📊 Sistema de Agendamento de Resumos (Digests)

✅ Estado Atual da Implementação

Implementado ✅

- ✅ **Modelo de preferências** completo no banco de dados
- ✅ **API endpoints** para gerenciar preferências
- ✅ **UI completa** em `/settings/notifications`
- ✅ **Serviço de geração de resumos** (`generateUserDigest`)
- ✅ **Função de busca de usuários** para cada período (`getUsersForDigest`)
- ✅ **Lógica de verificação** de preferências antes de enviar

Pendente ⏳

- ⏳ **Cron jobs** ou sistema de agendamento
- ⏳ **Envio real de emails** (aguarda configuração SMTP)
- ⏳ **Templates de email** para resumos

🔧 Como Configurar o Agendamento

Opção 1: Cron Jobs (Linux/Unix) 🐧

1. Criar script de execução

Crie o arquivo `/home/ubuntu/organizen/nextjs_space/scripts/send-digests.ts` :

```
import { prisma } from '../lib/db';
import { getUsersForDigest, generateUserDigest } from '../lib/notification-service';

async function sendDigests() {
  const now = new Date();
  const currentTime = `${now.getHours().toString().padStart(2, '0')}:00`;
  const currentDayOfWeek = now.getDay(); // 0-6
  const currentDayOfMonth = now.getDate(); // 1-31

  console.log(`🕐 Verificando resumos para ${currentTime}...`);

  // Buscar usuários para cada tipo de resumo
  const dailyUsers = await getUsersForDigest('daily', currentTime, currentDayOfWeek, c
urrentDayOfMonth);
  const weeklyUsers = await getUsersForDigest('weekly', currentTime,
currentDayOfWeek, currentDayOfMonth);
  const monthlyUsers = await getUsersForDigest('monthly', currentTime, currentDay-
OfWeek, currentDayOfMonth);

  console.log(`📊 Resumos a enviar:`);
  console.log(`   Diário: ${dailyUsers.length} usuários`);
  console.log(`   Semanal: ${weeklyUsers.length} usuários`);
  console.log(`   Mensal: ${monthlyUsers.length} usuários`);

  // Processar resumos diários
  for (const userId of dailyUsers) {
    try {
      const digest = await generateUserDigest(userId, 'daily');
      const user = await prisma.user.findUnique({
        where: { id: userId },
        select: { email: true, name: true },
      });

      if (user?.email) {
        // TODO: Enviar email com o resumo
        console.log(`📧 Resumo diário enviado para ${user.email}`);
        // await sendDigestEmail(user.email, user.name, digest);
      }
    } catch (error) {
      console.error(`Erro ao enviar resumo diário para ${userId}:`, error);
    }
  }

  // Processar resumos semanais
  for (const userId of weeklyUsers) {
    try {
      const digest = await generateUserDigest(userId, 'weekly');
      const user = await prisma.user.findUnique({
        where: { id: userId },
        select: { email: true, name: true },
      });

      if (user?.email) {
        // TODO: Enviar email com o resumo
        console.log(`📧 Resumo semanal enviado para ${user.email}`);
        // await sendDigestEmail(user.email, user.name, digest);
      }
    } catch (error) {
      console.error(`Erro ao enviar resumo semanal para ${userId}:`, error);
    }
  }
```

```
  // Processar resumos mensais
  for (const userId of monthlyUsers) {
    try {
      const digest = await generateUserDigest(userId, 'monthly');
      const user = await prisma.user.findUnique({
        where: { id: userId },
        select: { email: true, name: true },
      });

      if (user?.email) {
        // TODO: Enviar email com o resumo
        console.log(`📧 Resumo mensal enviado para ${user.email}`);
        // await sendDigestEmail(user.email, user.name, digest);
      }
    } catch (error) {
      console.error(`Erro ao enviar resumo mensal para ${userId}:`, error);
    }
  }

  console.log('✅ Processamento de resumos concluído!');
  process.exit(0);
}

sendDigests().catch((error) => {
  console.error('Erro ao enviar resumos:', error);
  process.exit(1);
});
```

2. Configurar crontab

Execute `crontab -e` e adicione:

```
# Executar a cada hora (verificar se há resumos para enviar)
0 * * * * cd /home/ubuntu/organizen/nextjs_space && NODE_ENV=production tsx scripts/
send-digests.ts >> /var/log/organizen-digests.log 2>&1
```

Explicação:

- `0 * * * *` - Executa no minuto 0 de cada hora (00:00, 01:00, 02:00, ...)
- O script verifica se há usuários configurados para receber resumos naquele horário
- Logs são salvos em `/var/log/organizen-digests.log`

Opção 2: Vercel Cron Jobs (Serverless) ☁️

Se estiver usando Vercel para deploy:

1. Criar API endpoint

`/app/api/cron/send-digests/route.ts`:

```
import { NextResponse } from 'next/server';
import { getUsersForDigest, generateUserDigest } from '@/lib/notification-service';
import { prisma } from '@/lib/db';

export async function GET(request: Request) {
  // Verificar token de autenticação do Vercel Cron
  const authHeader = request.headers.get('authorization');
  if (authHeader !== `Bearer ${process.env.CRON_SECRET}`) {
    return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
  }

  const now = new Date();
  const currentTime = `${now.getHours().toString().padStart(2, '0')}:00`;
  const currentDayOfWeek = now.getDay();
  const currentDayOfMonth = now.getDate();

  // Buscar usuários
  const dailyUsers = await getUsersForDigest('daily', currentTime, currentDayOfWeek, c
urrentDayOfMonth);
  const weeklyUsers = await getUsersForDigest('weekly', currentTime,
currentDayOfWeek, currentDayOfMonth);
  const monthlyUsers = await getUsersForDigest('monthly', currentTime, currentDay-
OfWeek, currentDayOfMonth);

  const results = {
    daily: 0,
    weekly: 0,
    monthly: 0,
  };

  // Processar resumos (similar ao script acima)
  // ...

  return NextResponse.json({
    success: true,
    timestamp: new Date().toISOString(),
    results,
  });
}
```

2. Configurar `vercel.json`

```
{
  "crons": [
    {
      "path": "/api/cron/send-digests",
      "schedule": "0 * * * *"
    }
  ]
}
```

Opção 3: Node-Cron (Aplicação Node.js) 📦

Se preferir gerenciar dentro da aplicação:

1. Instalar dependência

```
cd /home/ubuntu/organizen/nextjs_space
yarn add node-cron
yarn add -D @types/node-cron
```

2. Criar serviço de agendamento

`/lib/scheduler.ts`:

```
import cron from 'node-cron';
import { getUsersForDigest, generateUserDigest } from './notification-service';
import { prisma } from './db';

export function startDigestScheduler() {
  // Executar a cada hora
  cron.schedule('0 * * * *', async () => {
    const now = new Date();
    const currentTime = `${now.getHours().toString().padStart(2, '0')}:00`;
    const currentDayOfWeek = now.getDay();
    const currentDayOfMonth = now.getDate();

    console.log(`🕐 [${currentTime}] Verificando resumos...`);

    // Processar resumos (similar aos scripts acima)
    // ...
  });

  console.log('✅ Agendador de resumos iniciado!');
}
```

3. Iniciar no servidor

No arquivo principal do servidor (ex: `server.ts` ou `app.ts`):

```
import { startDigestScheduler } from './lib/scheduler';

// Iniciar agendador
if (process.env.NODE_ENV === 'production') {
  startDigestScheduler();
}
```

📧 Template de Email para Resumos

Quando o sistema de email estiver configurado, use este template:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <style>
    body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; }
    .container { max-width: 600px; margin: 0 auto; padding: 20px; }
    .header { background: {{primaryColor}}; color: white; padding: 20px; text-align: c
enter; }
    .section { margin: 20px 0; padding: 15px; background: #f9f9f9; border-radius:
5px; }
    .stat { display: inline-block; margin: 10px; text-align: center; }
    .stat-number { font-size: 24px; font-weight: bold; color: {{primaryColor}}; }
    .stat-label { font-size: 12px; color: #666; }
    .footer { text-align: center; margin-top: 30px; font-size: 12px; color: #666; }
  </style>
</head>
<body>
  <div class="container">
    <div class="header">
      <h1>📊 Resumo {{period}} - {{companyName}}</h1>
      <p>{{startDate}} a {{endDate}}</p>
    </div>

    <div class="section">
      <h2>Resumo Geral</h2>
      <div class="stat">
        <div class="stat-number">{{tasksCreated}}</div>
        <div class="stat-label">Tarefas Criadas</div>
      </div>
      <div class="stat">
        <div class="stat-number">{{tasksCompleted}}</div>
        <div class="stat-label">Tarefas Concluídas</div>
      </div>
      <div class="stat">
        <div class="stat-number">{{messagesReceived}}</div>
        <div class="stat-label">Mensagens Recebidas</div>
      </div>
      <div class="stat">
        <div class="stat-number">{{shiftsScheduled}}</div>
        <div class="stat-label">Turnos Agendados</div>
      </div>
    </div>

    <div class="section">
      <h3>Tarefas Recentes</h3>
      {{#each tasks}}
        <p>✓ {{title}} - {{status}}</p>
      {{/each}}
    </div>

    <div class="section">
      <h3>Mensagens Não Lidas</h3>
      {{#if messagesUnread}}
        <p>Você tem <strong>{{messagesUnread}}</strong> mensagens não lidas.</p>
        <a href="{{appUrl}}/messages">Ver Mensagens</a>
      {{else}}
        <p>Todas as mensagens foram lidas! 🎉</p>
      {{/if}}
    </div>

    <div class="footer">
```

```
      <p>Este é um resumo automático do OrganiZen</p>
      <p><a href="{{appUrl}}/settings/notifications">Gerenciar Preferências</a></p>
    </div>
  </div>
</body>
</html>
```

🧪 Como Testar

Teste Manual via API

```
# Gerar resumo para um usuário específico
curl -X GET "http://localhost:3000/api/settings/notifications/digest?period=weekly" \
  -H "Cookie: your-session-cookie"
```

Teste do Script

```
cd /home/ubuntu/organizen/nextjs_space
NODE_ENV=production tsx scripts/send-digests.ts
```

📊 Monitoramento

Logs Recomendados

- ✅ Horário de execução
- ✅ Número de usuários processados
- ✅ Emails enviados com sucesso
- ✅ Erros encontrados
- ✅ Tempo de execução total

Ferramentas Úteis

- **Logs**: `tail -f /var/log/organizen-digests.log`
- **Cron Status**: `systemctl status cron`
- **Verificar Crontab**: `crontab -l`

🎯 Próximos Passos

1. ✅ **Fase 4 Completa** - Estrutura de preferências implementada
2. ⏳ **Configurar SMTP** - Para envio real de emails
3. ⏳ **Escolher método de agendamento** - Cron, Vercel Cron ou Node-Cron
4. ⏳ **Implementar script de envio** - Usar um dos exemplos acima
5. ⏳ **Criar templates de email** - Baseado no exemplo fornecido
6. ⏳ **Testar em produção** - Começar com poucos usuários
7. ⏳ **Monitorar e ajustar** - Logs e métricas de entrega

📝 Notas Importantes

- Os resumos **não são enviados instantaneamente** - são agendados
- Todos os resumos de um usuário são enviados no **mesmo horário** configurado
- O sistema **verifica a cada hora** se há resumos para enviar
- Usuários podem **desabilitar** resumos a qualquer momento nas configurações
- O script **não envia emails duplicados** - apenas no horário configurado

---

Status: ✅ **Estrutura Completa - Aguardando Configuração de SMTP e Agendamento**