Planejamento de projeto Lista de Compra.

Entender o que preciso para desenvolver o projeto de lista de compras, regras de negócio, funções básicas e modelagem de banco de dados.

- [x] ~~Estudo de caso.~~
- [x] ~~Arquitetura e tecnologia~~
- [x] ~~User Stories de Aceitação (Porque de cada função)~~
- [x] ~~Modelagem de banco de dados~~
- [x] ~~Modelagem relacional refinada~~
- [x] ~~Especificação de APIs (mesmo em local)~~
- [x] ~~Fluxo de navegação e wireframes ok irei fazer~~

ESTUDO DE CASO E MODELAGEM DO BANCO DE DADOS INICIO

Usuários: Eu e a Raquel.

Basicamente, verificamos o que temos em casa e o que está em falta, anotamos o que está em falta em um bloco de notas ou no WhatsApp. Nessa lista, separamos os produtos por listas , como, por exemplo: produtos de limpeza, higiene, mistura, café da manhã, etc. Cada produto também colocamos uma quantidade mínima ao comprar, que seria a quantidade que usamos. Então, vamos ao mercado e fazemos compras seguindo a lista do que está em falta e, depois, outra lista menos importante de guloseimas e coisas para beliscar, como petiscos e coisas gostosas. Durante as compras, pegamos os produtos com os preços mais vantajosos, porém descartamos algumas marcas que julgamos ruins ou de má qualidade. Também preferimos algumas

marcas que gostamos, se elas estiverem em um preço acessível. Durante as compras, calculamos o preço, com um teto de gasto que definimos antes.

Funções básicas (essenciais):

Criar lista por nome

- Ex: Limpeza, Higiene, Café da manhã, Mistura, Petiscos, etc.

1. **Adicionar produto**
    - Nome, quantidade mínima, marca preferida (opcional), marca descartada (opcional), Valor, Observação.
2. **Editar produto da lista**
    - Alterar nome, quantidade, marca preferida ou descartada, valor.
3. **Remover produto da lista**
4. **Marcar produto como "comprado"**
5. **Somar valor dos produtos comprados**
    - Exibe o subtotal gasto até agora.
6. **Exibir o total estimado da compra**
    - Soma os preços estimados dos itens que ainda faltam.
7. **Definir teto de gasto**
    - Mostra o quanto ainda pode ser gasto com base no teto e subtotal atual.

Funções adicionais (para ajudar sem complicar):

1. **Separar lista por prioridade**
    - ✅ Essenciais (usar como filtro)
    - 🍭 Opcionais (petiscos/guloseimas, só se sobrar dinheiro)
2. **Adicionar marca preferida ou marca descartada**
    - Ajuda na hora da decisão de compra
3. **Adicionar preço estimado por item**
    - Para calcular antes de sair de casa se vai dar certo no orçamento

4. **Adicionar observações rápidas por item**
    - Tipo "só se tiver em promoção" ou "procurar embalagem maior"
5. **Filtrar produtos por, prioridade**
6. **Salvar listas anteriores**
    - Para reaproveitar sem digitar tudo de novo
7. **Modo 'verificação de despensa'**
    - Marca o que tem em casa antes de montar a lista do que falta
8. **Modo "compra em andamento"**
    - Tela mais simples para ir riscando os itens no mercado sem se perder

---

Extras inteligentes (se quiser evoluir no futuro, ainda sem complicar):

1. **Sugestão automática do que comprar**
    - Com base no histórico de compras anteriores ou tempo desde a última compra
2. **Relatório de gastos por mês**
    - Só se quiser acompanhar o quanto gastou com alimentos, limpeza etc.
3. modo COOP
    - Conectar a outro Celular mesma lista de compra, via Bluetooth ou Wi-Fi

Requisito Funcional:

1. **RF:** Armazenar os dados dos usuários, ID e nome
2. **RF:** Armazenar os dados dos produtos, ID, nome, marca, se a marca é ruim, se a marca é boa, ou se não sabe, preço,, quantidade, status(comprado ou não), Observação
3. **RF:** Armazenar os dados da marca, ID, nome, tipo(preferida/descartada) Observacao
4. **RF:** Armazenar os dados da compra, ID compra, teto de gastos, valor da compra, data da compra.
5. **RF:** Armazenar as listas da lista, ID lista, nome da lista, prioridade ou supérflua.

Modelo Lógico de Dados

Entidades, atributos e relacionamentos com cardinalidades:

1. **USUÁRIO**

- PK: `usuario_id`
- `nome`

1. Lista

- PK: `lista_id`
- `nome`

1. **MARCA**

- PK: `marca_id`
- `nome`

1. **PRODUTO**

- PK: `produto_id`
- `nome`
- `quantidade_minima`
- FK: `pref_id` → lista(N:1)

1. **PREFERÊNCIA_MARCA**

- PK: `pref_id`
- FK: `produto_id` → PRODUTO (N:1)
- FK: `marca_id` → MARCA (N:1)
- `tipo` ENUM{“preferida”, “descartada” , “não sabe” }

> Cada produto pode ter zero ou várias preferências de marca; cada preferência aponta a apenas uma marca.

1. **LISTA**

- PK: `lista_id`
- `nome`
- FK: `usuario_id` → USUÁRIO (N:1)

- `prioridade` ENUM{“essencial”, “supérflua”}
- `data_criacao`

> Um usuário cria várias listas; cada lista pertence a um único usuário.

1. **LISTA_ITEM**

- PK: `item_id`
- FK: `lista_id` → LISTA (N:1)
- FK: `produto_id` → PRODUTO (N:1)
- `quantidade`
- `preco_estimado`
- `observacao` TEXT
- `status` ENUM{“pendente”, “comprado”}

> Cada lista contém vários itens; cada item refere-se a um único produto em uma única lista.

1. **COMPRA**

- PK: `compra_id`
- FK: `lista_id` → LISTA (1:1)
- `teto_gasto`
- `gasto_atual`
- `data_compra`

> Cada compra refere-se exatamente a uma lista finalizada.

1. **COMPRA_ITEM**

- PK: `compra_item_id`
- FK: `compra_id` → COMPRA (N:1)
- FK: `produto_id` → PRODUTO (N:1)

- FK: `marca_id` → MARCA (N:1) — marca escolhida
- `quantidade_comprada`
- `preco_real`

> Cada compra engloba vários itens; cada item de compra referencia um único produto e a marca adquirida.

Cardinalidades Resumidas

- USUÁRIO 1:N LISTA
- LISTA 1:N LISTA_ITEM
- PRODUTO 1:N LISTA_ITEM
- CATEGORIA 1:N PRODUTO
- PRODUTO 1:N PREFERÊNCIA_MARCA
- MARCA 1:N PREFERÊNCIA_MARCA
- LISTA 1:1 COMPRA
- COMPRA 1:N COMPRA_ITEM
- PRODUTO 1:N COMPRA_ITEM
- MARCA 1:N COMPRA_ITEM

ESTUDO DE CASO E MODELAGEM DO BANCO DE DADOS FIM

ARQUITETURA E TECNOLOGIA INICIO

React com Vite + SQLite via IndexedDB (local) - **Progressive Web App.**

ARQUITETURA E TECNOLOGIA FIM

User Story Inicio

User Story

1. **Como usuário**

quero **criar múltiplas listas de compra com nomes e categorias**,

para **organizar produtos conforme tipo e prioridade (ex: essenciais, guloseimas, limpeza etc.)**.

2. **Como usuário**,

quero **adicionar produtos às listas com nome, quantidade e categoria**,

para **controlar o que preciso comprar e quanto**.

4. **Como usuário**

quero **atribuir marcas preferidas e marcas descartadas aos produtos**

para **decidir rapidamente durante a compra e evitar más escolhas.**

5. Como usuário,

quero **atribuir preços estimados aos produtos**,

para **calcular o valor total antes da compra**.

6. Como usuário,

quero **marcar produtos como comprados**,

para **acompanhamento visual e controle do que falta pegar**.

7. **Como usuário**

quero **ver o subtotal dos itens comprados e o total estimado do que falta**

para **acompanhar o quanto já gastei e quanto ainda posso gastar**

8 . **Como usuário**

quero **adicionar observações aos produtos**

para **lembrar de condições específicas como “só se tiver promoção”**

9. **Como usuário**

quero **filtrar produtos por, prioridade ou status (comprado/faltando)**

para **facilitar a navegação durante a compra.**

10 . **Como usuário**

quero **ativar um modo de verificação de despensa**,

para **marcar o que já temos em casa antes de montar a lista de compras**.

11. **Como usuário**

quero **ativar um modo de compra simplificado**

para **rastrear os itens comprados de forma rápida no mercado.**

User Story Fim

1 Caso de uso: Adicionar Lista

1.ENDPOINT :

POST /api/listas

2.

Descrição: Cria uma lista, com um nome descritivo por exemplo: alimentos básicos, gluoseimas, produtos de limpeza etc, nessa lista irão ser adicionados os produtos, nessa lista terá um checklist, se é ou não lista de produtos essenciais.

loren ipsun...

3. **Parâmetros**

- `listaID` (na URL): ID da lista à qual será adicionado as listas.
- `Nome` Nome da Lista

4. Corpo da Requisição (JSON)

```
{
 "Nome da Lista": "Produtos Básicos",
  "essencial": true
```

```
}
```

5. Resposta de Sucesso

Código: `201 Created`

Corpo:

```
{
"id": 1,
"nome": "Produtos Basicos"
"Essencial": True
}
```

6. Erros Possíveis

- `400 Bad Request` → Campo obrigatório ausente ou mal formatado.

2 Listar todas as listas de compras

1. Endpoint

GET /api/listas

POST /api/listas
2.
Descrição: Retorna todas as listas de compra criadas pelo usuário, com seus atributos principais.

3.. **Parâmetros**

- `essencial` (boolean) — filtra apenas listas marcadas como essenciais.
- `pagina` (integer) — número da página (para paginação).
- `tamanho` (integer) — quantos itens por página.

4. Resposta de Sucesso

Código: `201 Created`

Corpo:

```
[
  {
    "id": 1,
    "nome": "Produtos Básicos",
    "essencial": true,
    "dataCriacao": "2025-06-07T10:15:30Z"
  },
  {
    "id": 2,
    "nome": "Guloseimas",
    "essencial": false,
    "dataCriacao": "2025-06-06T14:22:10Z"
  }
  // ...
]
```

6. Erros Possíveis

- `400 Bad Request` → parâmetro de filtro inválido (ex.: `essencial=foo`).
- `500 Internal Server Error` → erro inesperado no servidor.

3. Caso de Uso: Atualizar dados de uma lista (nome, prioridade/essencial)

1. **ENDPOINT:**

   PUT /api/listas/{lista_id}

2. **Descrição:** Atualiza o nome e/ou a flag de essencial de uma lista de compras existente.

3. **Parâmetros de URL:**

   - lista_id (integer): ID da lista a ser atualizada.

4. **Corpo da Requisição (JSON):**

   ```
   {
   "nome": "Compras Essenciais da Semana",
   ```

```
  "essencial": true
}
```

content_copydownload Use code **with caution**.Json

Obs: Pelo menos um dos campos (nome ou essencial) deve ser fornecido.

5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

```
{
  "id": 1,
  "nome": "Compras Essenciais da Semana",
  "essencial": true,
  "dataCriacao": "2025-06-07T10:15:30Z" // dataCriacao não muda, mas é bom retornar o objeto completo
}
```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 400 Bad Request → Nenhum campo fornecido para atualização ou formato inválido.
   - 404 Not Found → Lista com o lista_id fornecido não encontrada.
   - 500 Internal Server Error → Erro ao atualizar os dados.

4 Caso de Uso: Excluir lista de compras

1. **ENDPOINT:**

   DELETE /api/listas/{lista_id}

2. **Descrição:** Remove permanentemente uma lista de compras e todos os seus itens associados.
3. **Parâmetros de URL:**
   - lista_id (integer): ID da lista a ser excluída.
4. **Corpo da Requisição:**

   Nenhum.

5. **Resposta de Sucesso:**

   Código: 204 No Content

Corpo: Nenhum.

6. **Erros Possíveis:**
   - 404 Not Found → Lista com o lista_id fornecido não encontrada.
   - 500 Internal Server Error → Erro ao excluir a lista.

5 Caso de Uso: Adicionar produto a uma lista

1. **ENDPOINT:**

   POST /api/listas/{lista_id}/itens

2. **Descrição:** Adiciona um novo item (produto) a uma lista de compras específica. Se o produto base não existir na tabela PRODUTO, ele pode ser criado implicitamente ou o sistema pode exigir que produtos base sejam pré-cadastrados. Para este caso, vamos assumir que o produto_id refere-se a um produto já existente na tabela PRODUTO. A categoria_id também é de uma categoria existente.

3. **Parâmetros de URL:**
   - lista_id (integer): ID da lista à qual o produto será adicionado.

4. **Corpo da Requisição (JSON):**

```
{
 "produto_id": 101, // ID de um produto existente na tabela PRODUTO
 "quantidade": 2,
 "preco_estimado": 5.99, // Opcional
 "observacao": "Pegar o da embalagem azul, se possível", // Opcional
 "prioridade_item": "essencial" // "essencial" ou "superflua" para este item específico na lista
}
```

   content_copydownload Use code **with caution**.Json

   Nota: produto_id refere-se a PRODUTO.produto_id. O nome e categoria do produto já estão em PRODUTO.

5. **Resposta de Sucesso:**

   Código: 201 Created

   Corpo: (Retorna o item de lista criado)

```
{
 "item_id": 50, // Novo ID do LISTA_ITEM
 "lista_id": 1,
 "produto_id": 101,
```

```
  "produto_nome": "Leite Integral UHT", // Incluído para conveniência
  "categoria_nome": "Laticínios", // Incluído para conveniência
  "quantidade": 2,
  "preco_estimado": 5.99,
  "observacao": "Pegar o da embalagem azul, se possível",
  "status": "pendente", // Default ao adicionar
  "prioridade_item": "essencial"
}
```

content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
    - 400 Bad Request → Campo obrigatório ausente ou mal formatado (ex: produto_id, quantidade).
    - 404 Not Found → Lista com lista_id não encontrada, ou produto_id não encontrado na tabela PRODUTO.
    - 500 Internal Server Error → Erro ao adicionar o item.

6 Caso de Uso: Listar todos os produtos de uma lista

1. **ENDPOINT:**

   GET /api/listas/{lista_id}/itens

2. **Descrição:** Retorna todos os itens (produtos) de uma lista de compras específica, com opções de filtro.
3. **Parâmetros de URL:**
    - lista_id (integer): ID da lista.
4. **Parâmetros de Query (opcionais):**
    - categoria_id (integer): Filtra por ID da categoria do produto.
    - prioridade_item (string): Filtra por prioridade do item na lista (ex: "essencial", "superflua").
    - status (string): Filtra por status do item (ex: "pendente", "comprado").
5. **Resposta de Sucesso:**

Código: 200 OK

Corpo:

```
[
 {
  "item_id": 50,
  "lista_id": 1,
  "produto_id": 101,
  "produto_nome": "Leite Integral UHT",
  "categoria_id": 3,
  "categoria_nome": "Laticínios",
  "quantidade": 2,
  "preco_estimado": 5.99,
  "observacao": "Pegar o da embalagem azul",
  "status": "pendente",
  "prioridade_item": "essencial"
 },
 {
  "item_id": 51,
  "lista_id": 1,
  "produto_id": 102,
  "produto_nome": "Pão de Forma Integral",
  "categoria_id": 4,
  "categoria_nome": "Padaria",
  "quantidade": 1,
  "preco_estimado": 7.50,
  "observacao": null,
  "status": "comprado",
  "prioridade_item": "essencial"
 }
 // ... mais itens
]
```

content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 400 Bad Request → Parâmetro de filtro inválido.
   - 404 Not Found → Lista com lista_id não encontrada.
   - 500 Internal Server Error → Erro ao buscar os itens.

7 Caso de Uso: Atualizar produto de lista

1. **ENDPOINT:**

   PUT /api/listas/{lista_id}/itens/{item_id}

2. **Descrição:** Atualiza os dados de um item específico em uma lista de compras.

3. **Parâmetros de URL:**
    - lista_id (integer): ID da lista.
    - item_id (integer): ID do item da lista a ser atualizado.
4. **Corpo da Requisição (JSON):**

```
{
 // Não se altera produto_id ou categoria_id aqui, pois isso mudaria o "produto base".
 // Se precisar mudar o produto em si, seria uma remoção e adição de um novo item.
 "quantidade": 3, // Campo obrigatório para atualização do item
 "preco_estimado": 6.10, // Opcional
 "observacao": "Verificar validade!", // Opcional
 "prioridade_item": "superflua" // Opcional
 // O status (comprado/pendente) é tratado em endpoint específico (Marcar/desmarcar)
}
```

    content_copydownload Use code **with caution**.Json
5. **Resposta de Sucesso:**

    Código: 200 OK

    Corpo: (Retorna o item de lista atualizado)

```
{
 "item_id": 50,
 "lista_id": 1,
 "produto_id": 101,
 "produto_nome": "Leite Integral UHT",
 "categoria_id": 3,
 "categoria_nome": "Laticínios",
 "quantidade": 3,
 "preco_estimado": 6.10,
 "observacao": "Verificar validade!",
 "status": "pendente", // Status não é alterado aqui
 "prioridade_item": "superflua"
}
```

    content_copydownload Use code **with caution**.Json
6. **Erros Possíveis:**
    - 400 Bad Request → Campo obrigatório ausente ou mal formatado.
    - 404 Not Found → Lista ou item não encontrado.
    - 500 Internal Server Error → Erro ao atualizar o item.

8 Caso de Uso: Excluir produto de lista

1. **ENDPOINT:**

DELETE /api/listas/{lista_id}/itens/{item_id}

1. **Descrição:** Remove um item específico de uma lista de compras.
2. **Parâmetros de URL:**
   - lista_id (integer): ID da lista.
   - item_id (integer): ID do item da lista a ser removido.
3. **Corpo da Requisição:**

   Nenhum.
4. **Resposta de Sucesso:**

   Código: 204 No Content

   Corpo: Nenhum.
5. **Erros Possíveis:**
   - 404 Not Found → Lista ou item não encontrado.
   - 500 Internal Server Error → Erro ao remover o item.

---

9. Caso de Uso: Marcar/desmarcar produto como comprado

1. **ENDPOINT:**

   PATCH /api/listas/{lista_id}/itens/{item_id}/status
2. **Descrição:** Altera o status de um item (produto) em uma lista para "comprado" ou "pendente".
3. **Parâmetros de URL:**
   - lista_id (integer): ID da lista.
   - item_id (integer): ID do item da lista.
4. **Corpo da Requisição (JSON):**

   ```
   {
    "status": "comprado" // ou "pendente"
   }
   ```

   content_copydownload Use code **with caution**.Json
5. **Resposta de Sucesso:**

Código: 200 OK

Corpo: (Retorna o item de lista com status atualizado)

```
{
 "item_id": 50,
 "lista_id": 1,
 "produto_id": 101,
 "produto_nome": "Leite Integral UHT",
 "categoria_nome": "Laticínios",
 "quantidade": 3,
 "preco_estimado": 6.10,
 "observacao": "Verificar validade!",
 "status": "comprado",
 "prioridade_item": "superflua"
}
```

content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 400 Bad Request → Status inválido ou ausente.
   - 404 Not Found → Lista ou item não encontrado.
   - 500 Internal Server Error → Erro ao atualizar o status.

---

10. Caso de Uso: Obter subtotal de produtos comprados

1. **ENDPOINT:**

   GET /api/listas/{lista_id}/subtotal-comprados

2. **Descrição:** Calcula e retorna a soma dos preco_estimado (ou preco_real se já em modo compra e preenchido) dos itens marcados como "comprado" em uma lista.
3. **Parâmetros de URL:**
   - lista_id (integer): ID da lista.
4. **Corpo da Requisição:**

   Nenhum.

5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

```
{
 "lista_id": 1,
 "subtotal_comprados": 75.80 // Soma dos preços dos itens comprados
}
```

content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 404 Not Found → Lista não encontrada.
   - 500 Internal Server Error → Erro ao calcular o subtotal.

---

11. Caso de Uso: Obter total estimado de produtos pendentes

1. **ENDPOINT:**

   GET /api/listas/{lista_id}/total-estimado-pendentes

2. **Descrição:** Calcula e retorna a soma dos preco_estimado dos itens marcados como "pendente" em uma lista.
3. **Parâmetros de URL:**
   - lista_id (integer): ID da lista.
4. **Corpo da Requisição:**

   Nenhum.

5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

```
{
 "lista_id": 1,
 "total_estimado_pendentes": 120.50 // Soma dos preços estimados dos itens pendentes
}
```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 404 Not Found → Lista não encontrada.
   - 500 Internal Server Error → Erro ao calcular o total estimado.

---

12. Caso de Uso: Definir/atualizar teto de gasto de uma lista/compra

Assumindo que uma COMPRA é criada/associada a uma LISTA quando o "modo compra" se inicia ou a lista é finalizada. Se a COMPRA não existir para a LISTA, este endpoint pode criá-la.

1. **ENDPOINT:**

   PUT /api/listas/{lista_id}/compra/teto-gasto

   (Alternativa: POST /api/compras para criar e PUT /api/compras/{compra_id} para atualizar, mas vincular pela lista pode ser mais direto para este caso de uso)

2. **Descrição:** Define ou atualiza o teto de gasto para a sessão de compra associada a uma lista. Pode criar o registro COMPRA se ele não existir para esta lista.

3. **Parâmetros de URL:**

   - lista_id (integer): ID da lista.

4. **Corpo da Requisição (JSON):**

   ```
   {
    "teto_gasto": 250.00
   }
   ```

   content_copydownload Use code **with caution**.Json

5. **Resposta de Sucesso:**

   Código: 200 OK (ou 201 Created se criou o registro COMPRA)

   Corpo: (Retorna o objeto COMPRA atualizado/criado)

   ```
   {
    "compra_id": 7, // ID da compra associada
    "lista_id": 1,
    "teto_gasto": 250.00,
    "gasto_atual": 75.80, // Calculado com base nos itens comprados com preço real, se houver
    "data_compra": "2025-06-08T00:00:00Z" // Data de início/atualização da compra
   }
   ```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**

   - 400 Bad Request → teto_gasto inválido ou ausente.
   - 404 Not Found → Lista com lista_id não encontrada.

- 500 Internal Server Error → Erro ao definir/atualizar o teto.

---

13. Caso de Uso: Ativar/desativar modo de verificação de despensa

Este "modo" pode ser um estado da UI ou um atributo persistido na lista. Se persistido, a lista teria um campo como modo_atual.

Para fins de API, vamos supor que é um estado que pode ser salvo no servidor/DB na entidade LISTA.

1. **ENDPOINT:**

   PATCH /api/listas/{lista_id}/modo

2. **Descrição:** Define o modo operacional atual para a lista.

3. **Parâmetros de URL:**

   - lista_id (integer): ID da lista.

4. **Corpo da Requisição (JSON):**

   ```
   {
    "modo_lista": "verificacao_despensa" // Outros valores: "planejamento", "compra_em_andamento"
   }
   ```

   content_copydownload Use code **with caution**.Json

   (Nota: você precisaria adicionar um campo modo_lista na sua tabela LISTA)

5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

   ```
   {
    "id": 1,
    "nome": "Compras Essenciais da Semana",
    "essencial": true,
    "dataCriacao": "2025-06-07T10:15:30Z",
    "modo_lista": "verificacao_despensa"
   }
   ```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**

   - 400 Bad Request → modo_lista inválido ou ausente.
   - 404 Not Found → Lista não encontrada.

- 500 Internal Server Error → Erro ao atualizar o modo.

---

13. Caso de Uso: Ativar/desativar modo de compra simplificado

(Seu item 12 da lista de faltantes)

Similar ao anterior, este também seria um estado da lista.

1. **ENDPOINT:**

   PATCH /api/listas/{lista_id}/modo

2. **Descrição:** Define o modo operacional atual para a lista (especificamente o modo de compra simplificado).

3. **Parâmetros de URL:**

   - lista_id (integer): ID da lista.

4. **Corpo da Requisição (JSON):**

   ```
   {
    "modo_lista": "compra_simplificado" // ou "planejamento", "verificacao_despensa"
   }
   ```

   content_copydownload Use code **with caution**.Json

5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

   ```
   {
    "id": 1,
    "nome": "Compras Essenciais da Semana",
    "essencial": true,
    "dataCriacao": "2025-06-07T10:15:30Z",
    "modo_lista": "compra_simplificado"
   }
   ```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**

   - 400 Bad Request → modo_lista inválido ou ausente.
   - 404 Not Found → Lista não encontrada.
   - 500 Internal Server Error → Erro ao atualizar o modo.

---

15. Caso de Uso: Adicionar marca preferida ou descartada a um produto

1. **ENDPOINT:**

   POST /api/produtos/{produto_id}/preferencias-marca

2. **Descrição:** Adiciona uma preferência de marca (preferida ou descartada) para um produto específico.

3. **Parâmetros de URL:**

   - produto_id (integer): ID do produto (da tabela PRODUTO).

4. **Corpo da Requisição (JSON):**

```
{
 "marca_id": 25, // ID da marca (da tabela MARCA)
 "tipo": "preferida" // ou "descartada"
}
```

   content_copydownload Use code **with caution**.Json

5. **Resposta de Sucesso:**

   Código: 201 Created

   Corpo:

```
{
 "pref_id": 101, // Novo ID da PREFERENCIA_MARCA
 "produto_id": 77,
 "marca_id": 25,
 "marca_nome": "Marca XYZ", // Incluído para conveniência
 "tipo": "preferida"
}
```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**

   - 400 Bad Request → Campo obrigatório ausente ou tipo inválido.
   - 404 Not Found → Produto ou Marca não encontrado.
   - 409 Conflict → Preferência similar já existe para este produto e marca.
   - 500 Internal Server Error → Erro ao adicionar preferência.

---

16. Caso de Uso: Listar marcas preferidas/descartadas de um produto

1. **ENDPOINT:**

   GET /api/produtos/{produto_id}/preferencias-marca

2. **Descrição:** Lista todas as preferências de marca (preferidas e/ou descartadas) para um produto.
3. **Parâmetros de URL:**
   - produto_id (integer): ID do produto.
4. **Parâmetros de Query (opcionais):**
   - tipo (string): Filtra por tipo ("preferida" ou "descartada").
5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

```
[
 {
  "pref_id": 101,
  "produto_id": 77,
  "marca_id": 25,
  "marca_nome": "Marca XYZ",
  "tipo": "preferida"
 },
 {
  "pref_id": 102,
  "produto_id": 77,
  "marca_id": 30,
  "marca_nome": "Marca Ruim ABC",
  "tipo": "descartada"
 }
 // ...
]
```

   content_copydownload Use code **with caution**.Json
6. **Erros Possíveis:**
   - 404 Not Found → Produto não encontrado.
   - 500 Internal Server Error → Erro ao buscar preferências.

---

17. Caso de Uso: Remover marca preferida ou descartada de um produto

1. **ENDPOINT:**

   DELETE /api/produtos/{produto_id}/preferencias-marca/{pref_id}

   (Alternativa: DELETE /api/preferencias-marca/{pref_id} se pref_id for globalmente único e suficiente)

2. **Descrição:** Remove uma preferência de marca específica de um produto.
3. **Parâmetros de URL:**
    - produto_id (integer): ID do produto (para escopo, opcional se pref_id for global).
    - pref_id (integer): ID da preferência de marca a ser removida.
4. **Corpo da Requisição:**

    Nenhum.
5. **Resposta de Sucesso:**

    Código: 204 No Content

    Corpo: Nenhum.
6. **Erros Possíveis:**
    - 404 Not Found → Preferência de marca (pref_id) ou produto não encontrado.
    - 500 Internal Server Error → Erro ao remover preferência.

---

18. Caso de Uso: Criar registro de compra a partir de lista finalizada

Este endpoint formaliza o início de uma compra baseada em uma lista.

1. **ENDPOINT:**

    POST /api/compras
2. **Descrição:** Cria um novo registro de compra, associando-o a uma lista existente e definindo o teto de gasto inicial.
3. **Parâmetros de URL:**

    Nenhum.
4. **Corpo da Requisição (JSON):**

```
{
 "lista_id": 1,
 "teto_gasto": 300.00,
 "data_compra": "2025-06-08T14:00:00Z" // Opcional, pode ser CURRENT_TIMESTAMP
}
```

content_copydownload Use code **with caution**.Json

5. **Resposta de Sucesso:**

   Código: 201 Created

   Corpo:

```
{
 "compra_id": 8, // Novo ID da COMPRA
 "lista_id": 1,
 "teto_gasto": 300.00,
 "gasto_atual": 0.00, // Inicialmente zero
 "data_compra": "2025-06-08T14:00:00Z"
}
```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 400 Bad Request → lista_id ou teto_gasto ausente/inválido.
   - 404 Not Found → Lista com lista_id não encontrada.
   - 409 Conflict → Já existe uma compra ativa para esta lista (se a regra for 1:1 ativa).
   - 500 Internal Server Error → Erro ao criar registro de compra.

---

19. Caso de Uso: Listar compras realizadas

1. **ENDPOINT:**

   GET /api/compras

2. **Descrição:** Retorna uma lista de todas as compras realizadas, possivelmente filtradas por usuário ou período.

3. **Parâmetros de URL:**

   Nenhum.

4. **Parâmetros de Query (opcionais):**
   - usuario_id (integer): Se o sistema suportar múltiplos usuários (seu modelo tem USUARIO_ID em LISTA, então indiretamente sim).
   - data_inicio (date string): Para filtrar por período.
   - data_fim (date string): Para filtrar por período.

5. **Resposta de Sucesso:**

Código: 200 OK

Corpo:

```
[
 {
  "compra_id": 8,
  "lista_id": 1,
  "lista_nome": "Compras Essenciais da Semana", // Conveniência
  "teto_gasto": 300.00,
  "gasto_atual": 285.50,
  "data_compra": "2025-06-08T14:00:00Z"
 },
 {
  "compra_id": 9,
  "lista_id": 2,
  "lista_nome": "Guloseimas Junho", // Conveniência
  "teto_gasto": 100.00,
  "gasto_atual": 95.20,
  "data_compra": "2025-06-01T10:00:00Z"
 }
 // ...
]
```

content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 500 Internal Server Error → Erro ao buscar compras.

---

20. Caso de Uso: Adicionar item de compra (produto, quantidade comprada, marca escolhida, preço real)

1. **ENDPOINT:**

   POST /api/compras/{compra_id}/itens-comprados

2. **Descrição:** Registra um item efetivamente comprado, com sua quantidade, marca escolhida e preço real.

3. **Parâmetros de URL:**
   - compra_id (integer): ID da compra à qual este item pertence.

4. **Corpo da Requisição (JSON):**

```
{
 "produto_id": 101, // ID do produto base
 "marca_id": 25,    // ID da marca efetivamente comprada (pode ser nulo se não rastreado)
 "quantidade_comprada": 2,
 "preco_real": 5.75 // Preço unitário real
}
```

content_copydownload Use code **with caution**.Json

5. **Resposta de Sucesso:**

   Código: 201 Created

   Corpo:

```
{
 "compra_item_id": 201, // Novo ID do COMPRA_ITEM
 "compra_id": 8,
 "produto_id": 101,
 "produto_nome": "Leite Integral UHT", // Conveniência
 "marca_id": 25,
 "marca_nome": "Marca XYZ", // Conveniência
 "quantidade_comprada": 2,
 "preco_real": 5.75
}
```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 400 Bad Request → Campos obrigatórios ausentes/inválidos.
   - 404 Not Found → compra_id, produto_id ou marca_id não encontrado.
   - 500 Internal Server Error → Erro ao adicionar item de compra.

---

21. Caso de Uso: Listar itens de uma compra realizada

1. **ENDPOINT:**

   GET /api/compras/{compra_id}/itens-comprados

2. **Descrição:** Lista todos os itens que foram efetivamente comprados em uma determinada sessão de compra.

3. **Parâmetros de URL:**
   - compra_id (integer): ID da compra.

4. **Corpo da Requisição:**

   Nenhum.

5. **Resposta de Sucesso:**

   Código: 200 OK

   Corpo:

```
[
 {
  "compra_item_id": 201,
  "compra_id": 8,
  "produto_id": 101,
  "produto_nome": "Leite Integral UHT",
  "marca_id": 25,
  "marca_nome": "Marca XYZ",
  "quantidade_comprada": 2,
  "preco_real": 5.75
 },
 {
  "compra_item_id": 202,
  "compra_id": 8,
  "produto_id": 105,
  "produto_nome": "Arroz Agulhinha Tipo 1",
  "marca_id": 33, // Marca escolhida
  "marca_nome": "Tio João",
  "quantidade_comprada": 1, // Pacote de 5kg
  "preco_real": 22.50
 }
 // ...
]
```

content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 404 Not Found → compra_id não encontrado.
   - 500 Internal Server Error → Erro ao buscar itens da compra.

---

22. Caso de Uso: Exportar/duplicar lista anterior

1. **ENDPOINT:**

   POST /api/listas/{lista_id}/duplicar

2. **Descrição:** Cria uma nova lista de compras copiando todos os itens de uma lista existente. A nova lista terá um novo ID e data de criação. Os status dos itens podem ser resetados para "pendente".

3. **Parâmetros de URL:**
   - lista_id (integer): ID da lista original a ser duplicada.

4. **Corpo da Requisição (JSON) (Opcional):**

```
{
 "novo_nome_lista": "Cópia - Compras Essenciais da Semana", // Se não fornecido, pode usar "Cópia de [Nome Original]"
 "resetar_status_itens": true // Default: true (itens da nova lista como "pendente")
}
```

content_copydownload Use code **with caution**.Json

5. **Resposta de Sucesso:**

   Código: 201 Created

   Corpo: (Retorna a nova lista criada, sem os itens para brevidade, ou com os itens se preferir)

```
{
 "id": 15, // ID da nova lista duplicada
 "nome": "Cópia - Compras Essenciais da Semana",
 "essencial": true, // Copiado da original
 "dataCriacao": "2025-06-09T10:00:00Z", // Nova data
 "modo_lista": "planejamento" // Resetado para o modo padrão
 // Poderia incluir uma URL para os itens da nova lista: /api/listas/15/itens
}
```

   content_copydownload Use code **with caution**.Json

6. **Erros Possíveis:**
   - 404 Not Found → Lista original com lista_id não encontrada.
   - 500 Internal Server Error → Erro ao duplicar a lista.

Regras de Negócio Detalhadas

RG-GERAL (Regras Gerais do Sistema)

- **RG-GERAL-001 (Usuários):** O sistema é projetado para uso por "Eu" e "Raquel". Embora a modelagem inclua USUARIO_ID, a implementação inicial pode focar em um único conjunto de dados compartilhado ou facilmente alternável, ou simplesmente não ter login explícito se for usado em dispositivos separados com cópias locais do banco de dados. Para o escopo atual de "local", assumimos um usuário implícito por instalação.
- **RG-GERAL-002 (Persistência):** Todos os dados de listas, produtos, compras, etc., devem ser persistidos localmente usando IndexedDB.
- **RG-GERAL-003 (Consistência de IDs):** IDs (PKs e FKs) devem ser gerenciados para garantir a integridade relacional dos dados.

RN-LISTA (Regras para Listas de Compras)

- **RN-LISTA-001 (Criação):**
  - Toda lista deve ter um nome descritivo.
  - Toda lista deve ser classificada quanto à prioridade (ex: "essencial" ou "supérflua" no momento da sua criação.

    - A data_criacao é registrada automaticamente.
- **RN-LISTA-002 (Unicidade):** Não há restrição para nomes de lista duplicados, mas cada lista terá um lista_id único.
- **RN-LISTA-003 (Exclusão):**
    - Ao excluir uma lista, todos os LISTA_ITEM (itens da lista) associados a ela também devem ser excluídos.
    - Se uma COMPRA estiver associada a esta lista, a regra de exclusão da compra (e seus COMPRA_ITEM) deve ser considerada (ex: impedir exclusão da lista se houver compra vinculada, ou excluir em cascata – a primeira opção parece mais segura para histórico). *Para simplificar, inicialmente, pode-se permitir excluir a lista e a compra associada se existir.*
- **RN-LISTA-004 (Duplicação):**
    - É possível duplicar uma lista existente.
    - A nova lista terá um novo lista_id e a data_criacao atual.
    - Todos os LISTA_ITEM da lista original são copiados para a nova lista.
    - Por padrão, o status de todos os itens na lista duplicada deve ser "pendente".
    - O nome da nova lista pode ser sugerido como "Cópia de [Nome da Lista Original]" ou personalizável.

RN-ITEM (Regras para Itens da Lista - LISTA_ITEM)

- **RN-ITEM-001 (Adição):**
    - Um item de lista (LISTA_ITEM) sempre pertence a uma LISTA e referencia um PRODUTO base.
    - É obrigatório informar a quantidade desejada para o item.
    - preco_estimado, observacao e prioridade_item (se diferente da prioridade da lista) são opcionais.
    - Ao ser adicionado, o status padrão do item é "pendente".
- **RN-ITEM-002 (Edição):**
    - Podem ser
      alterados: quantidade, preco_estimado, observacao, prioridade_item.

    - Para alterar o produto em si (ex: trocar "Arroz" por "Feijão"), o item atual deve ser removido e um novo adicionado.
- **RN-ITEM-003 (Remoção):** Um item pode ser removido de uma lista.
- **RN-ITEM-004 (Status Compra):**
    - Um item pode ser marcado como "comprado" ou "pendente".
    - A marcação de um item como "comprado" durante o "modo compra" pode opcionalmente solicitar o preco_real e a marca_id escolhida, para já popular os dados de COMPRA_ITEM (se uma COMPRA estiver ativa).
- **RN-ITEM-005 (Produto Base):**
    - Cada LISTA_ITEM refere-se a um PRODUTO.
    - O PRODUTO contém nome, quantidade_minima (sugestão de uso) e categoria_id.
    - Se um produto ainda não existe na base PRODUTO ao tentar adicioná-lo a uma lista, o sistema deve permitir sua criação (com nome e categoria) ou exigir que seja cadastrado previamente.

RN-PRODUTO (Regras para Produtos Base)

- **RN-PRODUTO-001 (Criação):** Um produto deve ter um nome e pertencer a uma CATEGORIA. quantidade_minima é opcional.
- **RN-PRODUTO-002 (Categorias):** Categorias (CATEGORIA) devem ser pré-definidas ou permitir cadastro pelo usuário (ex: "Limpeza", "Higiene").

RN-MARCA (Regras para Marcas e Preferências)

- **RN-MARCA-001 (Cadastro):** Marcas (MARCA) devem ter um nome.
- **RN-MARCA-002 (Preferência):**
    - Uma PREFERENCIA_MARCA associa um PRODUTO a uma MARCA com um tipo ("preferida" ou "descartada").
    - Um produto pode ter múltiplas marcas preferidas e múltiplas descartadas.
    - Uma mesma marca não pode ser simultaneamente "preferida" e "descartada" para o mesmo produto.
- **RN-MARCA-003 (Uso):** As preferências de marca são informativas e auxiliam na decisão durante a compra. Não impedem a compra de outras

marcas.

RN-COMPRA (Regras para Registro de Compra)

- **RN-COMPRA-001 (Criação/Início):**
  - Uma COMPRA é criada e associada a uma LISTA (relação 1:1 na sua modelagem, significando que uma lista gera uma compra).
  - Ao criar/iniciar uma COMPRA, um teto_gasto deve ser definido.
  - A data_compra é registrada.
  - Inicialmente, o gasto_atual é R$ 0,00.
- **RN-COMPRA-002 (Teto de Gasto):**
  - O teto_gasto pode ser definido ou atualizado a qualquer momento antes da finalização da compra.
- **RN-COMPRA-003 (Cálculo Gasto Atual):**
  - O gasto_atual da COMPRA é a soma de (quantidade_comprada * preco_real) de todos os COMPRA_ITEM associados a essa COMPRA.
  - Este valor deve ser recalculado sempre que um COMPRA_ITEM for adicionado/atualizado/removido ou quando um LISTA_ITEM for marcado como "comprado" e seu preco_real for informado.
- **RN-COMPRA-004 (Aviso de Teto Excedido):**
  - **Gatilho:** A verificação ocorre sempre que o gasto_atual é recalculado (ao adicionar/marcar item como comprado e informar preço real, ao editar preço real de item comprado).
  - **Condição:** Se gasto_atual > teto_gasto.
  - **Ação:**
    1. Exibir um aviso visual claro e não intrusivo na tela (ex: uma notificação flutuante, um banner no topo da seção de totais) informando "Atenção: Teto de gastos ultrapassado!".
    2. O valor do gasto_atual exibido na tela deve ser apresentado em cor vermelha.
    3. O valor que representa "o quanto ainda pode ser gasto" (calculado como teto_gasto - gasto_atual) deve ser exibido em cor vermelha se

for negativo.

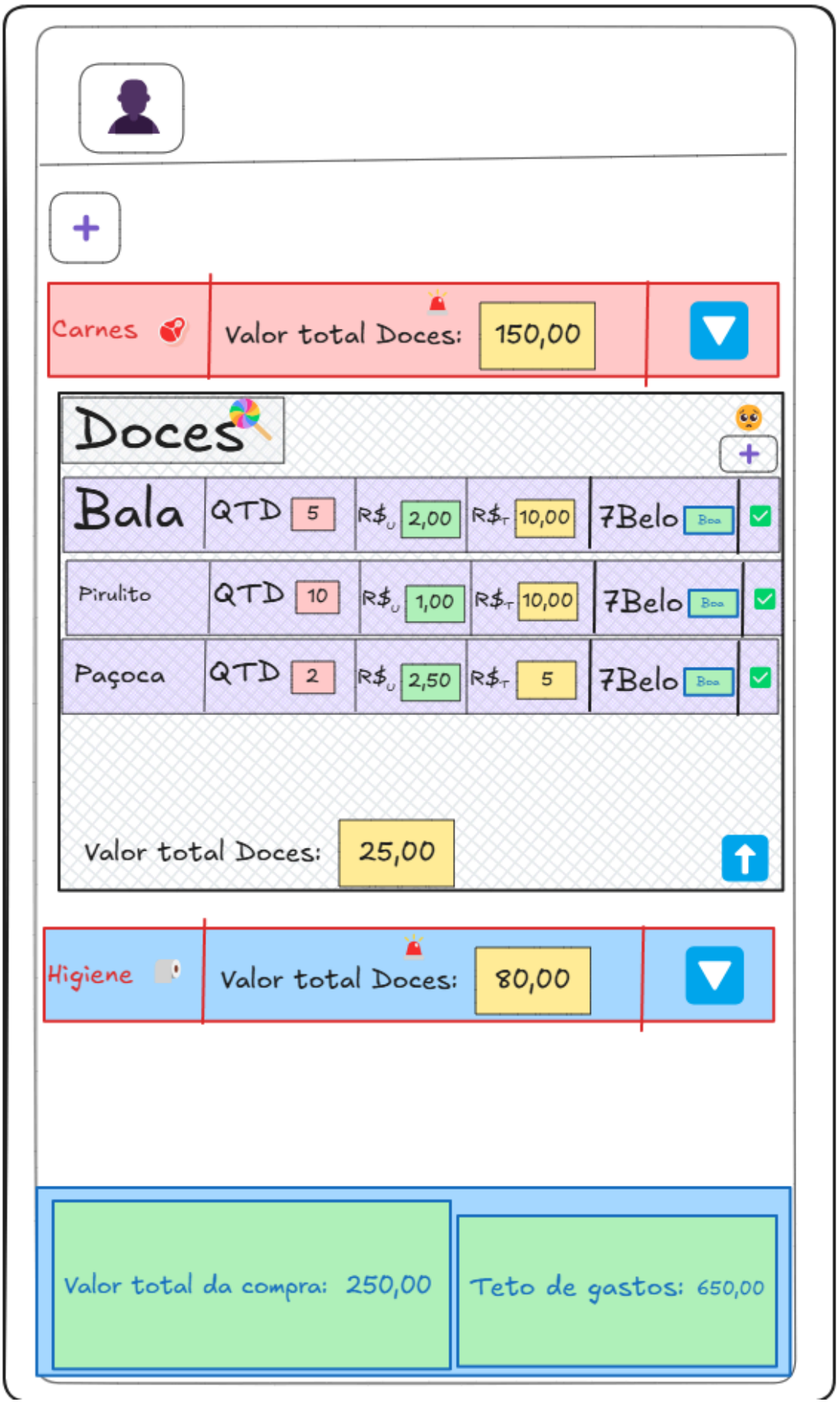
    - **Comportamento:** O aviso não impede o usuário de continuar adicionando itens ou ultrapassando ainda mais o teto. É apenas informativo.
- **RN-COMPRA-005 (Itens da Compra - COMPRA_ITEM):**
    - Um COMPRA_ITEM registra um PRODUTO efetivamente comprado.
    - Deve incluir quantidade_comprada, preco_real (unitário) e, opcionalmente, a marca_id da marca escolhida.
- **RN-COMPRA-006 (Finalização - Implícito):** Embora não haja um status "finalizada" na entidade COMPRA no seu modelo, pode-se considerar uma compra como "finalizada" quando o usuário sai do modo de compra ou a salva explicitamente. Isso poderia "travar" a compra para futuras edições e solidificar os dados para relatórios.
- **RN-COMPRA-007 (Histórico):** Compras (e seus itens) devem ser mantidas para consulta futura e para alimentar "Extras inteligentes" (sugestões, relatórios).

RN-CALCULO (Regras para Cálculos e Exibições)

- **RN-CALCULO-001 (Subtotal Comprados na Lista):**
    - Soma dos (quantidade * preco_estimado) para todos os LISTA_ITEM com status = "comprado".
    - *Alternativa durante a compra*: Se um preco_real já foi informado para um item comprado, usar este no lugar do preco_estimado para o subtotal. O gasto_atual da COMPRA é mais preciso para isso.
- **RN-CALCULO-002 (Total Estimado da Compra - Itens Pendentes):**
    - Soma dos (quantidade * preco_estimado) para todos os LISTA_ITEM com status = "pendente" e que tenham preco_estimado preenchido.
    - Itens sem preco_estimado não entram no cálculo para não distorcer a estimativa.
- **RN-CALCULO-003 (Saldo Restante):**
    - Exibido como: teto_gasto (da COMPRA) - gasto_atual (da COMPRA).

- Se este valor for negativo, deve ser destacado conforme **RN-COMPRA-004**.

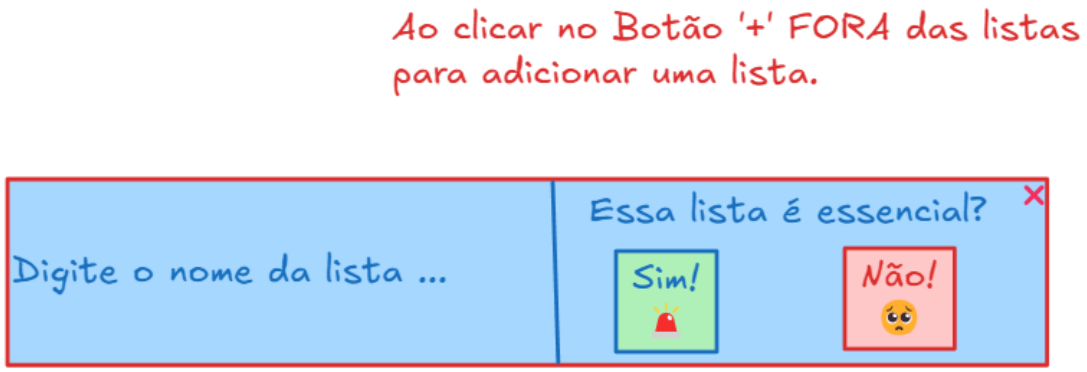
Wireframes para o Front End.


Carnes
Valor total Doces:
150,00
Doces
Bala
QTD
5
R$U
2,00
R$T
10,00
7Belo
Boa
Pirulito
QTD
10
R$U
1,00
R$T
10,00
7Belo
Boa
Paçoca
QTD
2
R$U
2,50
R$T
5
7Belo
Boa
Valor total Doces:
25,00
Higiene
Valor total Doces:
80,00
Valor total da compra: 250,00
Teto de gastos: 650,00

O Aplicativo consiste em 3 partes.

Header: contém apenas um seletor de perfil.

Body: Onde se inicia em formato de lista / tabela.
O primeiro elemento e um ícone de ‘+’ para adicionar as listas que surgem a baixo. Ao clicar em ‘+’ aparece um container em tela sobrepondo as listas:



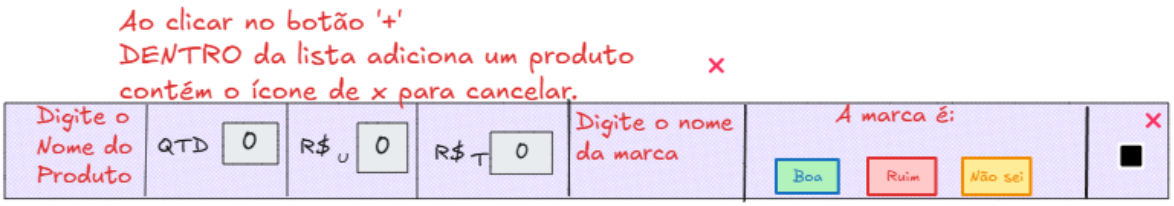
O container contem apenas o imput para digitar o nome da lista, e ao lado a pergunta se a lista é essencial ou não, se sim, a lista ficará com um ícone de uma sirene (ou outro mais relevante) se não a lista fica com um ícone de um emoji implorando (ou outro mais relevante).

As listas ficam por ordem de criação / essencial

A lista é uma s**eção colapsável** quando está ‘fechada’ contém apenas o nome, um ícone ou imagem representativo, o valor total da lista e o ícone para abrir/colapsar.

Ao abrir/colapsar a lista o container ocupa 50% do tamanho disponível, caso abra 2 listas, esse tamanho é reduzido para 25% cada. Se abrir uma terceira lista, a primeira lista aberta fecha / descolapsa automaticamente. O nome da lista diminui o tamanho e fica no topo esquerdo do contianer, no extremo esquerdo fica o ícone se a lista é essencial ou não.

Logo a baixo na esquerda um botão de ‘+’ para adicionar um produto, ao clicar aparece o seguinte container:



Apenas o nome é obrigatório para criar o produto, se a marca não for inserida se é ‘boa’ ou ‘ruim’ ou ‘não sei’, por padrão será ‘não sei’.

Ao começar a escrever o nome, irá aparecer sugestões de produtos salvos.

Os produtos aparecem e são organizados por ordem de checkin/edição, quando um produto é editado ele passa ser o primeiro, a baixo dos produtos que foram dado checkin, ao dar checkin o valor do produto é somado a compra, não é possível dar checkin em um produto sem preço.

Os produtos contém, nome, quantidade, preço unitário, preço total (UxQ) nome da marca, e se a marca é uma das que gostamos ou não.

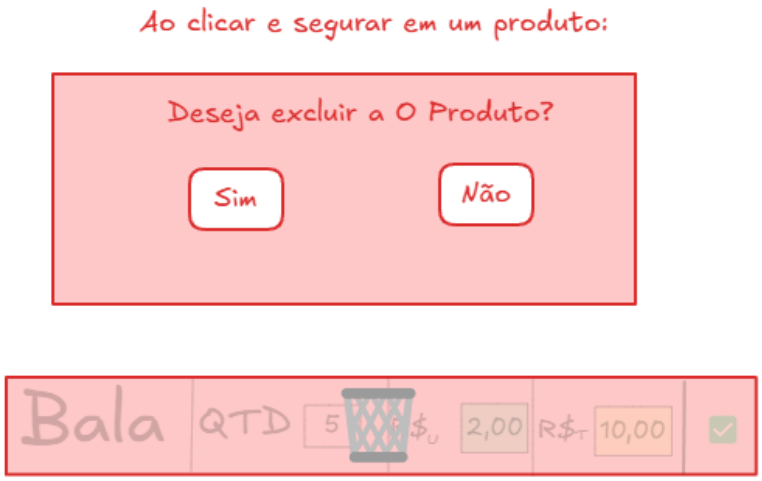
Section do rodapé:

Contém dois containers, um com o valor total gasto até agora e um com um valor do teto de gasto podendo ser editado a qualquer momento.

---

para excluir produtos ou listas é a mesma mecânica, clicar e segurar até aparecer as mensagens:

A lista ou produto fica com opacidade baixa, e o container de aviso aparece na tela.



ps: os produtos não são excluídos no banco de dados.

futuramente irei configurar uma maneira de excluir estes produtos permanentemnete.