

TODO - Timming LoveU

Lista de tarefas para melhorias e implementações.

Prioridade Alta (Essencial para Produção)

1. Sistema de Pagamento Stripe

Status:  Não Implementado

Estimativa: 2-3 dias

Responsável: -

- ☐ Instalar dependências Stripe
- ☐ Configurar conta Stripe e obter chaves
- ☐ Criar produtos/preços no Stripe Dashboard
- ☐ Atualizar schema.prisma com UserSubscription
- ☐ Implementar API routes de pagamento
- ☐ /api/payment/plans
- ☐ /api/payment/create-checkout
- ☐ /api/payment/webhook
- ☐ /api/payment/portal
- ☐ /api/payment/subscription-status
- ☐ Criar página de planos (/pricing)
- ☐ Adicionar badge de status no dashboard
- ☐ Implementar validação de plano ativo
- ☐ Testar fluxo completo (checkout → webhook → atualização DB)
- ☐ Configurar webhook em produção
- ☐ Documentar processo

Referência: Ver `PAYMENT_IMPLEMENTATION_GUIDE.md`

2. Migrar Upload de Mídia para Cloud Storage

Status:  Não Implementado

Estimativa: 1-2 dias

Responsável: -

Opções:

- AWS S3
- Cloudinary
- DigitalOcean Spaces

Tarefas:

- ☐ Escolher provedor de storage
- ☐ Criar conta e obter credenciais

- [] Instalar SDK (ex: `aws-sdk` ou `cloudinary`)
- [] Atualizar `/api/media/upload` para usar cloud storage
- [] Atualizar schema para armazenar URLs completas
- [] Migrar arquivos existentes (script de migração)
- [] Atualizar componentes para usar novas URLs
- [] Testar upload/download
- [] Configurar CDN (opcional)
- [] Atualizar backup strategy

Benefícios:

- ☒ Escalabilidade
- ☒ CDN para performance
- ☒ Backup automático
- ☒ Transformação de imagens

3. Aplicar Rate Limiting

Status: ⚠ Código existe mas não aplicado

Estimativa: 4-6 horas

Responsável: -

Rotas para proteger:

- [] POST `/api/signup` (5 req/15min por IP)
- [] POST `/api/auth/[...nextauth]` (10 req/15min por IP)
- [] POST `/api/media/upload` (20 req/hour por usuário)
- [] POST `/api/contact` (3 req/15min por IP)
- [] POST `/api/payment/create-checkout` (5 req/15min por usuário)

Implementação:

```
// Usar lib/rate-limit.ts existente
import { rateLimit } from '@lib/rate-limit'

export async function POST(req: NextRequest) {
  // Aplicar rate limit
  const rateLimitResult = await rateLimit(req)
  if (!rateLimitResult.success) {
    return NextResponse.json(
      { error: 'Too many requests' },
      { status: 429 }
    )
  }

  // ... resto da lógica
}
```

Prioridade Média (Importante)

4. Sistema de Email

Status: ⏳ Não Implementado

Estimativa: 1-2 dias

Responsável: -

Funcionalidades:

- ☐ Verificação de email no cadastro
- ☐ Reset de senha
- ☐ Confirmação de pagamento
- ☐ Alerta de expiração de plano
- ☐ Falha de pagamento
- ☐ Newsletter (opcional)

Providers sugeridos:

- SendGrid
- Resend
- AWS SES

Tarefas:

- ☐ Escolher provider
 - ☐ Configurar conta
 - ☐ Instalar SDK
 - ☐ Criar templates de email
 - ☐ Implementar envio
 - ☐ Implementar verificação de email
 - ☐ Implementar reset de senha
 - ☐ Testar em sandbox
 - ☐ Configurar em produção
-

5. Implementar Caching com Redis

Status: ⚠️ Redis configurado no Docker mas não utilizado

Estimativa: 1 dia

Responsável: -

O que cachear:

- ☐ Páginas públicas de casais (ISR)
- ☐ Stats do dashboard (5 min)
- ☐ Lista de planos (1 hora)
- ☐ User session data
- ☐ Media metadata

Implementação:

```
// lib/redis.ts
import { Redis } from 'ioredis'

const redis = new Redis(process.env.REDIS_URL!)

export async function getCache<T>({
  key: string,
  fetcher: () => Promise<T>,
  ttl = 300 // 5 min
}): Promise<T> {
  const cached = await redis.get(key)
  if (cached) return JSON.parse(cached)

  const data = await fetcher()
  await redis.setex(key, ttl, JSON.stringify(data))
  return data
}
```

6. Melhorar Cobertura de Testes

Status: ⚠ Cobertura limitada

Estimativa: 2-3 dias

Responsável: -

Meta: 80%+ de cobertura

Áreas a testar:

- [] API Routes
- [] /api/auth/signup
- [] /api/couple/
- [] /api/media/
- [] /api/payment/*
- [] Componentes do Dashboard
- [] Helpers e utils
- [] Validações (Zod schemas)
- [] Webhook handler

Tipos de testes:

- [] Testes unitários (Jest)
- [] Testes de integração (API routes)
- [] Testes E2E (Playwright)
- [] Testes de acessibilidade

Setup E2E:

```
npm install -D @playwright/test
npx playwright install
```

Prioridade Baixa (Nice to Have)

7. Logging e Monitoring

Status: 🕒 Não Implementado

Estimativa: 1 dia

Responsável: -

Logging:

- ☐ Instalar Winston ou Pino
- ☐ Criar logger centralizado
- ☐ Log de requests (API routes)
- ☐ Log de erros
- ☐ Log de eventos de negócio (pagamentos, uploads)
- ☐ Rotação de logs

Error Tracking:

- ☐ Configurar Sentry
- ☐ Capturar erros frontend
- ☐ Capturar erros backend
- ☐ Source maps em produção

Analytics:

- ☐ Google Analytics ou Mixpanel
 - ☐ Posthog (opcional)
 - ☐ Eventos customizados:
 - Signup
 - Login
 - Criação de página
 - Upload de mídia
 - Início de assinatura
 - Cancelamento
-

8. Backup Automático

Status: 🕒 Não Implementado

Estimativa: 4 horas

Responsável: -

Tarefas:

- ☐ Script de backup do PostgreSQL
- ☐ Cron job para backup diário
- ☐ Backup de uploads (se ainda local)
- ☐ Upload para S3 ou similar
- ☐ Retenção de 30 dias
- ☐ Testar restore
- ☐ Documentar processo

Script de Exemplo:

```
#!/bin/bash
# scripts/backup-db.sh

DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="backup_${DATE}.sql.gz"

pg_dump $DATABASE_URL | gzip > /backups/$BACKUP_FILE

# Upload para S3
aws s3 cp /backups/$BACKUP_FILE s3://my-bucket/backups/

# Limpar backups antigos (> 30 dias)
find /backups -mtime +30 -delete
```

9. Documentação de API

Status: 🕒 Não Implementado

Estimativa: 1 dia

Responsável: -

Opções:

- Swagger/OpenAPI
- Redoc
- Postman Collection

Tarefas:

- [] Documentar todos os endpoints
- [] Adicionar exemplos de request/response
- [] Documentar erros possíveis
- [] Documentar autenticação
- [] Publicar docs (ex: Swagger UI)
- [] Manter atualizado

10. Admin Dashboard

Status: 🕒 Não Implementado

Estimativa: 3-5 dias

Responsável: -

Funcionalidades:

- [] Lista de usuários
- [] Filtros e busca
- [] Ver detalhes de usuário
- [] Gerenciar assinaturas manualmente
- [] Ver estatísticas gerais
- Total de usuários
- Usuários ativos
- MRR (Monthly Recurring Revenue)
- Churn rate

- [] Ver logs de pagamento
- [] Enviar emails para usuários

Rota: /admin

Proteção:

```
// middleware
const session = await getServerSession(authOptions)
if (!session?.user?.isAdmin) {
  redirect('/dashboard')
}
```

11. PWA (Progressive Web App)

Status: ⏳ Não Implementado

Estimativa: 2 dias

Responsável: -

Tarefas:

- [] Adicionar manifest.json
- [] Service Worker
- [] Ícones de app (vários tamanhos)
- [] Splash screens
- [] Funcionar offline (básico)
- [] Push notifications (opcional)

Benefícios:

- ☒ Instalável no celular
- ☒ Funciona offline
- ☒ Mais rápido

12. Internacionalização (i18n)

Status: ⏳ Não Implementado

Estimativa: 3-4 dias

Responsável: -

Idiomas:

- Português (atual)
- Inglês
- Espanhol (opcional)

Tarefas:

- [] Instalar `next-intl` ou `i18next`
- [] Extrair strings para arquivos de tradução
- [] Criar traduções
- [] Atualizar componentes para usar traduções
- [] Seletor de idioma na UI
- [] Detectar idioma do browser

13. Melhorias de UX/UI

Status:  Contínuo

Estimativa: -

Responsável: -

Ideias:

- [] Loading skeletons em vez de spinners
- [] Animações mais suaves (Framer Motion)
- [] Toast notifications melhores
- [] Tutorial interativo para novos usuários
- [] Empty states melhores
- [] Confirmações antes de ações destrutivas
- [] Modo escuro (já tem ThemeProvider)

14. SEO Optimization

Status:  Básico implementado


Estimativa: 1-2 dias

Responsável: -

Tarefas:

- [] Meta tags dinâmicas por página
- [] Open Graph tags
- [] Twitter Cards
- [] Schema.org markup (JSON-LD)
- [] Sitemap.xml
- [] robots.txt
- [] Canonical URLs
- [] Performance (Lighthouse score > 90)

15. Referral Program (Indicação)

Status:  Não Implementado

Estimativa: 2-3 dias

Responsável: -

Funcionalidade:

- Usuário ganha link de indicação
- Indicado ganha desconto
- Indicador ganha desconto/crédito

Tarefas:

- [] Criar tabela Referral
- [] Gerar código único por usuário
- [] Página de indicação
- [] Tracking de conversões

- [] Dashboard de indicações
- [] Recompensas (desconto ou crédito)



Métricas e KPIs para Implementar

Status: ⏳ Não Implementado

Estimativa: 2 dias

Responsável: -

Métricas de Negócio

- [] Conversão (signup → criação de página → assinatura)
- [] MRR (Monthly Recurring Revenue)
- [] ARR (Annual Recurring Revenue)
- [] Churn rate
- [] ARPU (Average Revenue Per User)
- [] LTV (Lifetime Value)
- [] CAC (Customer Acquisition Cost)

Métricas de Produto

- [] DAU/MAU (Daily/Monthly Active Users)
- [] Tempo médio na plataforma
- [] Uploads por usuário
- [] Views de páginas públicas
- [] Taxa de compartilhamento

Dashboard Analytics

- [] Criar página /admin/analytics
- [] Gráficos de crescimento
- [] Funil de conversão
- [] Cohort analysis



Melhorias Técnicas

16. Otimizações de Performance

Status: ⚠️ Parcial

Estimativa: 2-3 dias

Responsável: -

Database:

- [] Adicionar índices em queries frequentes
- [] Usar `select` específico (não trazer tudo)
- [] Connection pooling (PgBouncer)
- [] Read replicas (se necessário)

Frontend:

- [] Code splitting otimizado
- [] Lazy loading de componentes pesados
- [] Otimizar bundle size (análise com `@next/bundle-analyzer`)
- [] Prefetch de páginas importantes
- [] Otimizar imagens (next/image)

Queries:

```
// ❌ Evitar
const user = await prisma.user.findUnique({ where: { id } })

// ✅ Melhor
const user = await prisma.user.findUnique({
  where: { id },
  select: {
    id: true,
    email: true,
    firstName: true,
    lastName: true,
    planoAtivo: true
  }
})
```

17. Segurança Adicional

Status: ⚠️ Básico implementado**Estimativa:** 1-2 dias**Responsável:** -**Tarefas:**

- [] Implementar CSRF protection
- [] Helmet.js para headers adicionais
- [] Validar todos os inputs (sanitização)
- [] Limitar tamanho de uploads
- [] Scan de vulnerabilidades (npm audit)
- [] Dependabot para updates
- [] Secrets scanning no Git
- [] 2FA (Two-Factor Authentication)

18. CI/CD Pipeline

Status: ⚠️ Scripts básicos existem**Estimativa:** 1 dia**Responsável:** -**GitHub Actions:**

```
# .github/workflows/ci.yml
name: CI

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
      - run: npm install
      - run: npm run type-check
      - run: npm run lint
      - run: npm test
      - run: npm run build
```

Tarefas:

- [] Criar workflow de CI
- [] Testes automáticos em PRs
- [] Deploy automático (prod/staging)
- [] Notificações no Slack/Discord
- [] Deploy preview (Vercel)



Roadmap Sugerido

Sprint 1 (Semana 1)

- Análise de arquitetura completa
- Implementar sistema de pagamento Stripe

Sprint 2 (Semana 2)

- Migrar uploads para cloud storage
- Aplicar rate limiting
- Sistema de email básico

Sprint 3 (Semana 3)

- Implementar caching com Redis
- Melhorar cobertura de testes
- Logging e monitoring

Sprint 4 (Semana 4)

- Backup automático
- Documentação de API
- SEO optimization

Sprint 5+ (Backlog)

- Admin dashboard
- PWA
- i18n

- Referral program
- Melhorias de UX/UI



Notas

Dependências Entre Tarefas

1. **Sistema de Pagamento** deve ser feito antes de:
 - Validação de plano em rotas
 - Admin dashboard (gerenciamento de assinaturas)
2. **Cloud Storage** deve ser feito antes de:
 - Otimizações de performance (CDN)
 - Backup strategy
3. **Logging** deve ser feito antes de:
 - Monitoring avançado
 - Alertas automáticos

Estimativas Totais

Prioridade Alta: ~4-6 dias

Prioridade Média: ~6-8 dias

Prioridade Baixa: ~15-20 dias

Total para MVP Completo: ~2-3 semanas de desenvolvimento focado



Checklist de Deploy em Produção

Antes de lançar:

- ☐ Sistema de pagamento funcionando
- ☐ Cloud storage configurado
- ☐ Rate limiting aplicado
- ☐ Email transacional funcionando
- ☐ Backup automático configurado
- ☐ Monitoring e alertas configurados
- ☐ Testes E2E passando
- ☐ Performance otimizada (Lighthouse > 85)
- ☐ Segurança revisada
- ☐ Documentação atualizada
- ☐ Legal (Termos de Uso, Política de Privacidade)

Última atualização: 23/10/2024

Responsável pela lista: DeepAgent - Abacus.AI