🚀 NEXUS SST - Do Zero à Produção (100% Gratuito)

📋 Checklist Inicial

Antes de começar, você precisará de:

- [ ] Computador com internet
- [ ] Email pessoal
- [ ] Navegador (Chrome, Firefox, Edge)
- [ ] Conta no Gmail (recomendado para facilitar logins)

🔧 PARTE 1: PREPARAÇÃO DO AMBIENTE LOCAL

1.1 Instalar Node.js (Obrigatório)

Windows:

```bash
# 1. Acesse: https://nodejs.org
# 2. Clique em "Download" (versão LTS - recomendada)
# 3. Execute o arquivo baixado
# 4. Clique "Next" em tudo, depois "Install"
# 5. Abra o "Prompt de Comando" e teste:
node --version
npm --version
```

Mac:

```bash
# 1. Acesse: https://nodejs.org
# 2. Baixe a versão LTS para Mac
# 3. Execute o .pkg e siga as instruções
# 4. Abra o Terminal e teste:
node --version
npm --version
```

Linux (Ubuntu/Debian):

```bash
```

```
# Abra o terminal e digite:
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
sudo apt-get install -y nodejs

# Teste:
node --version
npm --version
```

1.2 Instalar Git (Para gerenciar código)

Windows:

```bash
# 1. Acesse: https://git-scm.com/download/windows
# 2. Baixe e execute o instalador
# 3. Aceite todas as configurações padrão
# 4. Abra o "Git Bash" e teste:
git --version
```

Mac:

```bash
# No Terminal:
git --version
# Se não estiver instalado, o Mac pedirá para instalar automaticamente
```

Linux:

```bash
sudo apt update
sudo apt install git
git --version
```

1.3 Instalar VS Code (Editor de código)

```bash
```

```
# 1. Acesse: https://code.visualstudio.com
# 2. Clique "Download" para seu sistema
# 3. Instale normalmente
# 4. Abra o VS Code
# 5. Instale extensões úteis:
#    - Extensão "ES7+ React/Redux/React-Native snippets"
#    - Extensão "Prettier - Code formatter"
#    - Extensão "Auto Rename Tag"
```

🌐 PARTE 2: CRIAR TODAS AS CONTAS GRATUITAS

2.1 GitHub (Armazenar código)

bash

```
# 1. Acesse: https://github.com
# 2. Clique "Sign up"
# 3. Preencha:
#    - Username: nexus-sst-dev (ou outro de sua escolha)
#    - Email: seu-email@gmail.com
#    - Password: senha-forte-123!
# 4. Verifique o email
# 5. Escolha plano "Free"
```

Após criar conta no GitHub:

bash

```
# Configure git no seu PC:
git config --global user.name "Seu Nome"
git config --global user.email "seu-email@gmail.com"
```

2.2 Supabase (Banco de Dados PostgreSQL)

bash

```
# 1. Acesse: https://supabase.com
# 2. Clique "Start your project"
# 3. Clique "Sign in with GitHub" (use a conta criada acima)
# 4. Autorize a aplicação
# 5. Clique "New Project"
# 6. Preencha:
#    - Name: nexus-sst-db
#    - Database Password: senha-db-123! (ANOTE ESTA SENHA!)
#    - Region: South America (mais próximo do Brasil)
# 7. Clique "Create new project"
# 8. Aguarde 1-2 minutos para criar

# ✅ ANOTAR AS CREDENCIAIS:
# Na tela do projeto, vá em "Settings" > "API"
# Copie e salve:
# - Project URL: https://xxx.supabase.co
# - anon/public key: eyJ...
# - service_role key: eyJ... (só para admin)
```

2.3 Render (Backend 24/7)

bash

```
# 1. Acesse: https://render.com
# 2. Clique "Get Started"
# 3. Clique "Sign up with GitHub"
# 4. Autorize a aplicação
# 5. Complete o perfil (nome, empresa opcional)
# 6. Não precisa configurar nada agora - faremos depois
```

2.4 Vercel (Frontend Admin)

bash

```
# 1. Acesse: https://vercel.com
# 2. Clique "Sign Up"
# 3. Clique "Continue with GitHub"
# 4. Autorize a aplicação
# 5. Escolha "Hobby" (gratuito)
# 6. Não precisa configurar nada agora
```

2.5 Netlify (Frontend Público)

bash

```
# 1. Acesse: https://netlify.com
# 2. Clique "Sign up"
# 3. Clique "Sign up with GitHub"
# 4. Autorize a aplicação
# 5. Complete o perfil básico
# 6. Não precisa configurar nada agora
```

2.6 Cloudinary (Storage de Imagens/Vídeos)

bash

```
# 1. Acesse: https://cloudinary.com
# 2. Clique "Sign Up for Free"
# 3. Preencha o formulário:
#    - Email: seu-email@gmail.com
#    - Password: senha-forte-123!
#    - Cloud name: nexus-sst (será usado na URL)
# 4. Verifique o email
# 5. Na dashboard, clique no ícone de engrenagem (Settings)
# 6. Vá em "Access Keys"

# ✅ ANOTAR AS CREDENCIAIS:
# - Cloud name: nexus-sst
# - API Key: 123456789
# - API Secret: abc123def456 (clique no olho para ver)
```

2.7 Resend (Emails)

bash

```
# 1. Acesse: https://resend.com
# 2. Clique "Sign Up"
# 3. Preencha:
#    - Email: seu-email@gmail.com
#    - Password: senha-forte-123!
# 4. Verifique o email
# 5. Após login, vá em "API Keys"
# 6. Clique "Create API Key"
# 7. Nome: "NEXUS-SST-Production"
# 8. Permissions: "Sending access"
# 9. Clique "Add"

# ✅ ANOTAR A CHAVE:
# - API Key: re_123abc... (COPIE E GUARDE!)
```

💻 PARTE 3: CRIAR O PROJETO LOCALMENTE

3.1 Criar Estrutura do Projeto

bash

```
# Abra o terminal/prompt e navegue para uma pasta de sua escolha
# Exemplo: C:\projetos ou /home/usuario/projetos

mkdir nexus-sst-platform
cd nexus-sst-platform

# Criar estrutura de pastas:
mkdir backend
mkdir frontend-admin
mkdir frontend-public
mkdir docs

# Inicializar git:
git init
```

3.2 Criar Backend (API Node.js)

bash

```
# Entrar na pasta backend:
cd backend

# Inicializar projeto Node:
npm init -y

# Instalar dependências:
npm install express cors helmet bcryptjs jsonwebtoken pg nodemailer dotenv joi express-rate-limit morgan moment uu

# Instalar dependências de desenvolvimento:
npm install --save-dev nodemon

cd ..
```

3.2.1 Criar arquivo principal do backend:

javascript

```
// backend/server.js
const express = require('express');
const cors = require('cors');
const helmet = require('helmet');
const rateLimit = require('express-rate-limit');
const morgan = require('morgan');
require('dotenv').config();

const authRoutes = require('./routes/auth');
const coursesRoutes = require('./routes/courses');
const dashboardRoutes = require('./routes/dashboard');

const app = express();
const PORT = process.env.PORT || 3001;

// Middlewares de segurança
app.use(helmet());
app.use(cors({
    origin: process.env.FRONTEND_URLS?.split(',') || ['http://localhost:3000'],
    credentials: true
}));

// Rate limiting
const limiter = rateLimit({
    windowMs: 15 * 60 * 1000, // 15 minutos
    max: 100 // máximo 100 requests por IP
});
app.use(limiter);

// Middlewares
app.use(morgan('combined'));
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ extended: true }));

// Rotas
app.use('/api/auth', authRoutes);
app.use('/api/courses', coursesRoutes);
app.use('/api/dashboard', dashboardRoutes);

// Health check
app.get('/api/health', (req, res) => {
    res.json({
        status: 'OK',
        timestamp: new Date().toISOString(),
        service: 'NEXUS SST API'
    });
```

```
});

// Rota teste
app.get('/api/test', (req, res) => {
    res.json({
        message: 'NEXUS SST API funcionando!',
        environment: process.env.NODE_ENV || 'development'
    });
});

// Error handling
app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).json({
        error: 'Algo deu errado!',
        message: process.env.NODE_ENV === 'development' ? err.message : undefined
    });
});

app.listen(PORT, '0.0.0.0', () => {
    console.log(`🚀 NEXUS SST API rodando na porta ${PORT}`);
    console.log(`📊 Environment: ${process.env.NODE_ENV || 'development'}`);
});
```

3.2.2 Criar rotas de autenticação:

bash

```
# Criar pasta routes:
mkdir backend/routes
```

javascript

```
// backend/routes/auth.js
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { createClient } = require('@supabase/supabase-js');

const router = express.Router();

// Inicializar Supabase
const supabase = createClient(
    process.env.SUPABASE_URL,
    process.env.SUPABASE_ANON_KEY
);

// Registrar usuário
router.post('/register', async (req, res) => {
    try {
        const { email, password, firstName, lastName, companyName } = req.body;

        // Validar dados
        if (!email || !password || !firstName || !lastName) {
            return res.status(400).json({ error: 'Todos os campos são obrigatórios' });
        }

        // Hash da senha
        const hashedPassword = await bcrypt.hash(password, 12);

        // Criar usuário no Supabase
        const { data, error } = await supabase
            .from('users')
            .insert([
                {
                    email,
                    password_hash: hashedPassword,
                    first_name: firstName,
                    last_name: lastName,
                    company_name: companyName || null,
                    role: 'user'
                }
            ])
            .select();

        if (error) {
            console.error(error);
            return res.status(400).json({ error: 'Erro ao criar usuário' });
        }
```

```
        res.status(201).json({
            message: 'Usuário criado com sucesso!',
            user: {
                id: data[0].id,
                email: data[0].email,
                name: `${data[0].first_name} ${data[0].last_name}`
            }
        });

    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Erro interno do servidor' });
    }
});

// Login
router.post('/login', async (req, res) => {
    try {
        const { email, password } = req.body;

        // Buscar usuário
        const { data: users, error } = await supabase
            .from('users')
            .select('*')
            .eq('email', email)
            .limit(1);

        if (error || !users || users.length === 0) {
            return res.status(401).json({ error: 'Credenciais inválidas' });
        }

        const user = users[0];

        // Verificar senha
        const validPassword = await bcrypt.compare(password, user.password_hash);
        if (!validPassword) {
            return res.status(401).json({ error: 'Credenciais inválidas' });
        }

        // Criar token JWT
        const token = jwt.sign(
            {
                userId: user.id,
                email: user.email,
                role: user.role
            },
```

```
            process.env.JWT_SECRET,
            { expiresIn: '7d' }
        );

        res.json({
            message: 'Login realizado com sucesso!',
            token,
            user: {
                id: user.id,
                email: user.email,
                name: `${user.first_name} ${user.last_name}`,
                role: user.role
            }
        });

    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Erro interno do servidor' });
    }
});

module.exports = router;
```

3.2.3 Criar rotas de cursos:

```
javascript
```

```
// backend/routes/courses.js
const express = require('express');
const { createClient } = require('@supabase/supabase-js');

const router = express.Router();

const supabase = createClient(
    process.env.SUPABASE_URL,
    process.env.SUPABASE_ANON_KEY
);

// Listar todos os cursos
router.get('/', async (req, res) => {
    try {
        const { data, error } = await supabase
            .from('courses')
            .select('*')
            .eq('status', 'active')
            .order('created_at', { ascending: false });

        if (error) {
            console.error(error);
            return res.status(500).json({ error: 'Erro ao buscar cursos' });
        }

        res.json({ courses: data });

    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Erro interno do servidor' });
    }
});

// Buscar curso por ID
router.get('/:id', async (req, res) => {
    try {
        const { id } = req.params;

        const { data, error } = await supabase
            .from('courses')
            .select('*')
            .eq('id', id)
            .eq('status', 'active')
            .single();

        if (error || !data) {
```

```
            return res.status(404).json({ error: 'Curso não encontrado' });
        }

        res.json({ course: data });

    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Erro interno do servidor' });
    }
});

module.exports = router;
```

3.2.4 Criar rotas do dashboard:

javascript

```
// backend/routes/dashboard.js
const express = require('express');
const { createClient } = require('@supabase/supabase-js');

const router = express.Router();

const supabase = createClient(
    process.env.SUPABASE_URL,
    process.env.SUPABASE_ANON_KEY
);

// Estatísticas do dashboard
router.get('/stats', async (req, res) => {
    try {
        // Buscar estatísticas básicas
        const [usersResult, coursesResult, certificatesResult] = await Promise.all([
            supabase.from('users').select('id', { count: 'exact', head: true }),
            supabase.from('courses').select('id', { count: 'exact', head: true }),
            supabase.from('certificates').select('id', { count: 'exact', head: true })
        ]);

        const stats = {
            totalUsers: usersResult.count || 0,
            totalCourses: coursesResult.count || 0,
            totalCertificates: certificatesResult.count || 0,
            completionRate: 85 // Mock por enquanto
        };

        // Dados para gráficos (mock inicial)
        const chartData = {
            coursePopularity: [
                { name: 'NR-35', enrollments: 45 },
                { name: 'NR-10', enrollments: 38 },
                { name: 'NR-06', enrollments: 32 },
                { name: 'NR-17', enrollments: 28 },
                { name: 'NR-12', enrollments: 25 }
            ],
            certificateStatus: [
                { name: 'Válidos', value: 120, color: '#4caf50' },
                { name: 'Vencidos', value: 15, color: '#ff9800' },
                { name: 'Revogados', value: 3, color: '#f44336' }
            ]
        };

        res.json({ stats, chartData });
```

```
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Erro interno do servidor' });
    }
});

module.exports = router;
```

3.2.5 Configurar package.json do backend:

```
json

{
  "name": "nexus-sst-backend",
  "version": "1.0.0",
  "description": "Backend API para plataforma de treinamento SST",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "cors": "^2.8.5",
    "helmet": "^7.0.0",
    "bcryptjs": "^2.4.3",
    "jsonwebtoken": "^9.0.0",
    "nodemailer": "^6.9.3",
    "dotenv": "^16.1.4",
    "joi": "^17.9.2",
    "express-rate-limit": "^6.7.1",
    "morgan": "^1.10.0",
    "moment": "^2.29.4",
    "uuid": "^9.0.0",
    "multer": "^1.4.5-lts.1",
    "cloudinary": "^1.37.3",
    "@supabase/supabase-js": "^2.26.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}
```

3.3 Criar Frontend Admin (React)

```
bash
```

```
# Voltar para raiz do projeto:
cd ..

# Criar app React para admin:
npx create-react-app frontend-admin
cd frontend-admin

# Instalar dependências adicionais:
npm install @mui/material @emotion/react @emotion/styled @mui/icons-material axios react-router-dom recharts @

cd ..
```

3.3.1 Criar componente Dashboard:

```
jsx
```

```jsx
// frontend-admin/src/components/Dashboard.jsx
import React, { useState, useEffect } from 'react';
import {
  Grid,
  Card,
  CardContent,
  Typography,
  Box,
  Chip,
  Paper,
  Container
} from '@mui/material';
import {
  People as PeopleIcon,
  School as SchoolIcon,
  Assignment as AssignmentIcon,
  TrendingUp as TrendingUpIcon
} from '@mui/icons-material';
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  CartesianGrid,
  Tooltip,
  ResponsiveContainer,
  PieChart,
  Pie,
  Cell
} from 'recharts';
import axios from 'axios';

const Dashboard = () => {
  const [stats, setStats] = useState({
    totalUsers: 0,
    totalCourses: 0,
    totalCertificates: 0,
    completionRate: 0
  });

  const [chartData, setChartData] = useState({
    coursePopularity: [],
    certificateStatus: []
  });

  const [loading, setLoading] = useState(true);
```

```
useEffect(() => {
  fetchDashboardData();
}, []);

const fetchDashboardData = async () => {
  try {
    const API_URL = process.env.REACT_APP_API_URL || 'http://localhost:3001/api';
    const response = await axios.get(`${API_URL}/dashboard/stats`);

    setStats(response.data.stats);
    setChartData(response.data.chartData);
    setLoading(false);
  } catch (error) {
    console.error('Erro ao buscar dados:', error);
    setLoading(false);
  }
};

const StatCard = ({ title, value, icon, color, trend }) => (
  <Card sx={{
    height: '100%',
    background: `linear-gradient(135deg, ${color}20, ${color}10)`,
    border: `1px solid ${color}30`
  }}>
    <CardContent>
      <Box display="flex" alignItems="center" justifyContent="space-between">
        <Box>
          <Typography color="textSecondary" gutterBottom variant="h6">
            {title}
          </Typography>
          <Typography variant="h4" component="h2" sx={{ color: color, fontWeight: 'bold' }}>
            {value}
          </Typography>
          {trend && (
            <Chip
              label={`+${trend}%`}
              color="success"
              size="small"
              sx={{ mt: 1 }}
            />
          )}
        </Box>
        <Box sx={{ color: color, fontSize: 48, opacity: 0.8 }}>
          {icon}
        </Box>
      </Box>
```

```
    </CardContent>
  </Card>
);

if (loading) {
  return (
    <Container>
      <Box display="flex" justifyContent="center" alignItems="center" minHeight="400px">
        <Typography variant="h6">Carregando dashboard...</Typography>
      </Box>
    </Container>
  );
}

return (
  <Container maxWidth="xl" sx={{ py: 4 }}>
    {/* Header */}
    <Paper sx={{ p: 3, mb: 4, background: 'linear-gradient(135deg, #1976d2, #1565c0)' }}>
      <Typography variant="h3" component="h1" sx={{ color: 'white', fontWeight: 'bold' }}>
        🏢 NEXUS SST Dashboard
      </Typography>
      <Typography variant="h6" sx={{ color: 'white', opacity: 0.9, mt: 1 }}>
        Plataforma de Segurança e Saúde no Trabalho
      </Typography>
    </Paper>

    {/* Cards de Estatísticas */}
    <Grid container spacing={3} sx={{ mb: 4 }}>
      <Grid item xs={12} sm={6} md={3}>
        <StatCard
          title="Usuários Ativos"
          value={stats.totalUsers}
          icon={<PeopleIcon />}
          color="#1976d2"
          trend={12}
        />
      </Grid>
      <Grid item xs={12} sm={6} md={3}>
        <StatCard
          title="Cursos Disponíveis"
          value={stats.totalCourses}
          icon={<SchoolIcon />}
          color="#388e3c"
          trend={8}
        />
      </Grid>
      <Grid item xs={12} sm={6} md={3}>
```

```
    <StatCard
      title="Certificados Emitidos"
      value={stats.totalCertificates}
      icon={<AssignmentIcon />}
      color="#f57c00"
      trend={23}
    />
  </Grid>
  <Grid item xs={12} sm={6} md={3}>
    <StatCard
      title="Taxa de Conclusão"
      value={`${stats.completionRate}%`}
      icon={<TrendingUpIcon />}
      color="#7b1fa2"
      trend={5}
    />
  </Grid>
</Grid>

{/* Gráficos */}
<Grid container spacing={3}>
  <Grid item xs={12} md={8}>
    <Card>
      <CardContent>
        <Typography variant="h6" gutterBottom sx={{ fontWeight: 'bold' }}>
          📊 Cursos Mais Populares
        </Typography>
        <ResponsiveContainer width="100%" height={300}>
          <BarChart data={chartData.coursePopularity}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="name" />
            <YAxis />
            <Tooltip />
            <Bar dataKey="enrollments" fill="#1976d2" radius={[4, 4, 0, 0]} />
          </BarChart>
        </ResponsiveContainer>
      </CardContent>
    </Card>
  </Grid>

  <Grid item xs={12} md={4}>
    <Card>
      <CardContent>
        <Typography variant="h6" gutterBottom sx={{ fontWeight: 'bold' }}>
          🎯 Status dos Certificados
        </Typography>
        <ResponsiveContainer width="100%" height={300}>
```

```
            <PieChart>
              <Pie
                data={chartData.certificateStatus}
                dataKey="value"
                nameKey="name"
                cx="50%"
                cy="50%"
                outerRadius={80}
                fill="#8884d8"
                label
              >
                {chartData.certificateStatus?.map((entry, index) => (
                  <Cell key={`cell-${index}`} fill={entry.color} />
                ))}
              </Pie>
              <Tooltip />
            </PieChart>
          </ResponsiveContainer>
        </CardContent>
      </Card>
    </Grid>
  </Grid>

  {/* Status da API */}
  <Paper sx={{ p: 2, mt: 3, background: '#f5f5f5' }}>
    <Typography variant="body2" color="textSecondary">
      ✅ API Status: Conectado | 🌐 Ambiente: {process.env.NODE_ENV || 'Desenvolvimento'}
    </Typography>
  </Paper>
 </Container>
 );
};

export default Dashboard;
```

3.3.2 Atualizar App.js principal:

jsx

```
// frontend-admin/src/App.js
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { ThemeProvider, createTheme } from '@mui/material/styles';
import CssBaseline from '@mui/material/CssBaseline';
import Dashboard from './components/Dashboard';

const theme = createTheme({
  palette: {
    primary: {
      main: '#1976d2',
    },
    secondary: {
      main: '#dc004e',
    },
    background: {
      default: '#f5f5f5',
    },
  },
  typography: {
    fontFamily: '"Roboto", "Helvetica", "Arial", sans-serif',
  },
});

function App() {
  return (
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <Router>
        <Routes>
          <Route path="/" element={<Dashboard />} />
          <Route path="/admin" element={<Dashboard />} />
          <Route path="*" element={<Dashboard />} />
        </Routes>
      </Router>
    </ThemeProvider>
  );
}

export default App;
```

3.4 Configurar Banco de Dados no Supabase

```
sql
```

```sql
-- 1. Acesse https://supabase.com/dashboard
-- 2. Clique no seu projeto "nexus-sst-db"
-- 3. Vá em "SQL Editor"
-- 4. Cole e execute este código:

-- Criar tabelas principais
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    company_name VARCHAR(255),
    role VARCHAR(50) DEFAULT 'user',
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE courses (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    duration_hours INTEGER DEFAULT 1,
    category VARCHAR(100) DEFAULT 'SST',
    nr_reference VARCHAR(50), -- Ex: NR-35, NR-10
    status VARCHAR(50) DEFAULT 'active',
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE enrollments (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id),
    course_id UUID REFERENCES courses(id),
    status VARCHAR(50) DEFAULT 'enrolled',
    progress DECIMAL(5,2) DEFAULT 0.00,
    started_at TIMESTAMP DEFAULT NOW(),
    completed_at TIMESTAMP,
    certificate_issued BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE certificates (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    enrollment_id UUID REFERENCES enrollments(id),
    certificate_number VARCHAR(100) UNIQUE NOT NULL,
```

```
    issued_at TIMESTAMP DEFAULT NOW(),
    expires_at TIMESTAMP,
    file_path VARCHAR(500),
    status VARCHAR(50) DEFAULT 'valid'
);

-- Inserir dados de exemplo
INSERT INTO courses (title, description, duration_hours, nr_reference) VALUES
('Trabalho em Altura - NR-35', 'Curso básico sobre segurança em trabalhos em altura', 8, 'NR-35'),
('Segurança em Instalações Elétricas - NR-10', 'Treinamento sobre segurança elétrica', 40, 'NR-10'),
('Equipamentos de Proteção Individual - NR-06', 'Uso correto de EPIs', 4, 'NR-06'),
('Ergonomia - NR-17', 'Princípios de ergonomia no trabalho', 6, 'NR-17'),
('Segurança em Máquinas - NR-12', 'Operação segura de máquinas', 16, 'NR-12');

-- Criar índices para performance
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_enrollments_user_course ON enrollments(user_id, course_id);
CREATE INDEX idx_certificates_number ON certificates(certificate_number);
```

🚀 PARTE 4: CONFIGURAR VARIÁVEIS DE AMBIENTE

4.1 Backend (.env)

bash

```
# Criar arquivo .env na pasta backend:
cd backend
```

bash

```
# backend/.env
NODE_ENV=development
PORT=3001

# Supabase (copie do dashboard do Supabase)
SUPABASE_URL=https://xxx.supabase.co
SUPABASE_ANON_KEY=eyJ...
SUPABASE_SERVICE_KEY=eyJ...

# JWT
JWT_SECRET=minha-chave-super-secreta-jwt-123!

# URLs Frontend (para CORS)
FRONTEND_URLS=http://localhost:3000,https://sua-app-admin.vercel.app,https://sua-app-public.netlify.app

# Cloudinary (copie do dashboard do Cloudinary)
CLOUDINARY_CLOUD_NAME=nexus-sst
CLOUDINARY_API_KEY=123456789
CLOUDINARY_API_SECRET=abc123def456

# Resend (copie da dashboard do Resend)
RESEND_API_KEY=re_123abc...

# Email
SMTP_FROM=noreply@nexus-sst.com
```

4.2 Frontend Admin (.env)

bash

```
cd ../frontend-admin
```

bash

```
# frontend-admin/.env
REACT_APP_API_URL=http://localhost:3001/api
REACT_APP_SUPABASE_URL=https://xxx.supabase.co
REACT_APP_SUPABASE_ANON_KEY=eyJ...
```

🧪 PARTE 5: TESTAR LOCALMENTE

5.1 Testar Backend

bash

```
# Na pasta backend:
cd backend

# Instalar dependências do Supabase:
npm install @supabase/supabase-js

# Iniciar servidor:
npm run dev

# ✅ Deve aparecer:
# 🚀 NEXUS SST API rodando na porta 3001
# 📊 Environment: development

# Testar em outro terminal:
curl http://localhost:3001/api/health
# Deve retornar: {"status":"OK","timestamp":"...","service":"NEXUS SST API"}

curl http://localhost:3001/api/test
# Deve retornar: {"message":"NEXUS SST API funcionando!"}
```

5.2 Testar Frontend

bash

```
# Em outro terminal, na pasta frontend-admin:
cd frontend-admin

# Iniciar React:
npm start

# ✅ Deve abrir automaticamente: http://localhost:3000
# Você deve ver o dashboard com gráficos e estatísticas
```

🌐 PARTE 6: DEPLOY NA NUVEM (100% GRATUITO)

6.1 Preparar para Deploy

bash

```
# Na raiz do projeto:
cd ..

# Criar repositório Git:
git add .
git commit -m "🚀 Projeto NEXUS SST inicial"

# Criar repositório no GitHub:
# 1. Vá para https://github.com
# 2. Clique no "+" > "New repository"
# 3. Nome: nexus-sst-platform
# 4. Deixe público (para planos gratuitos)
# 5. Clique "Create repository"
# 6. Copie os comandos mostrados:

git remote add origin https://github.com/SEU-USUARIO/nexus-sst-platform.git
git branch -M main
git push -u origin main
```

6.2 Deploy Backend no Render

bash

```
# 1. Acesse https://render.com/dashboard
# 2. Clique "New +" > "Web Service"
# 3. Clique "Connect" no seu repositório GitHub
# 4. Configure:
#    - Name: nexus-sst-backend
#    - Root Directory: backend
#    - Environment: Node
#    - Build Command: npm install
#    - Start Command: npm start
# 5. Clique "Advanced" e adicione Environment Variables:

NODE_ENV=production
SUPABASE_URL=https://xxx.supabase.co
SUPABASE_ANON_KEY=eyJ...
JWT_SECRET=minha-chave-super-secreta-jwt-123!
FRONTEND_URLS=https://sua-app-admin.vercel.app,https://sua-app-public.netlify.app
CLOUDINARY_CLOUD_NAME=nexus-sst
CLOUDINARY_API_KEY=123456789
CLOUDINARY_API_SECRET=abc123def456
RESEND_API_KEY=re_123abc...

# 6. Clique "Create Web Service"
# 7. Aguarde 3-5 minutos para o deploy
# 8. Copie a URL gerada: https://nexus-sst-backend-xxx.onrender.com
```

6.3 Deploy Frontend Admin no Vercel

bash

```
# 1. Acesse https://vercel.com/dashboard
# 2. Clique "Add New..." > "Project"
# 3. Clique "Import" no seu repositório GitHub
# 4. Configure:
#    - Framework Preset: Create React App
#    - Root Directory: frontend-admin
#    - Build Command: npm run build
#    - Install Command: npm install
# 5. Adicione Environment Variables:
#    - REACT_APP_API_URL: https://nexus-sst-backend-xxx.onrender.com/api
#    - REACT_APP_SUPABASE_URL: https://xxx.supabase.co
#    - REACT_APP_SUPABASE_ANON_KEY: eyJ...
# 6. Clique "Deploy"
# 7. Aguarde 2-3 minutos
# 8. Copie a URL gerada: https://nexus-sst-admin-xxx.vercel.app
```

6.4 Atualizar CORS no Backend

```bash
# Volte ao Render > seu backend > Environment
# Atualize FRONTEND_URLS com as URLs reais:
FRONTEND_URLS=https://nexus-sst-admin-xxx.vercel.app,https://nexus-sst-public-xxx.netlify.app

# Clique "Save Changes"
# O Render fará redeploy automático
```

✅ PARTE 7: TESTAR PRODUÇÃO

7.1 Testar API em Produção

```bash
# Teste a API diretamente:
curl https://nexus-sst-backend-xxx.onrender.com/api/health

# Deve retornar:
# {"status":"OK","timestamp":"...","service":"NEXUS SST API"}

# Teste o endpoint de estatísticas:
curl https://nexus-sst-backend-xxx.onrender.com/api/dashboard/stats

# Deve retornar JSON com estatísticas
```

7.2 Testar Frontend

```bash
# 1. Acesse: https://nexus-sst-admin-xxx.vercel.app
# 2. Deve carregar o dashboard com:
#    - Cards de estatísticas
#    - Gráfico de barras dos cursos
#    - Gráfico de pizza dos certificados
# 3. Verifique no console do navegador (F12) se não há erros
```

🎉 RESULTADO FINAL

Parabéns! Agora você tem:

🏢 Plataforma NEXUS SST Completa:

- ✅ **Backend API**: https://nexus-sst-backend-xxx.onrender.com
- ✅ **Dashboard Admin**: https://nexus-sst-admin-xxx.vercel.app
- ✅ **Banco PostgreSQL**: Supabase (500MB gratuitos)
- ✅ **Storage**: Cloudinary (25GB gratuitos)
- ✅ **Email**: Resend (3.000 emails/mês)

💰 Custo Total: R$ 0,00/mês

⚡ Disponibilidade: 24/7

🔒 HTTPS: Automático

🌐 CDN Global: Incluído

🔄 PRÓXIMOS PASSOS

Funcionalidades para Adicionar:

1. **Sistema de Login/Registro** completo
2. **Área do Aluno** (frontend público)
3. **Geração de Certificados PDF**
4. **Upload de vídeos** para cursos
5. **Sistema de Pagamentos** (Stripe/PagSeguro)
6. **Relatórios avançados**
7. **Notificações por email**

Para Desenvolvimento Contínuo:

1. **Push para GitHub** = Deploy automático
2. **Monitoramento** via Render Dashboard
3. **Logs** disponíveis em tempo real
4. **Backups** automáticos no Supabase

Sua plataforma está **rodando 24/7 gratuitamente** e pronta para evoluir! 🚀