1. Visão geral do fluxo (o que o cliente faz no site)
2. Como o administrador controla dias e horários (o que ele faz na interface)
3. Como o banco de dados deve ser (tabelas e colunas) — explicado com exemplos simples
4. Passo a passo real, do clique do cliente até a reserva gravada — mostrando as checagens que ocorrem

**1) Visão geral — o que o cliente vê e faz**

Imagine a tela do site para reservar:

1. O cliente abre um **calendário mensal** (um quadrado por dia).
2. Ele clica em um dia (ex.: **15 de setembro**).
3. O site mostra **uma lista de horários** desse dia (ex.: 17:00, 18:00, 19:00).
4. Cada horário aparece como **Disponível**, **Ocupado** ou **Fechado**.
   * Disponível = pode reservar.
   * Ocupado = já tem reserva.
   * Fechado = o administrador bloqueou (por exemplo, feriado ou manutenção).
5. O cliente escolhe um horário disponível e confirma.
6. O sistema tenta gravar a reserva; se der certo, mostra “Reserva confirmada”; se outro cliente tinha acabado de reservar o mesmo horário, ele mostra “Desculpe — horário já ocupado”.

**2) O que o administrador faz (interface simples)**

No painel do administrador você oferece ações básicas:

* **Definir horário padrão da semana** (por exemplo: de segunda a sexta das 08:00 às 22:00, com blocos de 60 minutos). Isso é a “regra normal”: diz quais horários a quadra costuma estar aberta.
* **Fechar um dia inteiro** (ex.: feriado no dia 20/09) — o administrador clica numa data no calendário e marca “fechado”. Aquele dia fica fechado para todas as reservas.
* **Bloquear um horário específico** (ex.: 15/09 às 18:00 está reservado para um evento ou manutenção) — o administrador clica no horário e marca “bloquear”.
* **Desbloquear** ou **reabrir** quando quiser.

Essas ações ficam registradas no sistema e afetam o que os clientes veem no calendário e na lista de horários.

**3) O banco de dados — tabelas essenciais e como elas guardam essas informações**

Pense no banco como um conjunto de fichas. Cada tabela é uma ficha com colunas.

Vou listar as tabelas e explicar com exemplos práticos.

**Tabela 1 — usuario**

Guarda pessoas (clientes e administradores).  
Colunas principais:

* id — número da pessoa
* nome
* telefone
* email
* tipo — indica se é **A** (admin) ou **C** (cliente)

**Exemplo de registro:**  
(id=10, nome="João", telefone="11999999999", tipo="C")

**Tabela 2 — quadra**

Guarda a quadra (nome, endereço, preço por hora).  
Colunas:

* id, nome, endereco, preco\_hora, ativo

**Exemplo:**  
(id=1, nome="Quadra Principal", preco\_hora=80.00)

**Tabela 3 — disponibilidade\_semana (regra padrão)**

Guarda o horário padrão por dia da semana — por exemplo, toda segunda das 08:00 às 22:00 com duração dos blocos (60 min).

Colunas:

* id
* quadra\_id (qual quadra)
* dia\_semana (0=domingo … 6=sábado)
* hora\_inicio (ex.: "08:00")
* hora\_fim (ex.: "22:00")
* duracao\_min (ex.: 60)

**Exemplo:** se a quadra funciona de segunda a sexta:

* (quadra\_id=1, dia\_semana=1, hora\_inicio="08:00", hora\_fim="22:00", duracao\_min=60) — isto significa blocos 08:00,09:00,...21:00 nas segundas.

Essa tabela não lista todos os horários de cada dia — ela define o **padrão** de funcionamento semanal.

**Tabela 4 — dia\_fechado**

Guarda dias em que a quadra está totalmente fechada (feriado, evento).  
Colunas:

* id, quadra\_id, data\_fechada (ex.: "2025-09-15"), motivo

**Exemplo:**  
(id=1, quadra\_id=1, data\_fechada="2025-09-10", motivo="Evento privado")

Se existir esse registro para uma data, **nenhum horário desse dia estará disponível**, mesmo que a disponibilidade\_semana diga que normalmente estaria aberta.

**Tabela 5 — horario\_bloqueado**

Guarda bloqueios pontuais de horários (manutenção, reserva exclusiva).  
Colunas:

* id, quadra\_id, horario (data e hora ex.: "2025-09-15 18:00"), motivo, criado\_por (id do admin)

**Exemplo:**  
(quadra\_id=1, horario="2025-09-15 18:00", motivo="Treino privado")

Se existir esse registro para um horário, **mesmo que o dia não esteja fechado e o horário seja normalmente permitido**, ele fica bloqueado.

**Tabela 6 — reserva**

Guarda as reservas confirmadas.  
Colunas:

* id
* quadra\_id
* usuario\_id (cliente)
* data\_hora (ex.: "2025-09-15 18:00") — inicio do horário reservado
* duracao\_min (ex.: 60)
* preco (valor cobrado no momento)
* status (confirmada, cancelada)
* criado\_em

**IMPORTANTE para evitar dupla reserva:** no banco a tabela reserva terá uma **regra** que não permite duas linhas com a mesma quadra\_id + data\_hora. Ou seja, **é proibido gravar duas reservas para a mesma quadra exatamente na mesma data e hora**. Essa regra é automática no banco e protege contra conflitos.

**Exemplo:**  
Se já existir (quadra\_id=1, data\_hora="2025-09-15 18:00") em reserva, ninguém mais pode criar outra reserva com esses mesmos valores.

**Tabela 7 — LOG**

Guarda o histórico de mudanças (INSERT, UPDATE, DELETE).  
Colunas:

* id, datetime\_log, tipo\_evento(INSERT, UPDATE...), id\_usuario, detalhes, valores\_old e valores\_new

**4) Passo a passo prático — do clique do cliente até o banco, com exemplos**

Vou usar um exemplo real: **15/09/2025**, horários 17:00 e 18:00. A quadra normalmente funciona 08:00–22:00 com blocos de 60min.

**Cenário A — mostrar horários para o cliente (o que acontece quando ele clica no dia)**

1. O site pede ao servidor: “Quais horários existem para 15/09/2025?”
2. O servidor (no código) faz isso, nesta ordem:
   * Verifica se existe **dia\_fechado** para 2025-09-15. Se existir, devolve: dia fechado → não mostra horários.
   * Se não estiver fechado, pega o disponibilidade\_semana para o dia da semana (por exemplo, segunda) e **calcula** os horários possíveis: 08:00, 09:00, 10:00 ... até 21:00.
   * Para cada horário calculado, checa se existe um horario\_bloqueado para aquele horário exato → se sim, marca como **Fechado**.
   * Também checa na tabela reserva se já existe reserva confirmada para aquele horário → se sim, marca como **Ocupado**.
   * Caso contrário, marca como **Disponível**.
3. O servidor retorna a lista de horários com os estados (Disponível / Ocupado / Fechado) e o preço para cada um. O site mostra essa lista ao cliente.

**Exemplo de retorno que o site pode receber:**

* 17:00 — Disponível — R$80
* 18:00 — Ocupado — R$80
* 19:00 — Fechado (manutenção) — R$80

**Cenário B — cliente escolhe um horário e clica “Reservar”**

1. O cliente escolhe 17:00 e envia nome/telefone.
2. O servidor executa estas verificações ( na mesma ordem de antes, para garantir que nada mudou entre mostrar e clicar):
   * Verifica dia\_fechado para a data → se existir, retorna erro: dia fechado.
   * Verifica horario\_bloqueado para 2025-09-15 17:00 → se existir, retorna erro: horário bloqueado.
   * Verifica se o horário está dentro do disponibilidade\_semana (ou seja, se 17:00 é um horário permitido naquele dia) → se não estiver, retorna erro.
3. Se todas as checagens passarem, o servidor **tenta gravar a reserva** no banco:
   * Ele insere um registro em reserva com quadra\_id=1 e data\_hora="2025-09-15 17:00".
   * O banco tem a regra que impede duplicidade. Se alguém tiver gravado a mesma reserva no intervalo entre a checagem e a tentativa de inserir, o banco **vai recusar** a segunda inserção.
   * Se o insert for aceito, servidor devolve “Reserva confirmada”.
   * Se o banco recusar por duplicidade, servidor devolve “Desculpe — horário já ocupado” (o cliente escolhe outro horário).

Essa lógica garante que **mesmo se duas pessoas clicarem no mesmo horário ao mesmo tempo**, apenas uma reserva será gravada — o banco protege isso.

**5) Como o admin bloqueia dias/horários (na prática)**

**Fechar um dia inteiro**

* No painel, admin escolhe a data (ex.: 20/09/2025) e clica “Fechar dia”.
* O sistema grava uma linha em dia\_fechado com data\_fechada = '2025-09-20'.
* A partir desse momento, quando clientes pedirem horários para 20/09/2025, o servidor retorna “dia fechado” e não mostra horários.

**Bloquear um horário específico**

* No painel, admin vê a lista de horários do dia (por ex., 15/09) e clica em 18:00 -> “Bloquear horário”.
* O sistema grava em horario\_bloqueado a linha com horario = '2025-09-15 18:00'.
* A partir daí, esse horário aparecerá como “Fechado” para os clientes.

**Reabrir (desfazer)**

* Para reabrir, o admin remove o registro de dia\_fechado ou horario\_bloqueado pelo painel, e o horário volta a aparecer como disponível (se não existir reserva).

**6) Prioridade final — o que vale mais**

Quando o sistema decide se um horário está disponível ele segue essa ordem (mais forte para menos forte):

1. Se existe dia\_fechado → dia encerrado (nenhum horário disponível).
2. Senão, se existe horario\_bloqueado para aquele horário → horário fechado.
3. Senão, se existe reserva naquele horário → ocupado.
4. Senão, se o horário **não** faz parte da disponibilidade\_semana (ou seja, fora do horário normal) → não disponível.
5. Senão → disponível.

**7) Resumo do que precisa existir no banco (sem termos técnicos demais)**

* Uma tabela com **pessoas** (clientes e admins).
* Uma tabela com **quadras** (nome, preço).
* Uma tabela que guarda o **horário padrão semanal** (por exemplo: segunda 08:00–22:00, bloco 60 min).
* Uma tabela que guarda **dias fechados** (datas inteiras que não aceitam reserva).
* Uma tabela que guarda **horários bloqueados pontuais** (data+hora específica).
* Uma tabela que guarda as **reservas** (quem, quando, preço, status).
* Opcional: uma tabela para **registro de ações** (quem bloqueou, quem cancelou, quando).

**8) Exemplo concreto completo (passo a passo, com valores)**

1. disponibilidade\_semana diz: sexta tem de 08:00 a 22:00, blocos de 60min.
2. dia\_fechado está vazio para 15/09/2025.
3. horario\_bloqueado tem uma linha só para 15/09/2025 19:00 (manutenção).
4. Cliente A abre o sistema, clica em 15/09 → servidor calcula horários e vê: 17:00 (livre), 18:00 (livre), 19:00 (bloqueado).
5. Cliente B e Cliente A tentam reservar 18:00 ao mesmo tempo. Ambos enviam pedido. O servidor primeiro valida e tenta gravar em reserva. O banco aceita o primeiro insert; o segundo insert é rejeitado por regra de não duplicidade; o segundo recebe mensagem de erro e escolhe outro horário.

**9) Dicas práticas para começar (MVP)**

* Implemente primeiro: usuario, quadra, disponibilidade\_semana, reserva.
* Depois adicione dia\_fechado e horario\_bloqueado para o admin controlar exceções.
* Para a disponibilidade semanal, faça o cálculo dos horários no backend quando o cliente pedir um dia — isso é simples e evita ter que criar todas as linhas de horários no banco.
* Sempre trate o caso em que o banco recusa uma inserção por duplicidade — devolva mensagem amigável ao cliente (“já reservado”).