Requisitos funcionais e não funcionais

**Tabela Receita:**

**Requisitos Funcionais**

Os **requisitos funcionais** descrevem o que o sistema **deve fazer**, ou seja, as funcionalidades e comportamentos esperados relacionados à tabela Receita. Com base na estrutura fornecida, podemos inferir os seguintes:

1. **Cadastro de receita médica**
   1. O sistema deve permitir o **registro de uma nova receita médica** com as informações necessárias (remédio, paciente, funcionário, data, etc).
2. **Relacionamento com outras entidades**
   1. O sistema deve garantir que cada receita esteja **vinculada a um remédio existente**.
   2. O sistema deve garantir que cada receita esteja **vinculada a um paciente existente**.
   3. O sistema deve garantir que cada receita esteja **vinculada a um funcionário existente** (provavelmente o profissional que emitiu a receita).
3. **Armazenamento de informações da receita**
   1. O sistema deve armazenar os seguintes dados para cada receita:
      1. Data de emissão
      2. Quantidade
      3. Dosagem
      4. Período de consumo
      5. Unidade Federativa (UF) do paciente e do funcionário
4. **Consulta de receitas**
   1. O sistema deve permitir **consultar receitas** emitidas com base em diferentes critérios (por paciente, por funcionário, por data, por UF, etc).

**Requisitos Não funcionais**

Os **requisitos não funcionais** descrevem **como** o sistema deve se comportar — aspectos de qualidade, desempenho, segurança, etc. Com base na tabela, podemos inferir os seguintes:

1. **Integridade referencial**
   1. A integridade dos dados deve ser mantida por meio de **chaves estrangeiras** para garantir que remédio, paciente e funcionário existam antes da criação de uma receita.
2. **Consistência dos dados**
   1. Campos obrigatórios (NOT NULL) garantem que **não haja dados incompletos** nas receitas cadastradas.
3. **Padronização**
   1. Campos como UF\_paciente e UF\_funcionario usam **CHAR(2)** para manter o padrão de siglas de estados brasileiros (ex: SP, RJ), garantindo consistência.
4. **Desempenho e escalabilidade**
   1. O uso de tipos de dados adequados como INT, DATE, VARCHAR ajuda a manter o desempenho de armazenamento e consulta de dados.
5. **Segurança e controle de acesso**
   1. Embora não esteja diretamente visível no SQL, em um sistema completo, o acesso à inserção ou consulta de receitas deve ser **controlado por permissões de usuário**, especialmente por se tratar de informações sensíveis (dados de saúde).
6. **Validação de dados**
   1. A estrutura exige que campos como dosagem, quantidade e período sejam preenchidos corretamente (espera-se que o sistema tenha validações de entrada para isso).

**Tabela Remédio:**

**Requisitos Funcionais**

Os **requisitos funcionais** indicam o que o sistema deve fazer — neste caso, as funcionalidades relacionadas à tabela Remedio.

1. **Cadastro de remédio**
   1. O sistema deve permitir o **registro de um novo remédio** com nome, quantidade ingerida e o paciente associado.
2. **Associação de remédio a paciente**
   1. Cada remédio deve estar **vinculado a um paciente específico**.
   2. O sistema deve garantir que o paciente associado ao remédio **exista previamente** (via chave estrangeira).
3. **Armazenamento de informações do remédio**
   1. O sistema deve armazenar:
      1. Nome do remédio
      2. Quantidade ingerida
      3. ID do paciente
4. **Consulta de remédios**
   1. O sistema deve permitir **consultar os remédios registrados** por nome, paciente ou outros filtros.
5. **Controle de nomes de remédios**
   1. O sistema deve garantir que **não existam remédios com nomes duplicados** (via restrição UNIQUE em nome\_remedio).

**Requisitos Não Funcionais**

Os **requisitos não funcionais** tratam da qualidade, integridade e desempenho do sistema com base na estrutura da tabela.

1. **Integridade referencial**
   1. A chave estrangeira id\_paciente assegura que **um remédio só possa ser associado a um paciente existente**, mantendo a integridade dos dados.
2. **Consistência dos dados**
   1. A presença de NOT NULL em todos os campos garante que **nenhum campo essencial seja deixado em branco**.
3. **Unicidade do nome de remédio**
   1. A restrição UNIQUE em nome\_remedio garante que **cada nome de remédio seja único** na base, evitando ambiguidades.
4. **Padronização de dados**
   1. O uso de VARCHAR(100) para textos e INT para identificadores segue boas práticas de modelagem de dados.
5. **Validação de dados na aplicação**
   1. A aplicação deve validar que o campo quantidade\_ingerida receba um valor coerente (ex: "2 comprimidos por dia"), já que não há controle semântico direto via SQL (por ser VARCHAR).
6. **Desempenho e escalabilidade**
   1. A tabela é simples e bem indexada (PK e UNIQUE), o que contribui para **desempenho em consultas e inserções**.
7. **Segurança e privacidade**
   1. Em um sistema completo, o acesso às informações de remédios deve ser **restrito a usuários autorizados**, pois envolve dados sensíveis relacionados à saúde.