## 1. Catálogo de Materiais (CRUD Completo)

**models/Material.js**

```
const db = require('../config/db');
const ApiResponse = require('../utils/apiResponse');

class Material {
  // Criar material
  static async criar(nome, descricao, unidadeMedida) {
    try {
      const result = await db.query(
        'INSERT INTO material (nome_material, descricao_material, unidade_medida) VALUES ($1
        [nome, descricao, unidadeMedida]
      );
      return ApiResponse.success(result.rows[0]);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Listar todos materiais
  static async listar() {
    try {
      const result = await db.query('SELECT * FROM material ORDER BY nome_material');
      return ApiResponse.success(result.rows);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Buscar por ID
  static async buscarPorId(id) {
    try {
      const result = await db.query('SELECT * FROM material WHERE id_material = $1', [id]);
      if (result.rows.length === 0) {
        return ApiResponse.error('Material não encontrado', 404);
      }
      return ApiResponse.success(result.rows[0]);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Atualizar material
  static async atualizar(id, nome, descricao, unidadeMedida) {
```

```
    try {
      const result = await db.query(
        `UPDATE material
         SET nome_material = $1, descricao_material = $2, unidade_medida = $3
         WHERE id_material = $4
         RETURNING *`,
        [nome, descricao, unidadeMedida, id]
      );
      if (result.rowCount === 0) {
        return ApiResponse.error('Material não encontrado', 404);
      }
      return ApiResponse.success(result.rows[0]);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Remover material
  static async remover(id) {
    try {
      const result = await db.query(
        'DELETE FROM material WHERE id_material = $1 RETURNING *',
        [id]
      );
      if (result.rowCount === 0) {
        return ApiResponse.error('Material não encontrado', 404);
      }
      return ApiResponse.success(result.rows[0]);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }
}

module.exports = Material;
```

**controllers/materialController.js**

```
const Material = require('../models/Material');
const ApiResponse = require('../utils/apiResponse');

class MaterialController {
  // Criar material (apenas Controle de Materiais)
  static async criar(req, res) {
    const { nome, descricao, unidadeMedida } = req.body;
    const result = await Material.criar(nome, descricao, unidadeMedida);
```

```
    return res.status(result.statusCode || 200).json(result);
  }

  // Listar materiais
  static async listar(req, res) {
    const result = await Material.listar();
    return res.status(result.statusCode || 200).json(result);
  }

  // Buscar material por ID
  static async buscar(req, res) {
    const { id } = req.params;
    const result = await Material.buscarPorId(id);
    return res.status(result.statusCode || 200).json(result);
  }

  // Atualizar material (apenas Controle de Materiais)
  static async atualizar(req, res) {
    const { id } = req.params;
    const { nome, descricao, unidadeMedida } = req.body;
    const result = await Material.atualizar(id, nome, descricao, unidadeMedida);
    return res.status(result.statusCode || 200).json(result);
  }

  // Remover material (apenas Controle de Materiais)
  static async remover(req, res) {
    const { id } = req.params;
    const result = await Material.remover(id);
    return res.status(result.statusCode || 200).json(result);
  }
}

module.exports = MaterialController;
```

**routes/materialRoutes.js**

```
const express = require('express');
const router = express.Router();
const MaterialController = require('../controllers/materialController');
const { authRequired, checkUserType } = require('../middlewares/authMiddleware');

// CRUD completo (apenas para Controle de Materiais)
router.post('/', authRequired, checkUserType('ControleMateriais'), MaterialController.criar)
router.get('/', authRequired, MaterialController.listar);
router.get('/:id', authRequired, MaterialController.buscar);
router.put('/:id', authRequired, checkUserType('ControleMateriais'), MaterialController.atua
```

```
router.delete('/:id', authRequired, checkUserType('ControleMateriais'), MaterialController.r

module.exports = router;
```

## 2. Planejamento de Demanda

### models/Planejamento.js

```
const db = require('../config/db');
const ApiResponse = require('../utils/apiResponse');

class Planejamento {
  // Criar planejamento mensal
  static async criar(mesAno, idSetor, matriculaCriador, itens) {
    try {
      await db.query('BEGIN');

      // Criar planejamento
      const planejamentoResult = await db.query(
        `INSERT INTO planejamento_mensal
         (mes_ano_planejamento, id_setor, matricula_criador)
         VALUES ($1, $2, $3)
         RETURNING id_planejamento`,
        [mesAno, idSetor, matriculaCriador]
      );

      const idPlanejamento = planejamentoResult.rows[0].id_planejamento;

      // Inserir itens do planejamento
      for (const item of itens) {
        await db.query(
          `INSERT INTO item_planejamento
           (id_planejamento, id_material, quantidade_planejada)
           VALUES ($1, $2, $3)`,
          [idPlanejamento, item.idMaterial, item.quantidade]
        );
      }

      await db.query('COMMIT');
      return ApiResponse.success({ idPlanejamento });
    } catch (err) {
      await db.query('ROLLBACK');
      return ApiResponse.error(err.message);
    }
  }
```

```
// Buscar planejamento por ID
static async buscarPorId(id) {
  try {
    // Planejamento básico
    const planejamentoResult = await db.query(
      `SELECT p.*, s.nome_setor, u.nome_usuario as criador_nome
       FROM planejamento_mensal p
       JOIN setor s ON p.id_setor = s.id_setor
       JOIN usuario u ON p.matricula_criador = u.matricula
       WHERE p.id_planejamento = $1`,
      [id]
    );

    if (planejamentoResult.rows.length === 0) {
      return ApiResponse.error('Planejamento não encontrado', 404);
    }

    // Itens do planejamento
    const itensResult = await db.query(
      `SELECT ip.*, m.nome_material, m.unidade_medida
       FROM item_planejamento ip
       JOIN material m ON ip.id_material = m.id_material
       WHERE ip.id_planejamento = $1`,
      [id]
    );

    const response = {
      ...planejamentoResult.rows[0],
      itens: itensResult.rows
    };

    return ApiResponse.success(response);
  } catch (err) {
    return ApiResponse.error(err.message);
  }
}

// Listar planejamentos por setor
static async listarPorSetor(idSetor) {
  try {
    const result = await db.query(
      `SELECT p.*, s.nome_setor, u.nome_usuario as criador_nome
       FROM planejamento_mensal p
       JOIN setor s ON p.id_setor = s.id_setor
       JOIN usuario u ON p.matricula_criador = u.matricula
       WHERE p.id_setor = $1
```

```
         ORDER BY p.mes_ano_planejamento DESC`,
        [idSetor]
      );
      return ApiResponse.success(result.rows);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Obter histórico de consumo (últimos 3 meses)
  static async historicoConsumo(idSetor) {
    try {
      const result = await db.query(
        `SELECT m.id_material, m.nome_material, m.unidade_medida,
                SUM(is.quantidade_atendida) as quantidade_total,
                DATE_TRUNC('month', s.data_solicitacao) as mes
         FROM item_solicitacao is
         JOIN solicitacao s ON is.id_solicitacao = s.id_solicitacao
         JOIN material m ON is.id_material = m.id_material
         WHERE s.id_setor = $1
           AND s.status_solicitacao = 'Concluída'
           AND s.data_solicitacao >= DATE_TRUNC('month', NOW() - INTERVAL '3 months')
         GROUP BY m.id_material, m.nome_material, m.unidade_medida, mes
         ORDER BY mes DESC, m.nome_material`,
        [idSetor]
      );
      return ApiResponse.success(result.rows);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }
}

module.exports = Planejamento;
```

**controllers/planejamentoController.js**

```
const Planejamento = require('../models/Planejamento');
const ApiResponse = require('../utils/apiResponse');

class PlanejamentoController {
  // Criar planejamento (Coordenador)
  static async criar(req, res) {
    const { mesAno, itens } = req.body;
    const { matricula } = req.user;
    const idSetor = req.user.idSetor;
```

```
    const result = await Planejamento.criar(mesAno, idSetor, matricula, itens);
    return res.status(result.statusCode || 200).json(result);
  }

  // Buscar planejamento por ID
  static async buscar(req, res) {
    const { id } = req.params;
    const result = await Planejamento.buscarPorId(id);
    return res.status(result.statusCode || 200).json(result);
  }

  // Listar planejamentos do setor
  static async listarPorSetor(req, res) {
    const idSetor = req.user.idSetor;
    const result = await Planejamento.listarPorSetor(idSetor);
    return res.status(result.statusCode || 200).json(result);
  }

  // Obter histórico de consumo
  static async historicoConsumo(req, res) {
    const idSetor = req.user.idSetor;
    const result = await Planejamento.historicoConsumo(idSetor);
    return res.status(result.statusCode || 200).json(result);
  }
}

module.exports = PlanejamentoController;
```

**routes/planejamentoRoutes.js**

```
const express = require('express');
const router = express.Router();
const PlanejamentoController = require('../controllers/planejamentoController');
const { authRequired, checkUserType } = require('../middlewares/authMiddleware');

// Rotas para coordenadores
router.post('/', authRequired, checkUserType('Coordenador'), PlanejamentoController.criar);
router.get('/setor', authRequired, checkUserType('Coordenador'), PlanejamentoController.list
router.get('/historico-consumo', authRequired, checkUserType('Coordenador'), PlanejamentoCo

// Rota pública para visualização
router.get('/:id', authRequired, PlanejamentoController.buscar);

module.exports = router;
```

## 3. Histórico e Relatórios

**models/Relatorio.js**

```
const db = require('../config/db');
const ApiResponse = require('../utils/apiResponse');

class Relatorio {
  // Relatório de solicitações por período
  static async solicitacoesPorPeriodo(idSetor, dataInicio, dataFim) {
    try {
      const result = await db.query(
        `SELECT s.id_solicitacao, s.data_solicitacao, s.status_solicitacao,
                u.nome_usuario as solicitante,
                COUNT(is.id_item_solicitacao) as total_itens,
                SUM(is.quantidade_solicitada) as quantidade_total
         FROM solicitacao s
         JOIN usuario u ON s.matricula_solicitante = u.matricula
         LEFT JOIN item_solicitacao is ON s.id_solicitacao = is.id_solicitacao
         WHERE s.id_setor = $1
           AND s.data_solicitacao BETWEEN $2 AND $3
         GROUP BY s.id_solicitacao, u.nome_usuario
         ORDER BY s.data_solicitacao DESC`,
        [idSetor, dataInicio, dataFim]
      );
      return ApiResponse.success(result.rows);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Relatório de materiais mais solicitados
  static async materiaisMaisSolicitados(idSetor, limite = 5) {
    try {
      const result = await db.query(
        `SELECT m.id_material, m.nome_material, m.unidade_medida,
                COUNT(is.id_item_solicitacao) as vezes_solicitado,
                SUM(is.quantidade_solicitada) as quantidade_total
         FROM item_solicitacao is
         JOIN solicitacao s ON is.id_solicitacao = s.id_solicitacao
         JOIN material m ON is.id_material = m.id_material
         WHERE s.id_setor = $1
           AND s.status_solicitacao = 'Concluída'
         GROUP BY m.id_material, m.nome_material, m.unidade_medida
         ORDER BY quantidade_total DESC
         LIMIT $2`,
```

```
        [idSetor, limite]
      );
      return ApiResponse.success(result.rows);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }
}

module.exports = Relatorio;
```

**controllers/relatorioController.js**

```
const Relatorio = require('../models/Relatorio');
const ApiResponse = require('../utils/apiResponse');

class RelatorioController {
  // Relatório de solicitações por período
  static async solicitacoesPorPeriodo(req, res) {
    const { dataInicio, dataFim } = req.query;
    const idSetor = req.user.idSetor;

    const result = await Relatorio.solicitacoesPorPeriodo(idSetor, dataInicio, dataFim);
    return res.status(result.statusCode || 200).json(result);
  }

  // Relatório de materiais mais solicitados
  static async materiaisMaisSolicitados(req, res) {
    const { limite } = req.query;
    const idSetor = req.user.idSetor;

    const result = await Relatorio.materiaisMaisSolicitados(idSetor, limite);
    return res.status(result.statusCode || 200).json(result);
  }
}

module.exports = RelatorioController;
```

**routes/relatorioRoutes.js**

```
const express = require('express');
const router = express.Router();
const RelatorioController = require('../controllers/relatorioController');
const { authRequired } = require('../middlewares/authMiddleware');

router.get('/solicitacoes-periodo', authRequired, RelatorioController.solicitacoesPorPeriodo
```

```
router.get('/materiais-mais-solicitados', authRequired, RelatorioController.materiaisMaisSol

module.exports = router;
```

## 4. Notificações

**models/Notificacao.js**

```
const db = require('../config/db');
const ApiResponse = require('../utils/apiResponse');

class Notificacao {
  // Criar notificação
  static async criar(matriculaDestino, titulo, mensagem, tipo, idRelacionado) {
    try {
      const result = await db.query(
        `INSERT INTO notificacao
         (matricula_destino, titulo, mensagem, tipo, id_relacionado, lida, data_criacao)
         VALUES ($1, $2, $3, $4, $5, false, NOW())
         RETURNING *`,
        [matriculaDestino, titulo, mensagem, tipo, idRelacionado]
      );
      return ApiResponse.success(result.rows[0]);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Listar notificações do usuário
  static async listarPorUsuario(matricula) {
    try {
      const result = await db.query(
        `SELECT * FROM notificacao
         WHERE matricula_destino = $1
         ORDER BY data_criacao DESC
         LIMIT 50`,
        [matricula]
      );
      return ApiResponse.success(result.rows);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }

  // Marcar como lida
  static async marcarComoLida(id, matricula) {
```

```
    try {
      const result = await db.query(
        `UPDATE notificacao
         SET lida = true
         WHERE id_notificacao = $1 AND matricula_destino = $2
         RETURNING *`,
        [id, matricula]
      );
      if (result.rowCount === 0) {
        return ApiResponse.error('Notificação não encontrada', 404);
      }
      return ApiResponse.success(result.rows[0]);
    } catch (err) {
      return ApiResponse.error(err.message);
    }
  }
}

module.exports = Notificacao;
```

**controllers/notificacaoController.js**

```
const Notificacao = require('../models/Notificacao');
const ApiResponse = require('../utils/apiResponse');

class NotificacaoController {
  // Listar notificações do usuário
  static async listar(req, res) {
    const { matricula } = req.user;
    const result = await Notificacao.listarPorUsuario(matricula);
    return res.status(result.statusCode || 200).json(result);
  }

  // Marcar notificação como lida
  static async marcarComoLida(req, res) {
    const { id } = req.params;
    const { matricula } = req.user;
    const result = await Notificacao.marcarComoLida(id, matricula);
    return res.status(result.statusCode || 200).json(result);
  }
}

module.exports = NotificacaoController;
```

**routes/notificacaoRoutes.js**

```
const express = require('express');
const router = express.Router();
const NotificacaoController = require('../controllers/notificacaoController');
const { authRequired } = require('../middlewares/authMiddleware');

router.get('/', authRequired, NotificacaoController.listar);
router.put('/:id/marcar-lida', authRequired, NotificacaoController.marcarComoLida);

module.exports = router;
```

## Atualização do routes/index.js

```
const express = require('express');
const router = express.Router();

// Importar todas as rotas
const authRoutes = require('./authRoutes');
const setorRoutes = require('./setorRoutes');
const usuarioRoutes = require('./usuarioRoutes');
const materialRoutes = require('./materialRoutes');
const solicitacaoRoutes = require('./solicitacaoRoutes');
const planejamentoRoutes = require('./planejamentoRoutes');
const relatorioRoutes = require('./relatorioRoutes');
const notificacaoRoutes = require('./notificacaoRoutes');

// Configurar rotas
router.use('/auth', authRoutes);
router.use('/setores', setorRoutes);
router.use('/usuarios', usuarioRoutes);
router.use('/materiais', materialRoutes);
router.use('/solicitacoes', solicitacaoRoutes);
router.use('/planejamentos', planejamentoRoutes);
router.use('/relatorios', relatorioRoutes);
router.use('/notificacoes', notificacaoRoutes);

module.exports = router;
```

## Funcionalidades Adicionadas

1. **Catálogo de Materiais**:
   - CRUD completo (Create, Read, Update, Delete)
   - Acesso restrito ao setor de Controle de Materiais
   - Listagem pública para todos usuários autenticados
2. **Planejamento de Demanda**:

- Criação de planejamentos mensais com múltiplos itens
- Visualização de planejamentos por setor
- Histórico de consumo (últimos 3 meses)
- Acesso restrito a coordenadores de setor

3. **Histórico e Relatórios**:
   - Solicitações por período
   - Materiais mais solicitados
   - Dados agregados para análise
4. **Notificações**:
   - Listagem de notificações por usuário
   - Marcar notificações como lidas
   - Integração futura com eventos do sistema

Todas as rotas seguem o padrão RESTful e retornam JSON padronizado. As rotas sensíveis são protegidas por autenticação JWT e autorização por tipo de usuário.