Atividade Programada – CRUD de Pedido (com middleware de papéis, TS e Swagger)

> Fala, turma! Vamos criar o CRUD de **Pedido** do jeitinho que o projeto já usa. Nada de mudar pastas, nada de instalar lib nova e tudo em **TypeScript (.ts)**. Bora juntos passo a passo.

Projeto base: https://github.com/JhonataProf/projeto-basico-com-crud-usuario

Regras gerais: - **Persistência**: usar `src/models/pedido-model.ts` (não alterar). - **Status**: usar o enum já existente em `src/enum/status-pedido.ts`. - **Autenticação**: usar `src/middlewares/auth-middleware.ts`, agora incrementado para validar token JWT. - **Autorização (RBAC)**: criar um middleware **no mesmo formato do auth-middleware**, chamado `authorize-roles.ts`, que verifica se o papel do usuário pode acessar a rota. - **Swagger**: documentar **somente** no `src/docs/api/swagger.yaml` (incremental).

---

1) Estrutura de arquivos

- **Rotas (pedido):**
- `src/routes/criar-pedido.ts`
- `src/routes/listar-pedidos.ts`
- `src/routes/buscar-pedido.ts`
- `src/routes/atualizar-pedido.ts`
- `src/routes/deletar-pedido.ts`

- **Controllers (pedido):**

- `src/controllers/pedido/criar-pedido.ts`
- `src/controllers/pedido/listar-pedidos.ts`
- `src/controllers/pedido/buscar-pedido.ts`
- `src/controllers/pedido/atualizar-pedido.ts`
- `src/controllers/pedido/deletar-pedido.ts`

- **Service:**

- `src/service/pedido-service.ts`

- **Interfaces e Types:**

- `src/interfaces/index.ts` (ex.: `CreatePedidoDTO`, `UpdatePedidoDTO`, `PedidoItemDTO`)
- `src/types/index.ts` (ex.: `RequestComUsuario`)

- **Middlewares:**

- `src/middlewares/auth-middleware.ts` (já existe, vamos incrementar com validação de token JWT)
- `src/middlewares/authorize-roles.ts` (novo, mesmo padrão do auth)

---

2) Regras do Pedido

- **Total** = soma de `quantidade * precoUnitario` de todos os itens.
- **Status**: usar o enum `StatusPedido` de `src/enum/status-pedido.ts`.
- `CRIADO → PAGO` ou `CRIADO → CANCELADO`
- `PAGO → ENVIADO` ou `PAGO → CANCELADO`
- `ENVIADO` e `CANCELADO` não mudam mais.

---

3) Middleware de autenticação (`auth-middleware.ts`)

Vamos usar o `jsonwebtoken` como já foi feito em `refresh-token.ts`. Assim garantimos que só usuários com token válido passam.

```
// src/middlewares/auth-middleware.ts
import jwt from 'jsonwebtoken';
import { Request, Response, NextFunction } from 'express';

export default async function authMiddleware(req: Request, res: Response,
next: NextFunction) {
  try {
    const authHeader = req.headers.authorization || '';
    const [, token] = authHeader.split(' ');
    if (!token) return res.status(401).json({ message: 'Token ausente' });

    const decoded: any = jwt.verify(token, process.env.JWT_SECRET as string);

    (req as any).user = {
      id: decoded.sub,
      email: decoded.email,
      role: decoded.role
    };

    return next();
  } catch (err) {
    return res.status(401).json({ message: 'Token inválido' });
  }
}
```

---

4) Middleware de papéis (`authorize-roles.ts`)

Esse middleware é igual no estilo ao `auth-middleware`, só que confere os papéis.

```ts
// src/middlewares/authorize-roles.ts
import { Request, Response, NextFunction } from 'express';

export default function authorizeRoles(roles: string[]) {
  return (req: Request, res: Response, next: NextFunction) => {
    const role = (req as any).user?.role;
    if (!role || !roles.includes(role)) {
      return res.status(403).json({ message: 'Acesso negado' });
    }
    next();
  };
}
```

---

5) O que significa RBAC?

RBAC = **Role-Based Access Control**, ou **Controle de Acesso Baseado em Papéis**. É o jeito de dizer: "essa rota só pode ser acessada por quem tem tal papel".

Exemplo: - Criar pedido → só `ADMIN` e `GERENTE`. - Listar pedidos → `ADMIN`, `GERENTE`, `ATENDENTE`. - Buscar pedido por id → `ADMIN`, `GERENTE`, `ATENDENTE`. - Atualizar pedido → `ADMIN`, `GERENTE`. - Deletar pedido → só `ADMIN`.

---

6) Swagger – só no `src/docs/api/swagger.yaml`

Além dos schemas e paths básicos, vamos deixar tudo mais rico com **examples** e **respostas de erro** (400/401/403/404).

Schemas (já visto antes):

```yaml
components:
  schemas:
    PedidoItem:
      type: object
      properties:
        produtoId: { type: string, example: "p-123" }
        quantidade: { type: integer, minimum: 1, example: 2 }
        precoUnitario: { type: number, minimum: 0, example: 10.5 }
      required: [produtoId, quantidade, precoUnitario]
    Pedido:
      type: object
      properties:
```

```
        id: { type: string, example: "ped-001" }
        usuarioId: { type: string, example: "u-123" }
        itens:
          type: array
          items: { $ref: '#/components/schemas/PedidoItem' }
        total: { type: number, example: 21.0 }
        status: { type: string, enum: [CRIADO, PAGO, CANCELADO, ENVIADO],
example: CRIADO }
        createdAt: { type: string, format: date-time, example:
"2025-09-15T10:00:00Z" }
        updatedAt: { type: string, format: date-time, example:
"2025-09-15T10:00:00Z" }
    CreatePedido:
      type: object
      properties:
        usuarioId: { type: string, example: "u-123" }
        itens:
          type: array
          items: { $ref: '#/components/schemas/PedidoItem' }
      required: [usuarioId, itens]
    UpdatePedido:
      type: object
      properties:
        itens:
          type: array
          items: { $ref: '#/components/schemas/PedidoItem' }
        status: { type: string, enum: [CRIADO, PAGO, CANCELADO, ENVIADO],
example: PAGO }
```

Paths com exemplos e respostas:

```
paths:
  /pedidos:
    get:
      summary: Lista todos os pedidos
      security: [ { bearerAuth: [] } ]
      tags: [Pedidos]
      responses:
        '200':
          description: Lista de pedidos
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Pedido'
        '401': { description: Token inválido ou ausente }
        '403': { description: Sem permissão }
    post:
      summary: Cria um pedido
```

```
      security: [ { bearerAuth: [] } ]
      tags: [Pedidos]
      requestBody:
        required: true
        content:
          application/json:
            schema: { $ref: '#/components/schemas/CreatePedido' }
            example:
              usuarioId: "u-123"
              itens:
                - produtoId: "p-1"
                  quantidade: 2
                  precoUnitario: 10
      responses:
        '201': { description: Criado }
        '400': { description: Erro de validação }
        '401': { description: Token inválido ou ausente }
        '403': { description: Sem permissão }

  /pedidos/{id}:
    get:
      summary: Busca pedido por ID
      security: [ { bearerAuth: [] } ]
      tags: [Pedidos]
      parameters:
        - in: path
          name: id
          required: true
          schema: { type: string }
          example: "ped-001"
      responses:
        '200': { description: OK, content: { application/json: { schema: {
$ref: '#/components/schemas/Pedido' } } } }
        '401': { description: Token inválido ou ausente }
        '403': { description: Sem permissão }
        '404': { description: Não encontrado }
    put:
      summary: Atualiza pedido
      security: [ { bearerAuth: [] } ]
      tags: [Pedidos]
      parameters:
        - in: path
          name: id
          required: true
          schema: { type: string }
          example: "ped-001"
      requestBody:
        required: true
        content:
          application/json:
            schema: { $ref: '#/components/schemas/UpdatePedido' }
```

```
            example:
              status: "PAGO"
      responses:
        '200': { description: OK }
        '400': { description: Erro de validação }
        '401': { description: Token inválido ou ausente }
        '403': { description: Sem permissão }
        '404': { description: Não encontrado }
    delete:
      summary: Remove pedido
      security: [ { bearerAuth: [] } ]
      tags: [Pedidos]
      parameters:
        - in: path
          name: id
          required: true
          schema: { type: string }
          example: "ped-001"
      responses:
        '204': { description: Sem conteúdo }
        '401': { description: Token inválido ou ausente }
        '403': { description: Sem permissão }
        '404': { description: Não encontrado }
```

---

13) Patch de Swagger com exemplos (colar/mesclar em `src/docs/api/swagger.yaml`)

> Dica: ajuste nomes/chaves caso alguns `components` / `paths` já existam. Não duplique chaves.

Schemas (com exemplos):

```
components:
  schemas:
    PedidoItem:
      type: object
      properties:
        produtoId: { type: string, example: "p-100" }
        quantidade: { type: integer, minimum: 1, example: 2 }
        precoUnitario: { type: number, minimum: 0, example: 10.5 }
      required: [produtoId, quantidade, precoUnitario]
    Pedido:
      type: object
      properties:
        id: { type: string, example: "ped-abc123" }
        usuarioId: { type: string, example: "u-123" }
```

```
        itens:
          type: array
          items: { $ref: '#/components/schemas/PedidoItem' }
        total: { type: number, example: 21 }
        status: { type: string, enum: [CRIADO, PAGO, CANCELADO, ENVIADO],
example: CRIADO }
        createdAt: { type: string, format: date-time, example:
"2025-01-10T12:00:00.000Z" }
        updatedAt: { type: string, format: date-time, example:
"2025-01-10T12:00:00.000Z" }
    CreatePedido:
      type: object
      properties:
        usuarioId: { type: string, example: "u-123" }
        itens:
          type: array
          items: { $ref: '#/components/schemas/PedidoItem' }
      required: [usuarioId, itens]
      example:
        usuarioId: "u-123"
        itens:
          - produtoId: "p-100"
            quantidade: 2
            precoUnitario: 10.5
    UpdatePedido:
      type: object
      properties:
        itens:
          type: array
          items: { $ref: '#/components/schemas/PedidoItem' }
        status: { type: string, enum: [CRIADO, PAGO, CANCELADO, ENVIADO],
example: PAGO }
      example:
        status: PAGO
```

Paths (com respostas 400/401/403/404 e exemplos):

```
paths:
  /pedidos:
    get:
      summary: Lista todos os pedidos
      security: [ { bearerAuth: [] } ]
      tags: [Pedidos]
      responses:
        '200':
          description: Lista de pedidos
          content:
            application/json:
              schema:
                type: array
```

```
              items: { $ref: '#/components/schemas/Pedido' }
      '401': { description: Não autenticado }
      '403': { description: Proibido }
  post:
    summary: Cria um pedido
    security: [ { bearerAuth: [] } ]
    tags: [Pedidos]
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/CreatePedido' }
          example:
            usuarioId: "u-123"
            itens:
              - produtoId: "p-100"
                quantidade: 2
                precoUnitario: 10.5
    responses:
      '201':
        description: Criado
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Pedido' }
      '400': { description: Erro de validação }
      '401': { description: Não autenticado }
      '403': { description: Proibido }

/pedidos/{id}:
  get:
    summary: Busca pedido por ID
    security: [ { bearerAuth: [] } ]
    tags: [Pedidos]
    parameters:
      - in: path
        name: id
        required: true
        schema: { type: string }
        example: ped-abc123
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Pedido' }
      '401': { description: Não autenticado }
      '403': { description: Proibido }
      '404': { description: Não encontrado }
  put:
    summary: Atualiza pedido
    security: [ { bearerAuth: [] } ]
```

```
    tags: [Pedidos]
    parameters:
      - in: path
        name: id
        required: true
        schema: { type: string }
        example: ped-abc123
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/UpdatePedido' }
          examples:
            atualizarStatus:
              summary: Marcar como PAGO
              value: { status: PAGO }
            atualizarItens:
              summary: Atualizar itens
              value:
                itens:
                  - produtoId: "p-100"
                    quantidade: 3
                    precoUnitario: 10.5
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Pedido' }
      '400': { description: Erro de validação }
      '401': { description: Não autenticado }
      '403': { description: Proibido }
      '404': { description: Não encontrado }
  delete:
    summary: Remove pedido
    security: [ { bearerAuth: [] } ]
    tags: [Pedidos]
    parameters:
      - in: path
        name: id
        required: true
        schema: { type: string }
        example: ped-abc123
    responses:
      '204': { description: Removido }
      '401': { description: Não autenticado }
      '403': { description: Proibido }
      '404': { description: Não encontrado }
```

Pronto! Agora o `swagger.yaml` tem exemplos de **request** e **response** e cobre erros (400, 401, 403, 404).