🚀 Guia de Migração: Vercel → Abacus.AI

Projeto: DinDin Teens
Data: Janeiro 2026
Status: Pronto para migração

📋 Índice

🎯 Visão Geral

Estado Atual (Vercel)

- **URL**: https://profdindin-teens.vercel.app
- **Banco**: Supabase PostgreSQL (externo)
- **Deploy**: Automático via Git push
- **Domínio**: Subdomínio Vercel

Estado Futuro (Abacus.AI)

- **URL**: https://profdindin-teens.abacusai.app (ou domínio customizado)
- **Banco**: PostgreSQL gerenciado pela Abacus
- **Deploy**: Via DeepAgent (on-demand)
- **Estrutura**: `/home/ubuntu/profdindin-teens/nextjs_space`

Benefícios da Migração

✅ **Banco de dados incluído** - Não precisa gerenciar Supabase separadamente
✅ **Controle total** - Acesso SSH via DeepAgent
✅ **Custos previsíveis** - Sem surpresas com serverless
✅ **Melhor performance** - Otimizações específicas da Abacus
✅ **Integração nativa** - APIs da Abacus.AI (LLMs, etc)

⚙️ Pré-requisitos

1. Acesso ao DeepAgent da Abacus.AI

Você precisa ter acesso ao ambiente do DeepAgent onde executará os comandos de migração.

2. Repositório Git Atualizado

Garantir que o código está commitado e pushed:

```
cd /home/ubuntu/github_repos/profdindin-teens
git status
git add .
git commit -m "feat: Preparar projeto para migração Abacus.AI"
git push origin main
```

3. Backup do Banco de Dados Atual (Supabase)

IMPORTANTE: Exporte os dados antes de migrar!

```
# Conectar ao Supabase e exportar
pg_dump "postgresql://postgres.xsdlhzqvcgcovnxchmqe:wVg67IkNudcn1a1J@aws-1-sa-
east-1.pooler.supabase.com:5432/postgres" > backup_supabase_$(date +%Y%m%d).sql
```

Ou via Prisma:

```
cd /home/ubuntu/github_repos/profdindin-teens
npx prisma db pull --schema=backup-schema.prisma
```

4. Lista de Variáveis de Ambiente

Anote as variáveis atuais que precisam ser migradas:
- `NEXTAUTH_SECRET`
- Qualquer API key de terceiros
- Configurações específicas do projeto

🏗️ Estrutura do Projeto na Abacus

Estrutura de Diretórios

```
/home/ubuntu/profdindin-teens/
├── nextjs_space/              # Código Next.js (raiz do projeto)
│   ├── src/
│   ├── prisma/
│   ├── public/
│   ├── package.json
│   ├── next.config.js
│   ├── .env                   # Variáveis de ambiente
│   ├── .env.example           # Template
│   └── node_modules/          # Symlink otimizado
│
├── docs/                      # Documentação (opcional)
│   ├── MIGRACAO_ABACUS.md
│   └── GUIA_DEPLOY.md
│
└── backups/                   # Backups do banco (opcional)
    └── backup_YYYYMMDD.sql
```

Por que `nextjs_space`?

A Abacus.AI usa o diretório `nextjs_space` como convenção padrão para projetos Next.js. Isso permite:
- Separação clara entre código e documentação
- Otimizações específicas (symlink de node_modules)
- Compatibilidade com ferramentas internas

📦 Processo de Migração

Passo 1: Criar Estrutura do Projeto

```bash
# Criar diretório do projeto na Abacus
mkdir -p /home/ubuntu/profdindin-teens
cd /home/ubuntu/profdindin-teens

# Clonar repositório
git clone https://github.com/marceloegito-max/profdindin-teens.git nextjs_space

# Entrar no diretório
cd nextjs_space
```

Passo 2: Configurar Variáveis de Ambiente

```bash
# Copiar template
cp .env.example .env

# Editar variáveis manualmente
vim .env
```

Conteúdo inicial do `.env`:

```
# ===== GERADO AUTOMATICAMENTE PELA ABACUS =====
# Estas variáveis serão preenchidas automaticamente:
DATABASE_URL=''
ABACUSAI_API_KEY=''
WEB_APP_ID=''

# ===== VOCÊ PRECISA CONFIGURAR =====

# NextAuth.js
NEXTAUTH_SECRET='dindin-teens-super-secret-2026'  # Use o mesmo ou gere novo
NEXTAUTH_URL='http://localhost:3000'              # Será ajustado depois

# Ambiente
NODE_ENV='development'  # Mude para 'production' no deploy

# ===== OPCIONAL =====
# Google Analytics (se usar)
# NEXT_PUBLIC_GA_MEASUREMENT_ID='G-XXXXXXXXXX'

# Outras APIs de terceiros
# STRIPE_SECRET_KEY='sk_test_...'
# SENDGRID_API_KEY='SG...'
```

Passo 3: Inicializar Banco de Dados

NOTA: Este comando deve ser executado via DeepAgent (não bash manual).

O DeepAgent executará:

```
# Via interface do DeepAgent
initialize_postgres_db(
    project_path="/home/ubuntu/profdindin-teens"
)
```

Isso irá:
1. ✅ Criar banco PostgreSQL gerenciado
2. ✅ Gerar credenciais automaticamente
3. ✅ Atualizar `DATABASE_URL` no `.env`
4. ✅ Configurar connection pooling

Resultado esperado no `.env`:

```
DATABASE_URL='postgresql://role_abc123:senha_gerada@db-abc123.db003.hosteddb.reai.io:5432/abc123?connect_timeout=15'
```

Passo 4: Instalar Dependências

```
cd /home/ubuntu/profdindin-teens/nextjs_space

# Instalar com Yarn (preferencial na Abacus)
yarn install

# Ou com npm
# npm install
```

Passo 5: Configurar Prisma

```
# Gerar Prisma Client
yarn prisma generate

# Aplicar schema no banco novo
yarn prisma db push

# (Opcional) Popular dados iniciais
yarn prisma db seed
```

Passo 6: Migrar Dados do Supabase

Opção A: Importar via SQL dump

```
# Se você tem backup SQL do Supabase
psql "$DATABASE_URL" < backup_supabase_20260131.sql
```

Opção B: Usar Prisma para copiar

```
# Criar script de migração
cat > migrate-data.ts << 'EOF'
import { PrismaClient } from '@prisma/client';

const oldDb = new PrismaClient({
  datasources: {
    db: {
      url: 'postgresql://postgres.xsdlhzqvcgcovnxchmqe:wVg67IkNudcn1a1J@aws-1-sa-
east-1.pooler.supabase.com:5432/postgres'
    }
  }
});

const newDb = new PrismaClient();

async function migrate() {
  console.log('Migrando dados...');

  // Copiar usuários
  const users = await oldDb.user.findMany();
  await newDb.user.createMany({ data: users, skipDuplicates: true });

  // Copiar outros models...
  // await newDb.activity.createMany(...);

  console.log('Migração concluída!');
}

migrate()
  .catch(console.error)
  .finally(() => {
    oldDb.$disconnect();
    newDb.$disconnect();
  });
EOF

# Executar migração
npx tsx migrate-data.ts
```

Passo 7: Testar Localmente

```
# Iniciar servidor de desenvolvimento
yarn dev

# Abrir no navegador
# http://localhost:3000
```

Checklist de testes:
- [ ] Login funciona
- [ ] Dashboard carrega
- [ ] API `/api/dashboard` retorna dados
- [ ] Gamificação (badges, XP) funciona
- [ ] ISJF (Raio-X) funciona
- [ ] Mensagens funcionam
- [ ] Sem erros no console

Passo 8: Build Local (Teste)

```
# Build de produção
NODE_OPTIONS="--max-old-space-size=6144" yarn build

# Verificar se criou .next/standalone
ls -la .next/standalone/

# Testar build localmente
cd .next/standalone
node server.js
```

Resultado esperado:

```
✓ Compiled successfully
✓ Ready on http://localhost:3000
```

🗄️ Configuração do Banco de Dados

Schema Prisma

O schema já está definido em `prisma/schema.prisma`. Principais models:

```
model User {
  id            String    @id @default(cuid())
  email         String    @unique
  passwordHash  String?
  role          UserRole  @default(TEEN)
  // ... outros campos
}

model UserProgress {
  id        String   @id @default(cuid())
  userId    String   @unique
  xp        Int      @default(0)
  level     Int      @default(1)
  // ... estatísticas de progresso
}

model Badge {
  id             String      @id @default(cuid())
  name           String      @unique
  description    String
  icon           String
  criteria       String
  requiredValue  Int
  // ... gamificação
}

// ... outros 40+ models
```

Seed de Dados Iniciais

O projeto já tem seed configurado em `prisma/seed.ts`:

```
# Popular banco com dados de exemplo
yarn prisma db seed
```

Isso criará:
- ✅ Usuários de teste (teen, professor, responsável, admin)
- ✅ Instituições de ensino
- ✅ Turmas e atividades
- ✅ Badges e conquistas
- ✅ Missões diárias

Backup e Restore

Criar backup:

```
# Via pg_dump
pg_dump "$DATABASE_URL" > backup_$(date +%Y%m%d_%H%M%S).sql

# Via Prisma
npx prisma db pull --schema=backup-schema.prisma
```

Restaurar backup:

```
# Restaurar SQL
psql "$DATABASE_URL" < backup_20260131_143000.sql

# Ou via Prisma (recriar tabelas)
yarn prisma db push --force-reset
yarn prisma db seed
```

🔐 Variáveis de Ambiente

Template Completo (`.env.example`)

```
# ===== BANCO DE DADOS =====
# Gerado automaticamente pela Abacus via initialize_postgres_db
DATABASE_URL='postgresql://role_XXXXX:PASSWORD@db-XXXXX.db003.hosteddb.reai.io:5432/
XXXXX?connect_timeout=15'

# ===== NEXTAUTH.JS =====
# Gerar com: openssl rand -base64 32
NEXTAUTH_SECRET='your-secret-here'

# Desenvolvimento
NEXTAUTH_URL='http://localhost:3000'

# Produção (ajustar após deploy)
# NEXTAUTH_URL='https://profdindin-teens.abacusai.app'
# ou
# NEXTAUTH_URL='https://seu-dominio.com.br'

# ===== ABACUS.AI APIS =====
# Gerado automaticamente via initialize_llm_apis
ABACUSAI_API_KEY=''

# ID da aplicação web (gerado automaticamente)
WEB_APP_ID=''

# ===== AMBIENTE =====
NODE_ENV='development'  # ou 'production'

# ===== ANALYTICS (OPCIONAL) =====
# NEXT_PUBLIC_GA_MEASUREMENT_ID='G-XXXXXXXXXX'

# ===== TERCEIROS (SE NECESSÁRIO) =====
# STRIPE_SECRET_KEY='sk_test_...'
# SENDGRID_API_KEY='SG...'
# TWILIO_ACCOUNT_SID='AC...'
# TWILIO_AUTH_TOKEN='...'
```

Variáveis Obrigatórias

Variável	Obrigatória?	Gerada pela Abacus?	Descrição
`DATABASE_URL`	✅ Sim	✅ Sim	URL de conexão PostgreSQL
`NEXTAUTH_SECRET`	✅ Sim	❌ Não	Secret para NextAuth
`NEXTAUTH_URL`	✅ Sim	❌ Não	URL da aplicação
`NODE_ENV`	✅ Sim	❌ Não	Ambiente (dev/prod)
`ABACUSAI_API_KEY`	⚠️ Se usar LLMs	✅ Sim	API key da Abacus
`WEB_APP_ID`	⚠️ Deploy	✅ Sim	ID da aplicação web

Gerar `NEXTAUTH_SECRET`

```
# Linux/Mac
openssl rand -base64 32

# Node.js
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Python
python3 -c "import secrets; print(secrets.token_urlsafe(32))"
```

Diferenças Vercel vs Abacus

Aspecto	Vercel	Abacus.AI
Arquivo	Dashboard Web	`.env` no servidor
Sincronização	Automática	Manual (editar arquivo)
Acesso	Via UI	Via DeepAgent/SSH
Rollback	Por deploy	Via controle de versão
Secrets	Encriptados	Arquivo local (cuidado!)

🚀 Deploy para Produção

Pré-Deploy Checklist

- [ ] ✅ Todos os testes locais passando
- [ ] ✅ Build local bem-sucedido

- [ ] ✅ Dados migrados do Supabase
- [ ] ✅ Variáveis de ambiente configuradas
- [ ] ✅ `NODE_ENV=production` no `.env`
- [ ] ✅ `NEXTAUTH_URL` com URL de produção
- [ ] ✅ Git commitado e pushed
- [ ] ✅ Backup do banco atual

Deploy via DeepAgent

NOTA: O deploy é executado via DeepAgent, não manualmente.

O DeepAgent executará:

```
# Via interface do DeepAgent
deploy_nextjs_project(
    project_path="/home/ubuntu/profdindin-teens",
    hostname="profdindin-teens.abacusai.app"  # ou domínio customizado
)
```

O que o Deploy Faz Automaticamente

1. **Symlink node_modules**
   `bash`
   `ln -sf /opt/nodejs/node_modules nextjs_space/node_modules`
2. **Rsync do código**
   `bash`
   `rsync -av --exclude='node_modules' --exclude='.next' --exclude='.env' \`
   `nextjs_space/ /deploy/staging/`
3. **Instalar dependências**
   `bash`
   `cd /deploy/staging`
   `yarn install`
4. **Gerar Prisma Client**
   `bash`
   `yarn prisma generate`
5. **Build otimizado**
   `bash`
   `NODE_OPTIONS="--max-old-space-size=6144" yarn build`
6. **Criar standalone bundle**
   `bash`
   `cp -r .next/standalone /deploy/production/`
   `cp -r .next/static /deploy/production/.next/static`
   `cp -r public /deploy/production/public`
7. **Restart servidor**
   `bash`
   `pm2 restart profdindin-teens --update-env`

Verificar Deploy

```
# Verificar logs do deploy
tail -f /var/log/abacus/profdindin-teens-deploy.log

# Verificar status do servidor
pm2 status profdindin-teens

# Ver logs em tempo real
pm2 logs profdindin-teens

# Métricas
pm2 monit
```

Testar Produção

```
# Testar API
curl https://profdindin-teens.abacusai.app/api/dashboard

# Testar página inicial
curl https://profdindin-teens.abacusai.app
```

Checklist pós-deploy:
- [ ] ✅ Site carrega sem erros
- [ ] ✅ Login funciona
- [ ] ✅ Dashboard mostra dados reais
- [ ] ✅ Sem erros no console do navegador
- [ ] ✅ APIs retornam dados corretos
- [ ] ✅ SSL/HTTPS funcionando

Configurar Domínio Customizado (Opcional)

Se quiser usar domínio próprio (ex: `profdindin-teens.com.br`):

1. **Configurar DNS**:
   `Tipo: CNAME`
   `Nome: @ (ou www)`
   `Valor: profdindin-teens.abacusai.app`
   `TTL: 300`

2. **Atualizar `.env`**:
   `bash`
   `NEXTAUTH_URL='https://profdindin-teens.com.br'`

3. **Redeploy**:
   `python`
   `deploy_nextjs_project(`
   `project_path="/home/ubuntu/profdindin-teens",`
   `hostname="profdindin-teens.com.br"`
   `)`

🐛 Troubleshooting

Erro: “Module not found”

Sintoma: Build falha com `Module not found: Can't resolve '@/lib/...'`

Solução:

```
# Verificar tsconfig.json
cat tsconfig.json | grep -A 5 "paths"

# Deve ter:
# "paths": {
#   "@/*": ["./src/*"]
# }

# Limpar cache e recompilar
rm -rf .next
rm -rf node_modules/.cache
yarn build
```

Erro: Prisma Client não encontrado

Sintoma: `@prisma/client` não está gerando corretamente

Solução:

```
# Reinstalar Prisma
rm -rf node_modules/@prisma
yarn add -D prisma
yarn add @prisma/client

# Regenerar cliente
yarn prisma generate

# Verificar se gerou
ls -la node_modules/.prisma/client/
```

Erro: Conexão com banco de dados

Sintoma: `Error: P1001: Can't reach database server`

Solução:

```
# Verificar DATABASE_URL no .env
cat .env | grep DATABASE_URL

# Testar conexão
psql "$DATABASE_URL" -c "SELECT 1;"

# Verificar se Prisma consegue conectar
yarn prisma db pull
```

Erro: Build out of memory

Sintoma: `JavaScript heap out of memory`

Solução:

```
# Aumentar memória do Node.js
NODE_OPTIONS="--max-old-space-size=6144" yarn build

# Ou adicionar ao package.json:
# "scripts": {
#   "build": "NODE_OPTIONS='--max-old-space-size=6144' next build"
# }
```

Erro: NextAuth callback URL mismatch

Sintoma: `Callback URL mismatch` ao fazer login

Solução:

```
# Atualizar NEXTAUTH_URL no .env
vim .env

# Desenvolvimento:
NEXTAUTH_URL='http://localhost:3000'

# Produção:
NEXTAUTH_URL='https://profdindin-teens.abacusai.app'

# Reiniciar servidor
pm2 restart profdindin-teens
```

Erro: TypeScript type errors

Sintoma: Build falha com erros de tipo

Solução:

```
# Verificar erros
yarn tsc --noEmit

# Regenerar tipos do Prisma
yarn prisma generate

# Limpar cache do TypeScript
rm -rf .next
rm -rf node_modules/.cache
```

Deploy não atualiza

Sintoma: Deploy executado mas site não muda

Solução:

```
# Verificar se código foi rsync corretamente
ls -la /deploy/production/

# Verificar logs do PM2
pm2 logs profdindin-teens --lines 100

# Hard restart
pm2 delete profdindin-teens
deploy_nextjs_project(project_path="/home/ubuntu/profdindin-teens")
```

⏮️ Rollback

Rollback para Vercel (Emergência)

Se algo der muito errado, você pode reverter para a Vercel temporariamente:

1. **Garantir que Vercel está funcionando**:
   `bash`
   `curl https://profdindin-teens.vercel.app`
2. **Atualizar DNS** (se mudou):
   `Tipo: CNAME`
   `Nome: @`
   `Valor: cname.vercel-dns.com`
3. **Avisar usuários sobre downtime**

Rollback de Deploy na Abacus

Se o deploy na Abacus deu problema:

Opção 1: Rollback via Git

```
cd /home/ubuntu/profdindin-teens/nextjs_space

# Ver histórico de commits
git log --oneline -10

# Voltar para commit anterior
git reset --hard <commit_id_anterior>

# Redeploy
# Via DeepAgent: deploy_nextjs_project(...)
```

Opção 2: Restaurar Checkpoint

Se você salvou checkpoints:

```
# Via DeepAgent
restore_nextjs_project_checkpoint(
    project_path="/home/ubuntu/profdindin-teens",
    checkpoint_id="checkpoint_20260131_120000"
)
```

Opção 3: Rollback do Banco de Dados

```
# Restaurar backup anterior
psql "$DATABASE_URL" < backups/backup_20260131_120000.sql
```

📊 Comparação: Antes vs Depois

Aspecto	Vercel	Abacus.AI
Hospedagem	Serverless	Servidor dedicado
Banco de Dados	Supabase (externo)	PostgreSQL incluído
Deploy	Git push automático	DeepAgent on-demand
Custos	$20-50/mês	Incluso no plano
Performance	Edge Functions	Node.js otimizado
Acesso SSH	❌ Não	✅ Sim (via DeepAgent)
Logs	Dashboard Web	PM2 + SSH
Escalabilidade	Automática	Manual (otimizada)
Cold Starts	Sim	Não
Controle	Limitado	Total

✅ Checklist Final

Pré-Migração

- [ ] ✅ Backup do Supabase criado
- [ ] ✅ Variáveis de ambiente documentadas
- [ ] ✅ Código commitado no Git
- [ ] ✅ Testes locais passando

Durante Migração

- [ ] ✅ Projeto clonado em `/home/ubuntu/profdindin-teens/nextjs_space`
- [ ] ✅ Banco PostgreSQL inicializado via `initialize_postgres_db`
- [ ] ✅ Variáveis de ambiente configuradas no `.env`
- [ ] ✅ Dependências instaladas (`yarn install`)
- [ ] ✅ Prisma Client gerado (`yarn prisma generate`)
- [ ] ✅ Schema aplicado (`yarn prisma db push`)
- [ ] ✅ Dados migrados do Supabase
- [ ] ✅ Seed executado (`yarn prisma db seed`)
- [ ] ✅ Testes locais OK (`yarn dev`)
- [ ] ✅ Build local OK (`yarn build`)

Deploy

- [ ] ✅ `NODE_ENV=production` configurado

- [ ] ✅ `NEXTAUTH_URL` com URL de produção
- [ ] ✅ Deploy executado via DeepAgent
- [ ] ✅ Site acessível e funcionando
- [ ] ✅ Login funciona
- [ ] ✅ Dashboard mostra dados reais
- [ ] ✅ APIs retornam corretamente
- [ ] ✅ Sem erros no console

Pós-Deploy

- [ ] ✅ Domínio DNS configurado (se customizado)
- [ ] ✅ SSL/HTTPS funcionando
- [ ] ✅ Monitoramento ativo (PM2)
- [ ] ✅ Backups automáticos configurados
- [ ] ✅ Usuários notificados da migração
- [ ] ✅ Vercel deployment pausado/deletado

🆘 Suporte

Se encontrar problemas:

1. **Verificar logs do PM2**:
   ```bash
   pm2 logs profdindin-teens
   ```
2. **Verificar status do serviço**:
   ```bash
   pm2 status
   ```
3. **Consultar documentação da Abacus**:
   - Abacus.AI Docs (https://docs.abacus.ai)
   - Next.js Standalone Mode (https://nextjs.org/docs/advanced-features/output-file-tracing)
4. **Contatar suporte da Abacus.AI**

📚 Referências

- Next.js Documentation (https://nextjs.org/docs)
- Prisma Documentation (https://www.prisma.io/docs)
- NextAuth.js Documentation (https://next-auth.js.org)
- PostgreSQL Documentation (https://www.postgresql.org/docs/)
- PM2 Documentation (https://pm2.keymetrics.io/docs/)

Data de criação: 31/01/2026
Última atualização: 31/01/2026

Versão: 1.0
Status: ✅ Pronto para uso