Projeto Sistema de Farmácia

Documento Técnico – **Sprint 1: Auth + RBAC + Seeds + Audit Logs**

Objetivo: entregar o “esqueleto” operacional do sistema: autenticação, autorização por permissões, multi-loja (storeId no contexto), seeds iniciais (central/lojas/roles/permissões) e auditoria.

Princípio: qualidade e objetividade – o mínimo completo para destravar os módulos seguintes (estoque, PDV, caixa).

1) Estrutura de pastas (backend)

```
C:\pharma\backend\src
  \modules
    \auth
      auth.controller.ts
      auth.service.ts
      auth.dto.ts
      auth.guard.ts
      token.service.ts
    \rbac
      rbac.guard.ts
      permissions.ts
    \audit
      audit.service.ts
      audit.interceptor.ts
    \stores
      stores.controller.ts
      stores.service.ts
  \common
    prisma.service.ts
    request-context.middleware.ts
    errors.ts
  main.ts / server.ts
```

2) Decisões técnicas (sem complicar)

- Auth com JWT (access token curto) + refresh token (longo).
- RBAC por permissões (RolePermission.permissionKey).
- Contexto multi-loja: storeId obrigatório no request (via header ou claim).
- Audit log para ações sensíveis e para eventos-chave (login, pagamento, estorno, ajuste de estoque).

3) Contrato de API (endpoints mínimos)

Método	Rota	Descrição	Auth
POST	/auth/login	Login e retorno de access + refresh	Não
POST	/auth/refresh	Troca refresh por novo access	Não
POST	/auth/logout	Revoga refresh (opcional no MVP)	Sim
GET	/me	Retorna usuário + role + permissões + lojas	Sim
GET	/stores	Lista lojas permitidas para o usuário	Sim

3.1 Payloads mínimos (exemplo)

Login (request):

```
POST /auth/login
{ "email": "admin@pharma.local", "password": "********" }
```

Login (response):

```
{
  "accessToken": "jwt...",
  "refreshToken": "jwt...",
  "user": { "id": "...", "name": "...", "role": "ADMIN" },
  "stores": [{ "id": "...", "name": "CENTRAL", "type": "CENTRAL", "isDefault": true }]
}
```

4) Contexto de loja (storeId) – regra objetiva

Regra: toda chamada operacional deve carregar o storeId atual.

Implementação recomendada: header **X-Store-Id** em todas as requisições autenticadas.

```
GET /me
Authorization: Bearer <accessToken>
X-Store-Id: <storeId>
```

Validação:
• O usuário deve ter acesso à loja (StoreUser).

• Se não enviar X-Store-Id, usar a loja default (StoreUser.isDefault=true) quando existir.

5) RBAC (permissões) – como aplicar

Fluxo:
1) AuthGuard valida JWT e injeta userId no request.

2) RequestContext resolve storeId e valida acesso (StoreUser).

3) RbacGuard verifica permissionKey exigida na rota.

Exemplo de uso em rota:

```
@RequirePermission("cash.open")
POST /cash/sessions/open
```

5.1 Conjunto inicial de permissões (mínimo)

- **users.manage**
- **stores.manage**
- **products.manage**
- **inventory.receive**
- **inventory.adjust**
- **sales.create**
- **sales.cancel**
- **cash.open**
- **cash.close**

- **cash.refund**
- **reports.view**

6) Auditoria (audit_logs) – eventos mínimos

- Login realizado (action=AUTH_LOGIN).
- Troca de token (AUTH_REFRESH) – opcional.
- Abertura/fechamento de caixa.
- Estorno/cancelamento de venda.
- Ajuste de estoque.
- Alteração de preço ou produto (ADMIN).

Padrão de payload (Json): antes/depois quando houver alteração, e motivo obrigatório para ações sensíveis.

7) Seeds iniciais (objetivos e lista)

Objetivo: subir o sistema ‘com vida’ para teste já no primeiro dia.

7.1 Seeds obrigatórias

- Stores: 1 CENTRAL + 1 LOJA (mínimo).
- Roles: ADMIN, CAIXA, VENDEDOR, FARMACÊUTICO.
- RolePermissions: conjunto mínimo por role.
- Usuário ADMIN inicial + loja default.

7.2 Matriz mínima de permissões por role (sugestão)

Permissão	ADMIN	CAIXA	VENDEDOR	FARMACÊUTICO
users.manage	✓			
stores.manage	✓			
products.manage	✓			✓ (leitura/validação)
inventory.receive	✓			✓
inventory.adjust	✓			✓ (com auditoria)
sales.create	✓		✓	✓ (validação)
sales.cancel	✓	✓ (com supervisor)	✓ (antes de pagar)	
cash.open	✓	✓		
cash.close	✓	✓		
cash.refund	✓	✓ (com supervisor)		
reports.view	✓	✓ (loja)	✓ (limitado)	✓ (estoque/validade)

7.3 Seed script (localização recomendada)

Criar arquivo: **C:\pharma\backend\prisma\seed.ts**

```
// executar: npx prisma db seed
// (wire no package.json: "prisma": { "seed": "ts-node prisma/seed.ts" })
// seeds: stores, roles, permissions, admin user, store default
```

8) Definition of Done (DoD) – Sprint 1

- POST /auth/login retorna tokens e lista de lojas do usuário.
- GET /me retorna usuário, role, permissões e loja atual (storeId).
- Header X-Store-Id aplicado e validado por StoreUser.
- RbacGuard bloqueia rotas sem permissionKey.
- Audit logs registra eventos mínimos (login + 1 ação sensível).
- Seeds criam CENTRAL, 1 LOJA, roles, permissões e ADMIN inicial.

Pasta recomendada para este PDF: **C:\pharma\docs\10_sprint1_auth_rbac**