Modelo de design e modelagem

Para este modelo, atente-se muito mais à estrutura do que aos tópicos em si, pois eles podem variar muito de aplicação para aplicação. Não se preocupem com o tamanho do documento, é importante que ele esteja claro, coeso e correto.

1. Visão Geral da Arquitetura

Defina as fundações tecnológicas do sistema.

- **Padrão Arquitetural:** Ex: Monolito Modular, Microservices, Serverless, Layered Architecture.
    - **Justificativa:** Por que escolheu este padrão? Lembrem-se que o Django detém uma arquitetura também!.
- **Linguagem & Framework:** Ex: Java 17 com Spring Boot 3.0.
- **Banco de Dados:** Ex: PostgreSQL 15 (Relacional) para dados transacionais.
- **Segurança:** Ex: Spring Security com OAuth2 e JWT.

2. Diagramas

Insira aqui os diagramas de casos de uso, classes, e quaisquer outros que façam sentido para sua aplicação.

3. Design da API (Contrato de Interface)

Este é o contrato entre o Back-End e quem for consumi-lo. A consistência aqui é vital.

Padrões Globais

- **Formato de Dados:** JSON (application/json).
- **Tratamento de Erros:** Todas as respostas de erro seguirão o padrão RFC 7807

ou formato proprietário padronizado: { "timestamp": "...", "status": 400, "error": "...", "message": "..." }

Definição de Endpoints (Exemplo - Replique para cada recurso)

Recurso: Produtos (/products)

Método	Rota	Descrição	Auth Necessária?
GET	/api/v1/products	Lista produtos com paginação.	Não (Público)
POST	/api/v1/products	Cria novo produto.	Sim (Admin)
GET	/api/v1/products/{id}	Detalhes de um produto.	Não (Público)

Detalhe do Endpoint: Criar Produto (POST)

- **Request Body (Entrada):**

JSON

```
{
  "name": "Smartphone X (Obrigatório, min 3 chars)",
  "price": 1999.90 (Obrigatório, positivo),
  "sku": "SP-X-128 (Obrigatório, único)",
  "categoryId": 10
```

}

- **Response (Sucesso - 201 Created):**

JSON

```
{
  "id": 505,
  "name": "Smartphone X",
  "createdAt": "2023-10-27T10:00:00Z"
}
```

- **Códigos de Status Possíveis:**
    - 201 Created: Sucesso.
    - 400 Bad Request: Erro de validação (ex: preço negativo).
    - 401 Unauthorized: Token ausente ou inválido.
    - 403 Forbidden: Usuário logado mas não é Admin.
    - 409 Conflict: SKU já existente.

4. Modelagem de Dados (Schema)

Defina como os dados serão armazenados. Essencial para garantir integridade.

Diagrama ER

(Insira aqui a imagem do seu Diagrama Entidade-Relacionamento ou link para ferramenta)

Dicionário de Dados (Principais Tabelas)

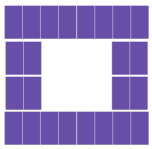
Tabela: tb_orders (Pedidos)

Coluna	Tipo de Dado	Restrições (Constraints)	Descrição

id	UUID ou BIGINT	PK, NOT NULL	Identificador único.
user_id	UUID ou BIGINT	FK, NOT NULL	Referência ao usuário (tb_users).
total_amount	DECIMAL(10,2)	NOT NULL, CHECK > 0	Valor monetário. **NUNCA usar FLOAT/DOUBLE.**
status	VARCHAR(20)	DEFAULT 'PENDING'	Enum: PENDING, PAID, SHIPPED, CANCELED.
created_at	TIMESTAMP	DEFAULT NOW()	Data de criação do registro.

5. Plano de Testes [OPCIONAL]

- **Testes Unitários:** Foco nas Regras de Negócio e Services. Meta: > 70% de cobertura.



DO NOT TOUCH THE PORTAL!