📘 Best Practices - TA Consulting Platform

Análise Técnica e Recomendações de Implementação

🎯 RESUMO EXECUTIVO

A plataforma TA Consulting está **tecnicamente sólida** com arquitetura moderna e funcionalidades robustas. Esta análise identifica melhorias específicas para **otimização, segurança e experiência do utilizador**.

✅ PONTOS FORTES ATUAIS

1. Arquitetura

- ✅ Next.js 14 com App Router (SSR + CSR híbrido)
- ✅ TypeScript para type safety
- ✅ PostgreSQL + Prisma ORM (migrations seguras)
- ✅ NextAuth.js (autenticação robusta)
- ✅ shadcn/ui (componentes acessíveis)

2. Automação

- ✅ 3 agentes de scraping configurados
- ✅ Agendamento via daemon management
- ✅ Sistema de email notifications
- ✅ Prevenção de duplicados

3. Base de Dados

- ✅ Schema bem estruturado
- ✅ Relações definidas corretamente
- ✅ Enums para controlo de estados
- ✅ Indexes implícitos (@unique, @id)

🔍 BEST PRACTICES - ANÁLISE DETALHADA

1. SEGURANÇA 🔐

✅ Já Implementado:

- Senhas hasheadas (bcrypt)
- NextAuth.js para autenticação
- HTTPS no deploy
- Variáveis de ambiente (.env)

⚠️ Melhorias Recomendadas:

A. Rate Limiting nas APIs

```
// middleware.ts ou nas API routes
import { RateLimiter } from '@/lib/rate-limiter';

const limiter = new RateLimiter({
  uniqueTokenPerInterval: 500,
  interval: 60000, // 1 min
});

// Aplicar em rotas sensíveis:
// - /api/auth/*
// - /api/avisos
// - /api/candidaturas
```

B. Validação de Input (Zod)

```
// Em TODAS as API routes
import { z } from 'zod';

const createAvisoSchema = z.object({
  nome: z.string().min(3).max(200),
  portal: z.enum(['PORTUGAL2030', 'PAPAC', 'PRR']),
  // ... outros campos
});

// Validar antes de inserir na DB
const validated = createAvisoSchema.parse(body);
```

C. CORS Seguro

```
// next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/api/:path*',
        headers: [
          { key: 'Access-Control-Allow-Origin', value: 'https://ta-consulting-platfo-
tfdltj.abacusai.app' },
          { key: 'Access-Control-Allow-Methods', value: 'GET,POST,PUT,DELETE' },
        ],
      },
    ];
  },
};
```

D. Sanitização de Dados

```
// Sanitizar HTML em campos de texto livre
import DOMPurify from 'isomorphic-dompurify';

const cleanDescription = DOMPurify.sanitize(aviso.descricao);
```

---

2. PERFORMANCE ⚡

✅ Já Implementado:

- Next.js com otimizações automáticas
- PostgreSQL (rápido)
- Prisma (queries otimizadas)

⚠️ Melhorias Recomendadas:

A. Paginação em Listagens

```
// api/avisos/route.ts
const avisos = await prisma.aviso.findMany({
  skip: (page - 1) * limit,
  take: limit,
  orderBy: { dataFimSubmissao: 'asc' },
});

// Retornar também o total
const total = await prisma.aviso.count();
```

B. Caching (Redis ou Next.js)

```
// Para avisos que não mudam frequentemente
import { unstable_cache } from 'next/cache';

export const getAvisos = unstable_cache(
  async () => {
    return await prisma.aviso.findMany();
  },
  ['avisos-list'],
  {
    revalidate: 3600, // 1 hora
    tags: ['avisos'],
  }
);

// Invalidar quando houver updates
import { revalidateTag } from 'next/cache';
revalidateTag('avisos');
```

C. Indexes na Base de Dados

```
// prisma/schema.prisma
model Aviso {
  // ...campos existentes

  @@index([portal, ativo])
  @@index([dataFimSubmissao])
  @@index([urgente])
}

model Empresa {
  @@index([setor])
  @@index([dimensao])
  @@index([regiao])
}
```

D. Loading States e Skeleton UI

```
// Melhorar UX durante loading
import { Skeleton } from '@/components/ui/skeleton';

function AvisosList() {
  const { data, isLoading } = useQuery(['avisos']);

  if (isLoading) {
    return (
      <div className="space-y-4">
        {[...Array(5)].map((_, i) => (
          <Skeleton key={i} className="h-20 w-full" />
        ))}
      </div>
    );
  }

  return // ... lista real
}
```

---

3. UX/UI 🎨

✅ Já Implementado:

- shadcn/ui (design consistente)
- Tailwind CSS (responsivo)
- Tema dark/light

⚠️ Melhorias Recomendadas:

A. Feedback Visual Imediato

```
// Usar Toasts para todas as ações
import { toast } from 'sonner';

// Sucesso
toast.success('Aviso criado com sucesso!');

// Erro
toast.error('Erro ao criar aviso. Tente novamente.');

// Loading
const loading = toast.loading('Criando aviso...');
// ... depois
toast.success('Criado!', { id: loading });
```

B. Estados Vazios (Empty States)

```
// Quando não há dados
function EmptyState({ title, description, action }: EmptyStateProps) {
  return (
    <div className="text-center py-12">
      <FileX className="mx-auto h-12 w-12 text-muted-foreground" />
      <h3 className="mt-4 text-lg font-semibold">{title}</h3>
      <p className="mt-2 text-sm text-muted-foreground">{description}</p>
      {action && (
        <Button onClick={action.onClick} className="mt-4">
          {action.label}
        </Button>
      )}
    </div>
  );
}
```

C. Atalhos de Teclado

```
// Para power users
useEffect(() => {
  const handleKeyDown = (e: KeyboardEvent) => {
    // Ctrl/Cmd + K → Abrir search
    if ((e.ctrlKey || e.metaKey) && e.key === 'k') {
      e.preventDefault();
      openSearch();
    }

    // Ctrl/Cmd + N → Novo aviso
    if ((e.ctrlKey || e.metaKey) && e.key === 'n') {
      e.preventDefault();
      openNewAvisoModal();
    }
  };

  document.addEventListener('keydown', handleKeyDown);
  return () => document.removeEventListener('keydown', handleKeyDown);
}, []);
```

D. Acessibilidade (a11y)

```
// ARIA labels
<button aria-label="Fechar modal">
  <X />
</button>

// Navegação por teclado
<div role="dialog" aria-modal="true">

// Focus management
useEffect(() => {
  inputRef.current?.focus();
}, []);
```

---

4. CÓDIGO 💻

✅ Já Implementado:

- TypeScript
- Estrutura organizada
- Componentes reutilizáveis

⚠️ Melhorias Recomendadas:

A. Custom Hooks Reutilizáveis

```
// hooks/useAvisos.ts
export function useAvisos(filters?: AvisoFilters) {
  return useQuery({
    queryKey: ['avisos', filters],
    queryFn: () => fetchAvisos(filters),
    staleTime: 5 * 60 * 1000, // 5 min
  });
}

// hooks/useAuth.ts
export function useAuth() {
  const { data: session } = useSession();

  return {
    user: session?.user,
    isAdmin: session?.user?.role === 'admin',
    isAuthenticated: !!session,
  };
}
```

B. Error Boundaries

```
// components/error-boundary.tsx
class ErrorBoundary extends React.Component {
  state = { hasError: false };

  static getDerivedStateFromError() {
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    console.error('Error:', error, errorInfo);
    // Enviar para Sentry/LogRocket
  }

  render() {
    if (this.state.hasError) {
      return <ErrorFallback />;
    }
    return this.props.children;
  }
}
```

C. Logging Estruturado

```typescript
// lib/logger.ts
export const logger = {
  info: (message: string, meta?: any) => {
    console.log(`[INFO] ${message}`, meta);
    // Enviar para serviço externo (Logtail, DataDog)
  },
  error: (message: string, error: Error, meta?: any) => {
    console.error(`[ERROR] ${message}`, error, meta);
    // Enviar para Sentry
  },
  warn: (message: string, meta?: any) => {
    console.warn(`[WARN] ${message}`, meta);
  },
};

// Uso
logger.info('Aviso criado', { avisoId: aviso.id });
logger.error('Falha ao criar aviso', error, { userId: user.id });
```

D. Testes

```typescript
// __tests__/avisos.test.ts
describe('Avisos API', () => {
  it('should create aviso with valid data', async () => {
    const response = await POST('/api/avisos', validData);
    expect(response.status).toBe(201);
    expect(response.body).toHaveProperty('id');
  });

  it('should reject invalid data', async () => {
    const response = await POST('/api/avisos', invalidData);
    expect(response.status).toBe(400);
  });
});
```

---

5. DEVOPS & MONITORING 🔧

⚠️ A Implementar:

A. Environment Variables Documentation

```bash
# .env.example
DATABASE_URL="postgresql://user:password@localhost:5432/ta_consulting"
NEXTAUTH_SECRET="generate-with-openssl-rand-base64-32"
NEXTAUTH_URL="http://localhost:3000"
EMAIL_USER="your-gmail@gmail.com"
EMAIL_PASSWORD="your-app-specific-password"
```

B. Health Check Endpoint

```
// app/api/health/route.ts
export async function GET() {
  try {
    // Verificar conexão DB
    await prisma.$queryRaw`SELECT 1`;

    return NextResponse.json({
      status: 'healthy',
      timestamp: new Date().toISOString(),
      services: {
        database: 'connected',
        email: 'operational',
      },
    });
  } catch (error) {
    return NextResponse.json(
      { status: 'unhealthy', error: error.message },
      { status: 503 }
    );
  }
}
```

C. Monitoring & Alerts

```
// Integrar com Sentry
import * as Sentry from '@sentry/nextjs';

Sentry.init({
  dsn: process.env.SENTRY_DSN,
  environment: process.env.NODE_ENV,
  tracesSampleRate: 0.1,
});

// Capturar erros
try {
  // código
} catch (error) {
  Sentry.captureException(error);
  throw error;
}
```

D. Backup Automation

```
# scripts/backup-db.sh
#!/bin/bash
pg_dump $DATABASE_URL > backups/db-$(date +%Y%m%d-%H%M%S).sql
# Upload para S3 ou Google Cloud Storage
```

🎯 ROADMAP DE IMPLEMENTAÇÃO

FASE 1: CRÍTICA (Antes da Demo)

- ✅ Popular base de dados com dados reais
- ✅ Sincronizar código com GitHub

- ⚠️ Criar .env.example
- ⚠️ Adicionar rate limiting básico
- ⚠️ Implementar toasts de feedback

FASE 2: ALTA PRIORIDADE (Esta Semana)

- 🤖 **Agente IA integrado**
- 📊 Sistema de recomendações
- 📱 Notificações push
- 📈 Analytics dashboard

FASE 3: MÉDIA PRIORIDADE (Próximas 2 Semanas)

- 🔄 Workflow engine customizável
- 📄 Sistema de templates
- 🔍 Search avançado
- 📊 Relatórios exportáveis (PDF/Excel)

FASE 4: OTIMIZAÇÕES (Contínuo)

- ⚡ Performance tuning
- 🔐 Security hardening
- 🧪 Testes automatizados
- 📊 Monitoring & observability

📊 MÉTRICAS DE SUCESSO

Performance:

- Time to First Byte (TTFB) < 200ms
- Largest Contentful Paint (LCP) < 2.5s
- First Input Delay (FID) < 100ms

Disponibilidade:

- Uptime > 99.9%
- Mean Time To Recovery (MTTR) < 1h

Usabilidade:

- Task Success Rate > 95%
- Average Session Duration > 5min
- Return Rate > 70%

✅ CONCLUSÃO

A plataforma TA Consulting tem uma **base técnica excelente**. As melhorias propostas vão:

1. **Aumentar a segurança** (rate limiting, validação)
2. **Melhorar a performance** (caching, pagination)
3. **Otimizar a UX** (feedback visual, estados vazios)

4. **Facilitar manutenção** (testes, logging)
5. **Preparar para escala** (monitoring, backups)

Recomendação: Implementar **FASE 1** imediatamente e **FASE 2** esta semana para máximo impacto na demo com o cliente.

Documento criado em: 5 de Novembro de 2025
Versão: 1.0
Autor: TA Consulting Platform - DeepAgent Analysis