Clique em "Import"

### 2. Configurações do Projeto

A Vercel detectará automaticamente que é um projeto Next.js. Configurações padrão:

- **Framework Preset:** Next.js
- **Root Directory:** ./ (raiz do projeto)

- **Build Command:** `npm run build`
- **Output Directory:** `.next` (automático)
- **Install Command:** `npm install`

### 3. Variáveis de Ambiente

Se necessário, configure as variáveis de ambiente:

1. Na seção "Environment Variables"
2. Adicione as variáveis necessárias para produção
3. Exemplo:

`NODE_ENV=production`

`NEXT_PUBLIC_APP_URL=https://seu-dominio.vercel.app`

### 4. Deploy

1. Clique em "Deploy"
2. Aguarde o processo de build e deploy
3. Sua aplicação estará disponível em <https://seu-projeto.vercel.app>

## Deploy via CLI

### 1. Instalar Vercel CLI

```
npm i -g vercel
```

### 2. Login

```
vercel login
```

### 3. Deploy

Na raiz do projeto:

```
# Deploy de preview  
vercel
```

```
# Deploy de produção
```

```
vercel --prod
```

## Deploy Manual via Dashboard

### 1. Build Local

```
npm run build
```

### 2. Upload

1. No dashboard da Vercel, clique em "New Project"
2. Selecione "Import Third-Party Git Repository"
3. Faça upload da pasta `.next` ou de todo o projeto

## Configurações Avançadas

### Domain Personalizado

1. No projeto na Vercel, vá em "Settings" > "Domains"
2. Adicione seu domínio personalizado
3. Configure os DNS conforme instruído

### Performance

A Vercel otimiza automaticamente:

- Compressão Gzip/Brotli
- Cache de assets estáticos
- Edge Network global
- Image Optimization (Next.js)

### Monitoramento

- **Analytics:** Ative nas configurações do projeto
- **Functions:** Monitoramento de serverless functions
- **Logs:** Acesse via dashboard ou CLI (`vercel logs`)

## Comandos Úteis da CLI

<code>vercel --help</code>	# Ajuda
<code>vercel ls</code>	# Listar projetos
<code>vercel logs</code>	# Ver logs
<code>vercel env ls</code>	# Listar variáveis de ambiente
<code>vercel env add</code>	# Adicionar variável
<code>vercel domains ls</code>	# Listar domínios
<code>vercel inspect [url]</code>	# Inspecionar deployment

## Considerações sobre Banco de Dados

### Problema com SQLite na Vercel

⚠ **Importante:** SQLite **NÃO funciona** em ambientes serverless como a Vercel devido às seguintes limitações:

1. **Filesystem Read-Only:** O sistema de arquivos da Vercel é somente leitura em produção
2. **Stateless Functions:** Cada requisição pode ser executada em um servidor diferente
3. **Sem Persistência:** Dados escritos em uma função serverless não persistem entre execuções
4. **Cold Starts:** Bancos locais são perdidos a cada cold start da função

### Solução Recomendada: Turso DB

Para resolver essas limitações, recomendamos o **Turso DB** - um banco SQLite distribuído para edge computing:

#### 1. Configurar Turso DB

```
# Instalar CLI do Turso
curl -sSfL <https://get.tur.so/install.sh> | bash
```

```
# Login
turso auth login
```

```
# Criar database
turso db create bookshelf-v2
```

```
# Obter URL de conexão
```

```
turso db show bookshelf-v2 --url  
  
# Criar token de acesso  
turso db tokens create bookshelf-v2
```

## 2. Instalar Cliente

```
npm install @libsql/client
```

## 3. Configurar Variáveis de Ambiente







Na Vercel, adicione as variáveis:

```
TURSO_DATABASE_URL=libsql://sua-database-url.turso.io  
TURSO_AUTH_TOKEN=seu-token-aqui
```

## 4. Exemplo de Configuração

```
// lib/db.ts  
import { createClient } from '@libsql/client';  
  
export const db = createClient({  
  url: process.env.TURSO_DATABASE_URL!,  
  authToken: process.env.TURSO_AUTH_TOKEN!,  
});
```

## 5. Vantagens do Turso

-  Compatible com SQLite (mesmo SQL)
-  Edge replicas globais (baixa latência)
-  Funciona perfeitamente em serverless
-  Migrations automáticas
-  Free tier generoso (500 databases, 9GB storage)
-  Backup automático

## 6. Alternativas

Se não quiser usar Turso, outras opções compatíveis com Vercel:

- **PostgreSQL:** Neon, Supabase, Railway
- **MySQL:** PlanetScale, Upstash
- **NoSQL:** MongoDB Atlas, FaunaDB