1) Catálogo de Produtos (Loja Básica)

Contexto: cadastre, liste, atualize e desative produtos vendidos.

Tabelas:

- produto(id PK, sku UNIQUE, nome, descricao, preco NUMERIC(12,2), quantidade INT, ativo BOOLEAN, criado_em TIMESTAMP, atualizado_em TIMESTAMP)

CRUD:

- **Create:** POST /produtos – inserir (retornar id).
- **Read:** GET /produtos/{id}, GET /produtos?ativo=true&nome=... (filtros, paginação).
- **Update:** PUT /produtos/{id} – atualizar campos.
- **Delete:** DELETE /produtos/{id} – exclusão lógica (ativo=false).

10 Regras de negócio:

1. sku obrigatório, único e imutável após criação.
2. preco ≥ 0; rejeitar negativos.
3. quantidade ≥ 0; operações de estoque não podem deixar negativo.
4. nome obrigatório (3–120 chars).
5. Exclusão é **lógica** (não pode apagar linha).
6. ativo=false impede aparecer em listagens padrão (filtro default).
7. Ao atualizar preco, guardar atualizado_em=NOW().
8. Proibir atualização de id e sku.

---

2) Gestão de Clientes (CRM simples)

Contexto: cadastro de pessoa física com contato e status.

Tabelas:

- cliente(id PK, nome, email UNIQUE, telefone, cpf UNIQUE, status VARCHAR(20), criado_em, atualizado_em)

CRUD:

- **Create:** POST /clientes
- **Read:** GET /clientes/{id}, GET /clientes?status=ATIVO&nome=...
- **Update:** PUT /clientes/{id}
- **Delete:** DELETE /clientes/{id} – exclusão lógica via status='INATIVO'

Regras:

1. cpf válido (formato e dígitos verificadores).
2. email obrigatório e único.
3. status ∈ {ATIVO, INATIVO, PROSPECT}.
4. nome obrigatório (min 3 chars).
5. Telefone opcional, se enviado validar DDI+DDD.
6. Exclusão torna status=INATIVO.
7. Não permitir duplicar cpf/email.
8. Atualização de cpf proibida.
9. atualizado_em sempre que alterar.

---

3) Ordens de Serviço (OS) para Assistência Técnica

Contexto: abrir, atualizar e fechar ordens ligadas a clientes.

Tabelas:

- cliente(id PK, nome, email)
- ordem_servico(id PK, cliente_id FK, descricao, status, valor NUMERIC(12,2), aberto_em, fechado_em, atualizado_em)

CRUD:

- **Create:** POST /os (cliente existente).
- **Read:** GET /os/{id}, GET /os?status=ABERTA&clienteId=...
- **Update:** PUT /os/{id}
- **Delete:** não permitido (apenas fechar/cancelar).

Regras:

1. status {ABERTA, EM_ANDAMENTO, FECHADA, CANCELADA}.
2. cliente_id deve existir.
3. Fechamento seta fechado_em e trava edição de cliente_id/descricao.
4. Cancelada não pode voltar a ABERTA.
5. Alterar status somente em transições válidas (ABERTA→EM_ANDAMENTO→FECHADA | ABERTA→CANCELADA).
6. Não excluir OS; manter histórico.
7. Filtro por período aberto_em BETWEEN.
8. Índice em cliente_id e status.

9. Auditar mudanças de status (tabela extra opcional).

---

4) Biblioteca: Livros & Empréstimos

Contexto: cadastro de livros e controle de empréstimos a usuários.

Tabelas:

- livro(id PK, isbn UNIQUE, titulo, autor, estoque INT, ativo BOOLEAN)
- usuario(id PK, nome, email UNIQUE)
- emprestimo(id PK, livro_id FK, usuario_id FK, retirado_em, devolucao_prevista, devolvido_em)

CRUD:

- **Create:** POST /livros, POST /usuarios, POST /emprestimos
- **Read:** GET /livros?titulo=..., GET /emprestimos?usuarioId=...
- **Update:** PUT /livros/{id}, PUT /emprestimos/{id} (registrar devolução)
- **Delete:** exclusão lógica em livro.ativo=false

Regras:

1. Empréstimo só se estoque > 0.
2. Ao emprestar, estoque--; ao devolver, estoque++.
3. devolucao_prevista = retirado_em + 7.
4. Um usuário não pode pegar o mesmo livro em aberto duplicado.
5. isbn obrigatório e único.
6. Devolução exige preencher devolvido_em.
7. Livros inativos não podem ser emprestados.
8. Multa (R$2,00 por dia) se devolução atrasada (calcular dias).
9. Lista de livros emprestados e usuário que pegou.

---

5) Controle de Despesas Pessoais

Contexto: cadastre categorias e despesas do usuário.

Tabelas:

- categoria(id PK, nome UNIQUE)
- despesa(id PK, categoria_id FK, descricao, valor NUMERIC(12,2), pago_em DATE, usuario_id, criado_em)

CRUD:

- **Create:** POST /categorias, POST /despesas
- **Read:** GET /despesas?categoriaId=...&de=...&ate=...
- **Update:** PUT /despesas/{id}
- **Delete:** DELETE /despesas/{id} (hard delete permitido)

Regras:

1. categoria_id deve existir.
2. pago_em não no futuro.
3. descricao 3–200 chars.
4. Não permitir remover categoria se houver despesas.
5. Somatórios por período e categoria.
6. Filtro por data de pago_em
7. Validar que usuário só vê/edita suas despesas.

---

6) Agendador de Consultas (Clínica)

Contexto: pacientes, médicos, agenda e consultas.

Tabelas:

- paciente(id PK, nome, cpf UNIQUE)
- medico(id PK, nome, crm UNIQUE, especialidade)
- consulta(id PK, paciente_id FK, medico_id FK, inicio TIMESTAMP, fim TIMESTAMP, status)

CRUD:

- **Create:** POST /pacientes, POST /medicos, POST /consultas
- **Read:** GET /consultas?medicoId=...&de=...&ate=...
- **Update:** PUT /consultas/{id} (alterar horário/status)
- **Delete:** proibir (usar status=CANCELADA)

Regras:

1. Bloquear conflito de horário por medico_id (overlap).
2. status ∈ {AGENDADA, REALIZADA, CANCELADA}.
3. fim > inicio.
4. Paciente e médico devem existir.
5. Cancelamento até X horas antes (config).

6. Não permitir REALIZADA voltar para AGENDADA.
7. Consulta dura de 15/30 min.
8. Índices por medico_id, inicio.
9. Transação ao agendar (valida + insere).

---

7) Kanban de Tarefas (Projetos)

Contexto: tarefas em colunas (TODO, DOING, DONE) por projeto.

Tabelas:

- projeto(id PK, nome UNIQUE, ativo BOOLEAN)
- tarefa(id PK, projeto_id FK, titulo, descricao, status, prioridade, criado_em, atualizado_em)

CRUD:

- **Create:** POST /projetos, POST /tarefas
- **Read:** GET /tarefas?projetoId=...&status=...
- **Update:** PUT /tarefas/{id}
- **Delete:** DELETE /tarefas/{id} (lógica: mover para status=ARCHIVED ou ativo=false via coluna extra)

Regras:

1. status ∈ {TODO, DOING, DONE, ARCHIVED}.
2. Não mover para DONE sem descricao mínima.
3. prioridade ∈ {BAIXA, MEDIA, ALTA}.
4. Limite WIP: máx. N tarefas em DOING por projeto.
5. projeto.ativo=false bloqueia novas tarefas.
6. Reordenar por atualizado_em desc nas listagens.
7. Pesquisa full-text por titulo/descricao (opcional).
8. Não apagar fisicamente tarefas DONE (histórico).
9. Métricas: lead time (criado_em→DONE).

---

8) E-Learning: Cursos e Matrículas

Contexto: cursos, alunos e matrículas com progresso.

Tabelas:

- curso(id PK, codigo UNIQUE, titulo, carga_horaria INT, ativo BOOLEAN)

- aluno(id PK, nome, email UNIQUE)
- matricula(id PK, aluno_id FK, curso_id FK, status, progresso INT, criado_em)

CRUD:

- **Create:** POST /cursos, POST /alunos, POST /matriculas
- **Read:** GET /matriculas?alunoId=...&status=...
- **Update:** PUT /matriculas/{id} (status/progresso)
- **Delete:** proibido (cancelar).

Regras:

1. status ∈ {ATIVA, CONCLUIDA, CANCELADA}.
2. progresso 0–100; CONCLUIDA exige progresso=100.
3. Um aluno não pode ter mais de uma matrícula ATIVA no mesmo curso.
4. Cursos inativos não aceitam novas matrículas.
5. Cancelamento bloqueia aumento de progresso.
6. codigo de curso único e imutável.
7. Índices por (aluno_id, curso_id) e status.
8. Atualizar criado_em apenas na criação.

---

9) Estoque & Entradas/Saídas

Contexto: controle de movimentações de estoque por item.

Tabelas:

- item(id PK, codigo UNIQUE, nome, estoque INT, ativo BOOLEAN)
- movimentacao(id PK, item_id FK, tipo, quantidade INT, motivo, criado_em)

CRUD:

- **Create:** POST /itens, POST /movimentacoes
- **Read:** GET /itens?ativo=true, GET /movimentacoes?itemId=...
- **Update:** PUT /itens/{id} (nome/ativo), movimentação não atualiza (imutável)
- **Delete:** proibir; reversão via nova movimentação.

Regras:

1. tipo ∈ {ENTRADA, SAIDA}.
2. quantidade > 0.
3. Em SAIDA, verificar estoque >= quantidade.

4. Atualizar item.estoque dentro de transação.
5. codigo único e imutável.
6. Itens inativos não aceitam SAIDA (apenas ENTRADA para acerto).
7. Motivo obrigatório em SAIDA.
8. Extrato por período e por item.
9. Bloquear estoque negativo.

---

10) Suporte: Tickets & Interações

Contexto: registrar tickets de suporte e interações de atendimento.

Tabelas:

- ticket(id PK, protocolo UNIQUE, cliente_email, assunto, status, prioridade, criado_em, atualizado_em)
- interacao(id PK, ticket_id FK, autor, mensagem, criado_em)

CRUD:

- **Create:** POST /tickets, POST /tickets/{id}/interacoes
- **Read:** GET /tickets?status=ABERTO&prioridade=ALTA, GET /tickets/{id}/interacoes
- **Update:** PUT /tickets/{id} (status/prioridade/assunto)
- **Delete:** proibir; fechar (status=FECHADO)

Regras:

1. status ∈ {ABERTO, EM_ATENDIMENTO, AGUARDANDO_CLIENTE, FECHADO}.
2. prioridade ∈ {BAIXA, MEDIA, ALTA, CRITICA}.
3. protocolo gerado e único (ex.: YYYYMMDDHHMMSSNNN).
4. Primeiro comentário automático ao abrir (template).
5. Fechado não recebe novas interações.
6. Alterar status exige registro de interação de sistema.
7. SLA opcional por prioridade (validar prazos).
8. Filtro padrão oculta FECHADO (usar ?incluirFechados=true).