INTEGRACAO-MODULOS-OTIMIZACAO.md

🚀 Integração dos Módulos de Otimização - YUNA

📋 Status da Implementação

✅ Módulos Criados (Prontos)

1. **`performance-monitor.js`** - 349 linhas
   - Monitoramento de performance, memória e erros
   - Snapshots automáticos a cada 5 minutos
   - Alertas quando RAM > 200MB
2. **`listener-manager.js`** - 286 linhas
   - Gerenciamento centralizado de listeners Firestore
   - Auto-cleanup em unload
   - Rastreamento com descrições
3. **`cache-manager.js`** - 410 linhas
   - Cache LRU com limite de 200 itens
   - Eviction automática de itens antigos
   - Sincronização com cache legado
4. **`query-helper.js`** - 380 linhas
   - Paginação com limit(50) e startAfter()
   - Cache de queries
   - Logging de reads

⏳ Integração em Andamento

- **`admin-panel.js`** (13.400 linhas) - 30% integrado
  - ✅ Inicialização dos módulos no topo
  - ✅ Performance monitor em `carregarSolicitacoes()`
  - ✅ QueryHelper com fallback
  - ⏳ Substituir atribuições diretas `window.cachedSolicitacoes =` por `cacheManager.setSolicitacao()`
  - ⏳ Registrar listeners com `listenerManager.register()`
  - ⏳ Finalizar timers de performance

🎯 Próximos Passos de Integração

Fase 1: Substituir Cache Direto (CRÍTICO)

Arquivo: `admin-panel.js` **Linhas afetadas:** 11 ocorrências de `window.cachedSolicitacoes =`

Padrão atual (PROBLEMA):

```
// Linha 4844 - atualiza cache inteiro (RUIM: sem limite)
window.cachedSolicitacoes = Array.isArray(solicitacoes) ? solicitacoes : [];
```

Novo padrão (SOLUÇÃO):

```
// Atualiza cache com limite LRU
if (window.cacheManager) {
    solicitacoes.forEach(sol => window.cacheManager.setSolicitacao(sol));
} else {
    // Fallback: cache legado
    window.cachedSolicitacoes = Array.isArray(solicitacoes) ? solicitacoes :
        [];
}
```

Substituições necessárias: - [ ] Linha 4844: `window.cachedSolicitacoes = Array.isArray(solicitacoes)` - [ ] Linha 8554: `window.cachedSolicitacoes = solicitacoesProcessadas` - [ ] Linha 8570: `window.cachedSolicitacoes = []` (reset em erro) - [ ] Outras 8 ocorrências

---

Fase 2: Registrar Listeners (CRÍTICO - Memory Leaks)

Arquivo: `admin-panel.js` **Buscar:** `.onSnapshot(` (todos os listeners)

Padrão atual (PROBLEMA):

```
// Listener sem cleanup - MEMORY LEAK!
const unsubscribe = db.collection('solicitacoes')
    .where('usuarioId', '==', userId)
    .onSnapshot(snapshot => {
        // processar
    });
```

Novo padrão (SOLUÇÃO):

```
// Listener com gerenciamento centralizado
const unsubscribe = db.collection('solicitacoes')
    .where('usuarioId', '==', userId)
    .onSnapshot(snapshot => {
        // processar
    });
```

```
// Registrar para auto-cleanup
if (window.listenerManager) {
    window.listenerManager.register(
        unsubscribe,
        'solicitacoes-usuario',
        { userId }
    );
}
```

Locais para buscar: - [ ] Listeners em `initRealtimeSync()` - [ ] Listeners em `carregarSolicitacoes()` - [ ] Listeners em funções de monitoramento - [ ] Listeners em telas específicas (relatórios, usuários)

Fase 3: Adicionar Timers de Performance (ALTA)

Arquivo: `admin-panel.js` **Funções críticas para monitorar:**

Padrão:

```
async function funcaoPesada() {
    const timerId = window.perfMonitor?.startTimer('funcaoPesada');

    try {
        // código pesado aqui

        window.perfMonitor?.endTimer(timerId);
    } catch (error) {
        window.perfMonitor?.endTimer(timerId);
        window.perfMonitor?.logError(error, 'funcaoPesada');
        throw error;
    }
}
```

Funções para adicionar timers: - [x] `carregarSolicitacoes()` - JÁ IMPLEMENTADO - [ ] `carregarUsuarios()` - [ ] `buscarAcompanhantes()` - [ ] `buscarEquipes()` - [ ] `renderizarCardsEquipe()` - [ ] `exportarExcel()` - [ ] `gerarRelatorio()`

Fase 4: Usar QueryHelper em Todas as Queries (ALTA)

Arquivo: `admin-panel.js` **Buscar:** `db.collection('solicitacoes').get()` e similares

Padrão atual (PROBLEMA):

```
// Busca tudo sem limite - LENTO com 300+ pacientes!
const snapshot = await db.collection('solicitacoes')
    .where('status', '==', 'pendente')
    .get();
```

Novo padrão (SOLUÇÃO):

```
// Paginação automática com limite 50
const resultado = await window.queryHelper.buscarSolicitacoes({
    filtros: { status: 'pendente' },
    limit: 50,
    ordenacao: { campo: 'criadoEm', direcao: 'desc' }
});

const solicitacoes = resultado.solicitacoes;
const hasMore = resultado.hasMore;
```

Queries para migrar: - [ ] `carregarSolicitacoes()` - já parcialmente implementado - [ ] `carregarUsuarios()` - [ ] `buscarSolicitacoesPorEquipe()` - [ ] `buscarSolicitacoesPorStatus()` - [ ] Relatórios e exports

---

Fase 5: Cleanup em Logout (CRÍTICO)

Arquivo: `admin-panel.js` **Função:** `performAutoLogout()` e `logout()`

Adicionar no logout:

```
async function performAutoLogout() {
    console.log('[TIMEOUT] Iniciando logout automático...');

    // Limpar listeners ativos
    if (window.listenerManager) {
        const count = window.listenerManager.unregisterAll();
        console.log(`[CLEANUP] ${count} listeners removidos`);
    }

    // Limpar cache
    if (window.cacheManager) {
        window.cacheManager.limpar();
        console.log('[CLEANUP] Cache LRU limpo');
    }

    // Gerar relatório final de performance
    if (window.perfMonitor) {
        const report = window.perfMonitor.generateReport();
        console.log('[PERFORMANCE] Relatório final:', report);
    }

    // Continuar com logout normal...
    await firebase.auth().signOut();
}
```

🧪 Como Testar Após Integração

1. Testar Cache LRU

```
// Console do navegador
console.log('=== TESTE CACHE LRU ===');
window.cacheManager.showCacheStats();
// Deve mostrar: tamanho <= 200, hits/misses, evictions
```

2. Testar Listeners

```
// Console do navegador
console.log('=== TESTE LISTENERS ===');
window.listenerManager.showListeners();
// Deve mostrar: lista de listeners ativos com descrições
// Máximo esperado: 10-15 listeners
```

3. Testar Performance

```
// Console do navegador
console.log('=== TESTE PERFORMANCE ===');
window.perfMonitor.showPerformanceReport();
// Verificar: tempos de carregarSolicitacoes < 1000ms
// Verificar: memória < 200MB
```

4. Testar Paginação

```
// Console do navegador
console.log('=== TESTE PAGINAÇÃO ===');
window.queryHelper.showPaginationStats();
// Verificar: reads por query <= 50
// Verificar: cache hits > 0 após segunda busca
```

5. Teste de Carga (Manual)

1. Criar 100+ solicitações de teste
2. Recarregar página
3. Verificar:
   - Tempo de carregamento < 3 segundos
   - Memória usada < 200MB (DevTools → Memory)
   - Console sem erros

6. Teste de Memory Leak

1. Navegar entre telas 10x (Dashboard → Relatórios → Usuários → Dashboard)
2. Abrir Chrome DevTools → Memory → Take Heap Snapshot
3. Verificar:
   - Listeners não aumentam indefinidamente

- Detached DOM nodes < 10
- Memória não cresce >10MB por ciclo

📊 Métricas de Sucesso

Antes da Otimização (100 pacientes)

- ⏱️ Tempo de carregamento: **3-5 segundos**
- 💾 Memória usada: **150-300MB**
- 📊 Firestore reads/carga: **600-800 documentos**
- 🦠 Memory leaks: **Sim** (listeners não limpos)

Depois da Otimização (300+ pacientes)

- ⏱️ Tempo de carregamento: **< 2 segundos** (target)
- 💾 Memória usada: **< 150MB** (target)
- 📊 Firestore reads/carga: **< 100 documentos** (paginação)
- 🦠 Memory leaks: **Não** (cleanup automático)

🚨 Problemas Conhecidos e Soluções

Problema 1: “window.cacheManager is undefined”

Causa: Ordem de carregamento dos scripts no HTML incorreta

Solução: Verificar `admin/index.html` - deve ter:

```
<script src="cache-manager.js"></script>
<script src="admin-panel.js"></script> <!-- SEMPRE POR ÚLTIMO -->
```

Problema 2: Cache legado ainda cresce ilimitado

Causa: Código legado ainda usa `window.cachedSolicitacoes.push()`

Solução: Buscar TODOS os `.push(` no código:

```
# PowerShell
Select-String -Path "admin-panel.js" -Pattern "cachedSolicitacoes\.push"
```

Problema 3: Listeners não são removidos

Causa: Esqueceu de registrar no ListenerManager

Solução: Verificar console no logout:

```
window.listenerManager.showListeners();
// Se lista não está vazia, há listeners esquecidos
```

Problema 4: Performance não melhora

Causa: Query sem paginação ainda busca tudo

Solução: Verificar logs no console:

```
[QueryHelper] Buscando solicitações com filtros...
[QueryHelper] Reads executados: 50 (esperado: <= 50)
```

📝 Checklist de Integração Completa

admin-panel.js

- [x] Inicialização dos módulos no topo do arquivo
- [x] Performance timer em `carregarSolicitacoes()`
- [ ] Performance timers em todas funções pesadas
- [ ] Substituir `window.cachedSolicitacoes =` por `cacheManager.setSolicitacao()`
- [ ] Registrar todos `.onSnapshot()` no `listenerManager`
- [ ] Usar `queryHelper.buscarSolicitacoes()` em todas queries
- [ ] Cleanup completo em `performAutoLogout()`

acompanhantes/index.html

- [ ] Adicionar imports dos módulos
- [ ] Implementar mesmos padrões que admin
- [ ] Testar paginação de histórico de solicitações

Testes

- [ ] Teste de cache LRU (console)
- [ ] Teste de listeners (console)
- [ ] Teste de performance (console)
- [ ] Teste de paginação (console)
- [ ] Teste de carga (100+ solicitações)
- [ ] Teste de memory leak (navegação 10x)

Deploy

- [ ] Commit com mensagem descritiva
- [ ] Push para GitHub
- [ ] Verificar deploy automático (Netlify/GitHub Pages)
- [ ] Testar em produção
- [ ] Monitorar logs do Firebase por 24h

🎓 Recursos de Referência

- **Módulos criados:** `admin/performance-monitor.js`, `listener-manager.js`, `cache-manager.js`, `query-helper.js`
- **Documentação de índices:** `FIRESTORE-INDEXES.md`
- **Guia para IA:** `.github/copilot-instructions.md`
- **Análise de escalabilidade:** (conversa anterior)

Última atualização: 08/01/2026
Autor: GitHub Copilot
Status: 🟡 Integração 30% completa