**Arquitetura de Domínio do Aplicativo “Ponto Certo”**

**Etapa 1 — Domínio Principal**

O domínio principal do aplicativo **Ponto Certo** é a **Gestão de Reservas de Lugares**.  
O sistema tem como propósito central **facilitar a reserva e o gerenciamento de assentos ou espaços disponíveis em eventos, salas ou locais específicos**, garantindo controle de disponibilidade em tempo real e uma experiência prática para o usuário.  
O público-alvo são **usuários que desejam reservar lugares com antecedência** e **administradores que controlam o espaço e as reservas**.  
As principais regras envolvem **bloqueio automático de lugares reservados, limitação de capacidade, confirmação e cancelamento de reservas**.

**Etapa 2 — Subdomínios**

| **Tipo** | **Subdomínio** | **Função** |
| --- | --- | --- |
| **Principal** | Gestão de Reservas | Controla criação, atualização, cancelamento e status de reservas |
| **Suporte** | Gestão de Lugares | Mantém o cadastro e disponibilidade dos assentos ou espaços |
| **Suporte** | Relatórios e Administração | Permite ao administrador visualizar ocupação, histórico e estatísticas |
| **Genérico** | Autenticação | Gerencia login, cadastro e perfis de usuários |
| **Genérico** | Notificações | Envia confirmações e lembretes de reserva por e-mail ou sistema |

**Etapa 3 — Linguagem Ubíqua (Glossário do Sistema)**

| **Termo** | **Significado** |
| --- | --- |
| **Reserva** | Ação de garantir um ou mais lugares disponíveis em determinado local ou evento |
| **Lugar** | Assento, mesa, sala ou espaço físico que pode ser reservado |
| **Usuário** | Pessoa cadastrada que realiza reservas no sistema |
| **Administrador** | Usuário com permissões para gerenciar lugares, reservas e visualizar relatórios |
| **Disponibilidade** | Estado que indica se um lugar está livre, reservado ou bloqueado |
| **Evento** | Contexto em que os lugares estão organizados (ex: show, palestra, cinema) |

**Etapa 4 — Entidades, Value Objects e Agregados**

**Entidades (possuem identidade própria):**

* **Usuário** → identificado por id\_usuario, representa quem faz as reservas.
* **Lugar** → identificado por id\_lugar, representa um espaço físico.
* **Reserva** → identificada por id\_reserva, relaciona um usuário a um lugar em um evento ou data específica.
* **Evento** → identificado por id\_evento, agrupa os lugares disponíveis para reserva.

**Value Objects (existem apenas por seus atributos):**

* **DataHora** → representa o horário de início e fim da reserva.
* **StatusReserva** → indica o estado atual da reserva (ativa, cancelada, pendente).

**Agregado principal:**

* **Agregado “Reserva”**, cuja **raiz é a entidade Reserva**, pois ela conecta as entidades **Usuário**, **Lugar** e **Evento**, controlando as regras de disponibilidade e confirmação.

*(Um diagrama simples mostraria: Usuário → Reserva → Lugar → Evento.)*

**Etapa 5 — Boas Práticas Arquiteturais**

O **Ponto Certo** segue uma **arquitetura em camadas inspirada no DDD (Domain-Driven Design)**:

* **Camadas:**
  + /domain → contém entidades, regras e interfaces de repositórios.
  + /application → casos de uso como criar, cancelar e listar reservas.
  + /infra → implementa persistência no banco (ex: Sequelize ou Prisma).
  + /presentation → controllers, rotas e front-end (React ou PHP).
* **Princípios SOLID** aplicados:
  + **Single Responsibility:** cada classe tem uma função específica (ex: ReservaService apenas lida com regras de reserva).
  + **Dependency Inversion:** o domínio não depende diretamente de frameworks ou banco.
* **Bounded Contexts:**
  + “Gestão de Reservas” separado de “Autenticação”, evitando acoplamento.
  + “Gestão de Lugares” tratado como um subdomínio independente, mas integrado via API interna.
* **Independência tecnológica:** o domínio é testável sem depender do banco ou do front-end.
* **Testabilidade:** regras de disponibilidade e conflito de horários podem ser testadas isoladamente.

**Etapa 6 — Reflexão Final**

Modelar corretamente o domínio do **Ponto Certo** é essencial para garantir que o sistema cresça de forma organizada e fiel às regras do negócio.  
Ao compreender bem os relacionamentos entre **reservas, usuários e lugares**, o desenvolvimento se torna mais previsível e sustentável.  
Essa abordagem permite evoluir o projeto — por exemplo, adicionando pagamentos, filas de espera ou integração com eventos — sem quebrar a estrutura existente.  
Em resumo, um domínio bem modelado aumenta a **qualidade, manutenibilidade e longevidade** do sistema