

Guia de Implementação - Dashboard e Upload de Mídia

✓ Funcionalidades Implementadas

1. Dashboard do Usuário Completo

- ✓ Página de dashboard acessível após login (/dashboard)
- ✓ Visualização de dados do casal (nomes, data de início, foto, mensagem)
- ✓ Estatísticas do relacionamento em tempo real:
 - Tempo juntos (anos, meses, dias, horas, minutos, segundos)
 - Total de memórias (fotos e vídeos)
 - Visualizações da página
- ✓ Marcos importantes (milestones):
 - Conquistas alcançadas
 - Próximos marcos
 - Progresso visual
- ✓ Navegação intuitiva com abas (Visão Geral, Estatísticas, Galeria, Marcos)
- ✓ Design responsivo com tema romântico
- ✓ Menu de usuário com opções de navegação e logout

2. Sistema de Upload de Mídia

- ✓ Interface para upload de imagens e vídeos
- ✓ Validação de tipos de arquivo:
 - Imagens: JPEG, PNG, GIF, WebP (até 10MB)
 - Vídeos: MP4, WebM, MOV (até 50MB)
- ✓ Armazenamento seguro em /public/uploads/
- ✓ Integração com banco de dados (tabela Media)
- ✓ Galeria para visualizar mídias:
 - Filtro por tipo (todas, imagens, vídeos)
 - Preview de imagens e player de vídeo
 - Informações (título, descrição, data)
 - Opção de deletar
- ✓ Associação das mídias ao perfil do casal

Estrutura de Arquivos Criados/Modificados

Banco de Dados

```
prisma/schema.prisma
- Adicionado modelo Media
- Relações com User e CouplePage
```



API Routes

app/api/media/upload/route.ts	# Upload de mídia
app/api/media/list/route.ts	# Listagem de mídias
app/api/media/delete/route.ts	# Deleção de mídia
app/api/couple/stats/route.ts	# Estatísticas do relacionamento

Componentes do Dashboard

app/dashboard/		
 page.tsx		# Página principal (server component)
 components/		
 dashboard-client.tsx		# Componente cliente principal
components/dashboard/		
 couple-info.tsx		# Informações do casal
 relationship-stats.tsx		# Estatísticas do relacionamento
 milestones.tsx		# Marcos importantes
 media-upload.tsx		# Componente de upload
 media-gallery.tsx		# Galeria de mídias

Diretórios de Upload

public/uploads/	
 images/	# Imagens enviadas
 videos/	# Vídeos enviados



Como Executar

Pré-requisitos

- Node.js 18+
- PostgreSQL
- Variáveis de ambiente configuradas no `.env`

Instalação

1. Instalar dependências:

```
cd /home/ubuntu/timming_loveu
npm install
```

1. Gerar Prisma Client:

```
npx prisma generate
```

1. Sincronizar banco de dados:

```
npx prisma db push
```

1. Iniciar servidor de desenvolvimento:

```
npm run dev
```

1. Acessar aplicação:

- Página inicial: <http://localhost:3000>
- Dashboard: <http://localhost:3000/dashboard> (requer login)
- Login: <http://localhost:3000/login>

Funcionalidades do Dashboard

Visão Geral

- Card com informações do casal (banner, nomes, mensagem, data de início)
- Botões para compartilhar e visualizar página pública
- Estatísticas em tempo real
- Resumo dos marcos importantes

Estatísticas

- Tempo juntos detalhado (anos, meses, dias)
- Contador em tempo real (horas, minutos, segundos)
- Total de memórias (fotos e vídeos)
- Cards visuais e interativos

Galeria de Mídia

- Botão para enviar novas mídias
- Filtro por tipo (todas/imagens/vídeos)
- Grade responsiva de mídias
- Preview de imagens e player de vídeo integrado
- Informações de cada mídia (título, descrição, data)
- Opção de deletar mídias

Marcos Importantes

- Lista de conquistas alcançadas
- Próximos marcos a serem celebrados
- Contagem regressiva
- Barra de progresso visual

Design Responsivo

O dashboard é totalmente responsivo e adapta-se a diferentes tamanhos de tela:

- **Mobile:** Layout em coluna única, abas compactas
- **Tablet:** Layout em 2 colunas onde aplicável
- **Desktop:** Layout completo com múltiplas colunas

Segurança

- Autenticação obrigatória via NextAuth
- Validação de tipos e tamanhos de arquivo
- Verificação de ownership antes de deletar

- Proteção contra acesso não autorizado
- Armazenamento seguro de arquivos

Próximos Passos (Opcional)

1. Otimizações:

- Compressão de imagens no upload
- Lazy loading de mídias na galeria
- Paginação para grandes quantidades de mídia

2. Funcionalidades Extras:

- Edição de título e descrição de mídias
- Reordenação de mídias na galeria
- Álbuns ou categorias de mídias
- Download de mídias
- Compartilhamento individual de mídias

3. Melhorias de UX:

- Drag & drop para upload
- Upload múltiplo de arquivos
- Barra de progresso no upload
- Preview antes do upload



Notas Técnicas

Tecnologias Utilizadas

- **Framework:** Next.js 14 (App Router)
- **Linguagem:** TypeScript
- **ORM:** Prisma
- **Banco de Dados:** PostgreSQL
- **Autenticação:** NextAuth
- **UI:** Tailwind CSS + Radix UI
- **Ícones:** Lucide React
- **Notificações:** Sonner (toast)
- **Formatação de Datas:** date-fns

Considerações Importantes

1. Os arquivos são armazenados localmente em `/public/uploads/`
2. Para produção, considere usar serviço de storage (S3, Cloudinary, etc.)
3. O Prisma Client deve ser regenerado após mudanças no schema
4. Certifique-se de que o diretório `public/uploads` tem permissões adequadas



Recursos Implementados

- ☒ Upload de imagens (JPEG, PNG, GIF, WebP)
- ☒ Upload de vídeos (MP4, WebM, MOV)
- ☒ Validação de tipo e tamanho
- ☒ Preview de mídia antes do envio

- ☒ Campos opcionais (título, descrição, data do evento)
- ☒ Galeria com filtros
- ☒ Visualização de imagens (Next.js Image)
- ☒ Player de vídeo integrado
- ☒ Deleção de mídias (com confirmação)
- ☒ Contador de tempo em tempo real
- ☒ Cálculo automático de marcos
- ☒ Interface responsiva
- ☒ Tema romântico consistente
- ☒ Navegação intuitiva
- ☒ Integração completa com autenticação



Troubleshooting

Problema: Prisma Client não inicializa

Solução:

```
rm -rf node_modules/.prisma
rm -rf .next
npx prisma generate
npm run dev
```

Problema: Erro ao fazer upload

Solução:

- Verifique permissões do diretório `/public/uploads/`
- Confirme que o tamanho do arquivo está dentro do limite
- Verifique se o tipo de arquivo é suportado

Problema: Dashboard não carrega dados

Solução:

- Verifique se o usuário está autenticado
- Confirme se existe uma CouplePage para o usuário
- Verifique logs do servidor para erros da API

Desenvolvido com  para Timming LoveU