**Análise Geral do Modelo Conceitual**

* **Perspectiva do Usuário:**
  + O diagrama representa bem os conceitos que eu, como usuário, imagino para o sistema. Existem "Churrascos", "Participantes", "Compras", "Pagamentos" e uma "Prestação de Contas". As informações dentro de cada caixa fazem sentido, como o "Churrasco" ter um "título" e "data", e a "Compra" ter um "valor" e "item". Fica claro que um churrasco é meu (Usuario -> Churrasco) e que ele terá vários participantes e várias compras.
* **Perspectiva do Projetista:**
  + Este diagrama serve como um excelente ponto de partida para o design do banco de dados e para a criação das classes do modelo de domínio (Models). As classes e seus atributos estão bem definidos.
  + As relações de multiplicidade (os números como 1, 0..\*, 1..\*) são cruciais, pois definem as regras de negócio e as chaves estrangeiras no banco de dados. Por exemplo, a relação Churrasco (1) --- (0..\*) Compra indica que uma compra deve, obrigatoriamente, pertencer a um churrasco.
  + **Pontos de Melhoria Identificados:**
    - **Relação Usuario vs. Participante:** Há uma lacuna crítica. A classe Usuario (quem usa o sistema) não tem uma ligação direta com a classe Participante (quem vai ao churrasco). Um usuário logado *é* um participante. O modelo precisa de uma associação clara entre Usuario e Participante (provavelmente uma relação 1 para 1) para saber qual usuário corresponde a qual participante nos eventos.
    - **Relação Pagamento:** O Pagamento está ligado ao Participante (0..1 --- 0..1). Isso é problemático, pois um participante pode ir a vários churrascos e fazer vários pagamentos. O ideal seria associar o Pagamento à classe Participacao, pois o pagamento é referente à participação de uma pessoa em um evento específico.
    - **Relação ParametroGlobal:** A multiplicidade Churrasco (1) --- (0..\*) ParametroGlobal está invertida. Ela sugere que um churrasco tem vários parâmetros globais, quando na verdade um parâmetro global (ex: "gramas de carne por pessoa") deveria se aplicar a vários churrascos. Esta classe deveria ser uma tabela de configuração global, não diretamente ligada a uma instância de Churrasco.
* **Perspectiva do Testador:**
  + Este diagrama é a base para o planejamento de testes de dados e de integração. Ele define as regras que os dados devem seguir.
  + **Testes de Restrição (Multiplicidade):** Com base no diagrama, podemos criar testes para validar as regras de negócio:
    - Tentar criar uma Compra sem associá-la a um Churrasco (deve falhar).
    - Tentar criar um Carrinho com zero ItemCarrinho (deve falhar, pois a multiplicidade é 1..\*).
    - Tentar apagar um Churrasco e verificar se todas as suas Compras, Participacoes, etc., são removidas ou tratadas corretamente (teste de exclusão em cascata).
  + **Testes de Dados:** Para cada classe, criar testes que verifiquem se todos os atributos (titulo, valor, data, etc.) aceitam e armazenam os tipos de dados corretos.

**Análise por Entidade/Classe**

**Churrasco**

* **Usuário:** É o meu evento. Contém todas as informações principais que eu preciso para identificá-lo: título, data, local e o valor da entrada.
* **Projetista:** É a entidade central do sistema. Possui relacionamentos com quase todas as outras classes. O atributo status (booleano) é importante para controlar o ciclo de vida do evento (ex: Aberto/Encerrado).
* **Testador:** Criar testes CRUD completos para a entidade Churrasco. Validar que ao criar um churrasco, ele é corretamente associado ao Usuario criador. Testar a alteração do status e verificar se as regras de negócio associadas (ex: impedir edição após encerramento) são aplicadas.

**Participacao e PreferenciaConsumo**

* **Usuário:** Representa a minha participação ou a de um amigo em um churrasco. É útil que o sistema saiba minha origem ("Cadastrado" ou "Email") e se eu já fiz o check-in. As PreferenciasConsumo são ótimas para que o organizador saiba se sou vegetariano ou não bebo álcool.
* **Projetista:** Participacao funciona como uma classe associativa (ou "tabela de junção com atributos") entre Churrasco e Participante. PreferenciaConsumo é uma extensão opcional (0..1) de Participacao, o que está bem modelado, pois nem todos os participantes informarão suas preferências.
* **Testador:** Criar um churrasco e adicionar participantes com e sem preferências de consumo. Validar que o status do convite e do check-in podem ser atualizados independentemente. Testar o caso em que um participante é removido do churrasco para garantir que o registro de Participacao seja excluído.

**Compra e Comprovante**

* **Usuário:** Aqui eu registro meus gastos. Para cada Compra, posso opcionalmente adicionar um Comprovante (a foto da nota fiscal), o que é essencial para a transparência.
* **Projetista:** A relação de Compra (1) para Comprovante (0..\*) está correta, permitindo múltiplos comprovantes por compra (ex: nota fiscal e recibo do cartão). A classe Comprovante armazena metadados do arquivo (nomeArquivo, tipo, tamanho); o arquivo em si deve ser armazenado em um serviço de storage, não no banco de dados.
* **Testador:** Testar o registro de uma compra. Em seguida, testar a adição de zero, um e múltiplos comprovantes a essa compra. Validar que, ao excluir uma compra, os comprovantes associados também sejam tratados (excluídos ou arquivados).

**PrestacaoConta e Relatorio**

* **Usuário:** No final do evento, quero ver a PrestacaoConta, que me mostra o resumo financeiro. Também quero poder gerar um Relatorio para talvez imprimir ou salvar um registro do que aconteceu.
* **Projetista:** PrestacaoConta parece ser uma entidade que armazena o "snapshot" do resultado financeiro em um determinado momento (parcial ou final), o que é um bom design para evitar recálculos constantes. Relatorio é uma entidade que representa uma geração de documento, armazenando seu tipo e formato, o que também é uma abordagem sólida.
* **Testador:** Criar um cenário com pagamentos confirmados e despesas registradas. Executar a funcionalidade de consolidar contas e validar se os valores na entidade PrestacaoConta (totalArrecadado, totalGasto) estão corretos. Testar a geração de Relatorio e verificar se a entidade é criada com os metadados corretos.

––