

# Clone do iFood: Gerenciador de pedidos

# Cenário

Você está criando uma startup para competir com o iFood como um marketplace B2C de entrega de comida.

O seu diferencial será o modelo de comissão, onde o restaurante escolhe o percentual que deseja pagar, podendo ser zero.

Restaurantes que pagarem mais terão destaque nas pesquisas.

# Descrição geral deste entregável

Nesta etapa o sistema deverá permitir que o restaurante processe os pedidos recebidos diretamente na cozinha.

O sistema deverá oferecer uma tela simples onde listará os pedidos recebidos e será possível atualizá-los facilmente.



# Descrição geral deste entregável

A aplicação será escrita em Python com [Flask](#) e executará no navegador.

Para simplificar, vamos usar o banco de dados SQLite para persistir os dados.

# Descrição geral deste entregável

Seu projeto deverá dar continuidade à etapa anterior, conectando ao mesmo banco de dados.

Sempre que possível reutilize o código existente. Mova métodos de lugar, crie classes 'super' ou faça outros ajustes se necessário.

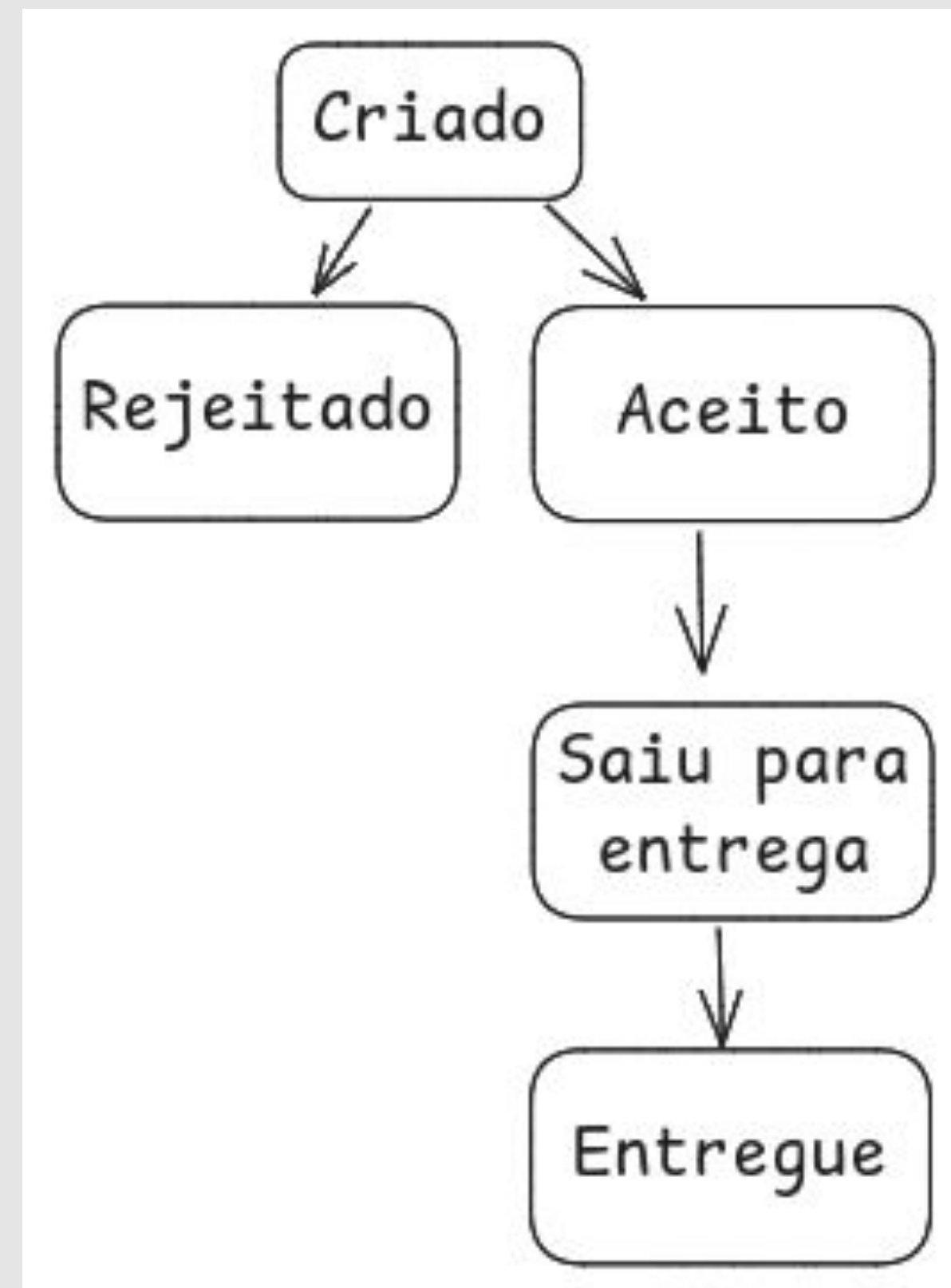
Os projetos anteriores devem continuar funcionando.

# Fluxo de um pedido

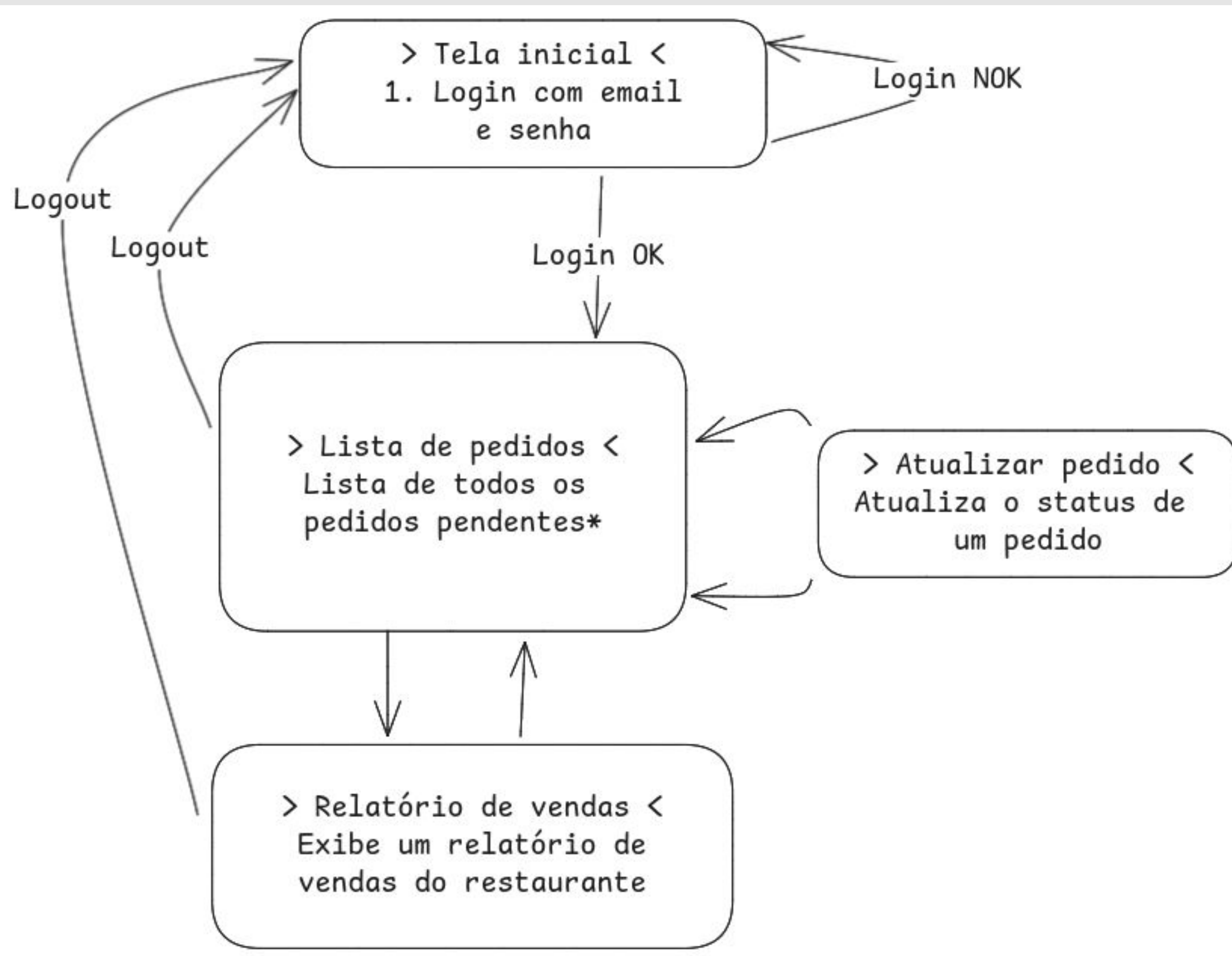
Adicione uma nova coluna na tabela dos pedidos para armazenar o status.

Todos os pedidos iniciam como 'criado' e podem ser 'rejeitado's ou 'aceito's. Um pedido rejeitado não recebe outras atualizações.

Pedidos aceitos serão considerados como “em preparo” e serão atualizados para “saiu para entrega” e depois para “Entregue”.



# Fluxo



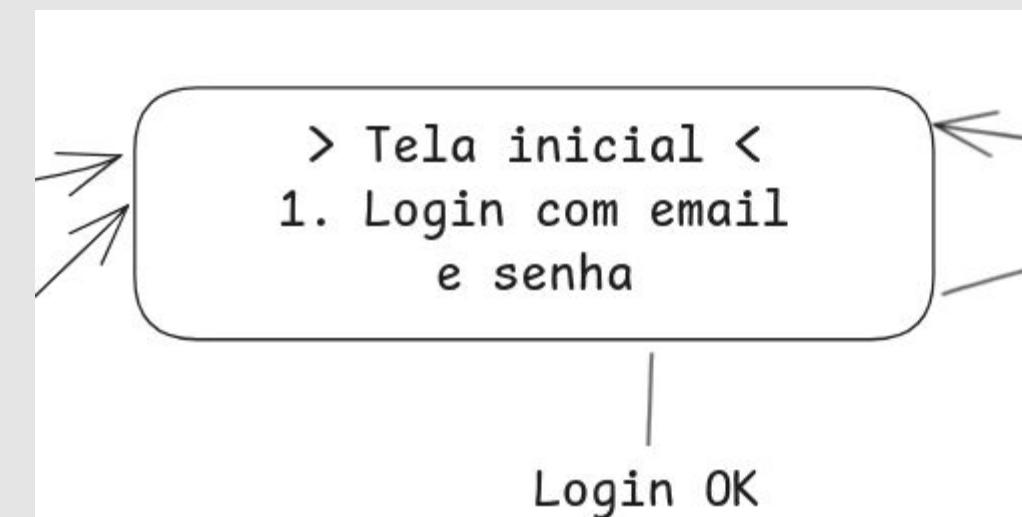
# Fluxo: Tela inicial/login

Oferece um formulário de login (email e senha) para os usuários donos de restaurantes.

Se o login falhar, mostre uma mensagem e recarregue a mesma tela.

Se o usuário tentar acessar qualquer outra tela sem estar logado, redirecione ele para esta tela.

**Sempre** considere que o usuário pode digitar um valor inválido.



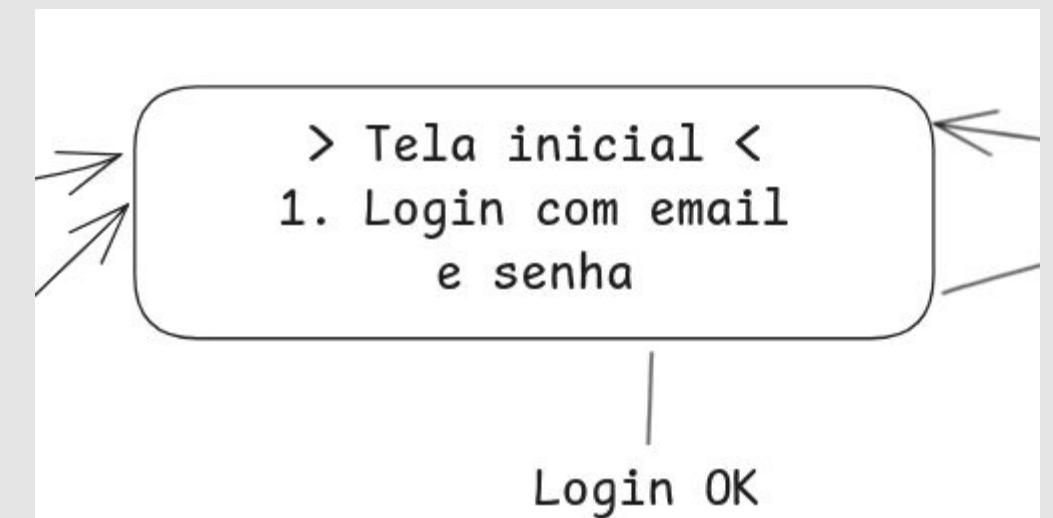


# Fluxo: Tela inicial/login

Converta o email para lowercase antes de confirmar se os dados estão corretos.

Campo senha é case-sensitive.

Armazene a data/hora do último login no registro do usuário.



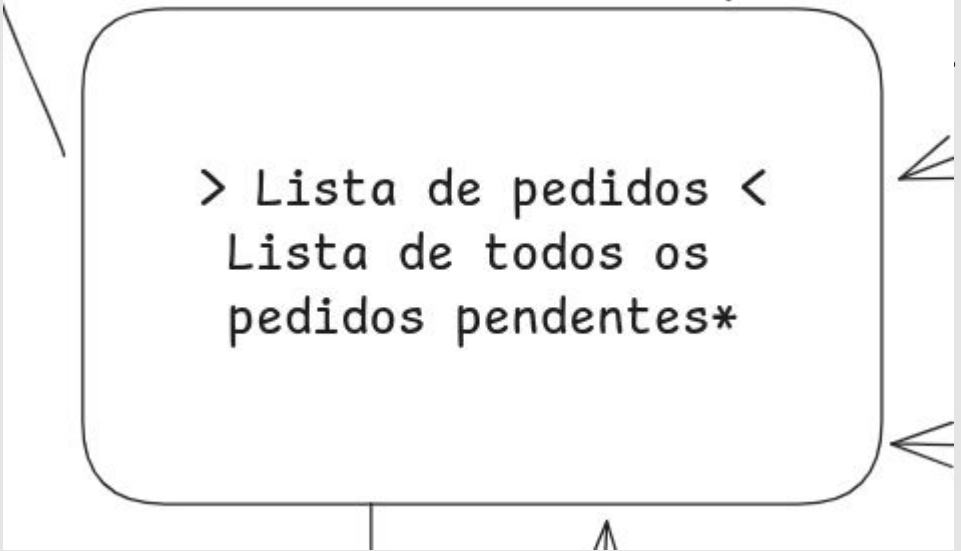
# Fluxo: Catálogo de restaurantes

Liste todos os pedidos que não estão 'rejeitados' ou 'entregues' em uma tabela, conforme a ordem de sua criação.

A tabela terá as colunas: ID do pedido, nome do cliente, total da compra e ações.

Em ações coloquei botões para mudar o status de cada pedido:

- Em 'criado's coloque as opções 'aceitar' e 'rejeitar'
- Em 'aceito's coloque a opção 'saiu para entrega'
- Em 'saiu para entrega' coloque a opção 'entregue'

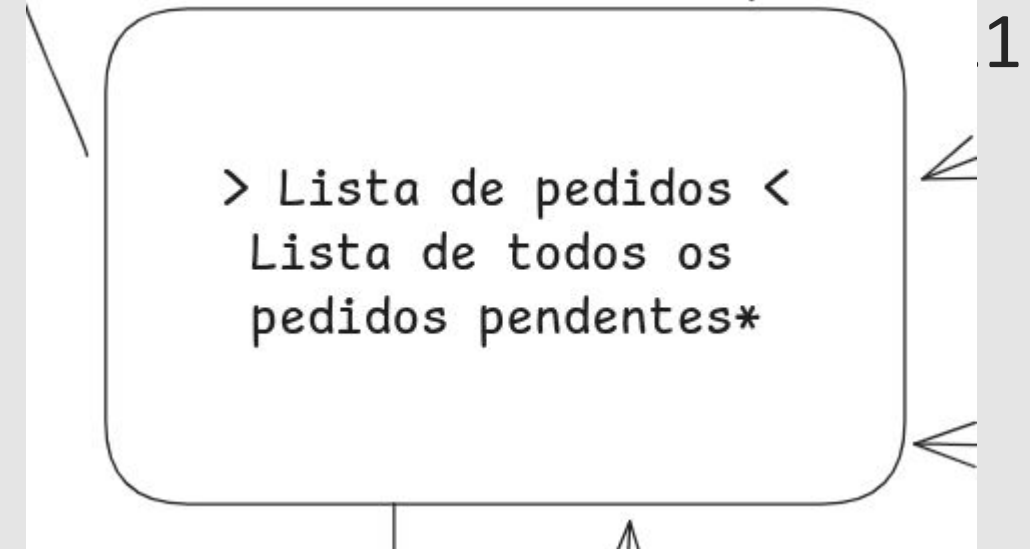


> Lista de pedidos <  
Lista de todos os  
pedidos pendentes\*

# Fluxo: Catálogo de restaurantes

Ao clicar em qualquer ação, altere o valor do pedido no banco de dados e atualize a tela com os novos valores.

Acrescente nesta tela um botão 'logout' (leva o usuário novamente o login) e um botão 'relatório' (leva o usuário para outra tela).

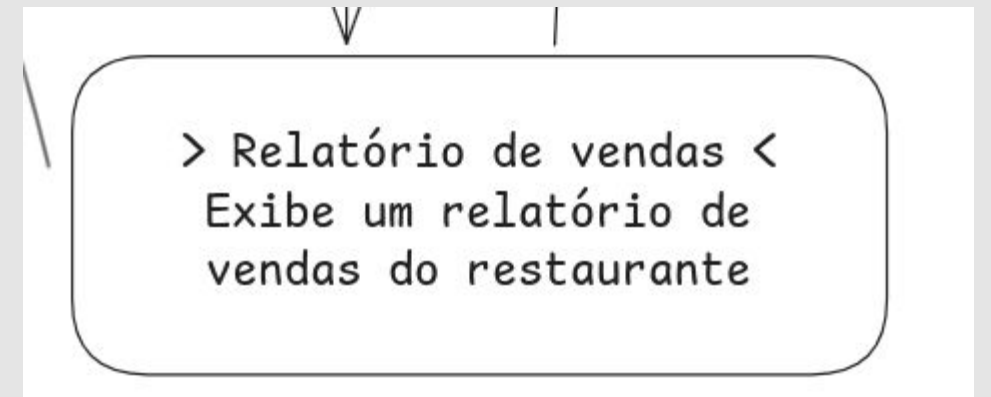


# Fluxo: Relatório de Vendas

Exiba nesta tela as informações gerais de venda do estabelecimento.

- Qual a média de gasto de cada pessoa?
- Qual a maior compra (em valor) feita no restaurante?
- Qual o maior pedido (em quantidade de itens) feita no restaurante?
- Liste a maior e a menor comissão paga pelo restaurante
- Qual o item mais pedido?
- Quantos pedidos em cada status? Liste todos os status, mesmo que não haja pedido
- Calcule a quantidade média de pedidos por cada dia da semana. Pivote o resultado.

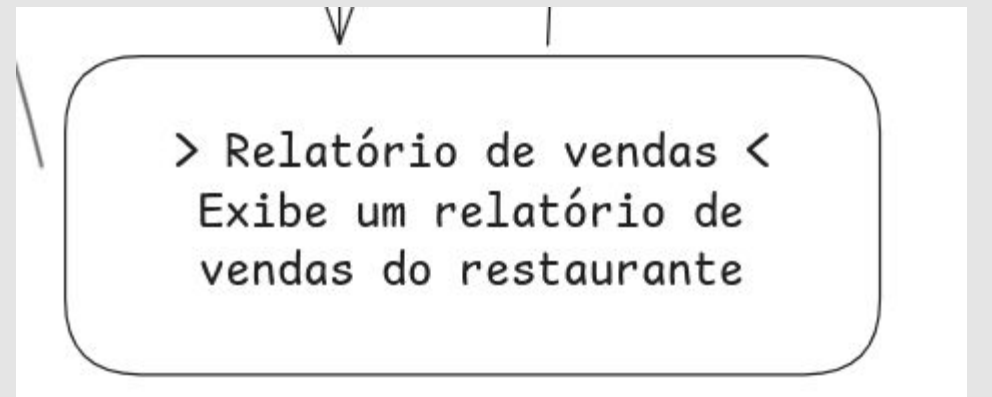
Acrescente nesta tela um botão 'logout' (leva o usuário novamente o login) e um botão 'pedidos' (leva o usuário para tela anterior).



# Fluxo: Relatório Administrativo

Exiba nesta tela as informações gerais do sistema:

- Quantidade de restaurantes e clientes cadastrados
- Quantidade de clientes únicos que já fizeram um pedido em cada restaurante
- Ticket médio por restaurante (valor médio de cada pedido)
- Pivote a quantidade de pedidos de cada restaurante (linhas) e meses (colunas)
- Crie um insight para ajudar a administração do sistema



Esta tela será acessada somente pelo Administrador do sistema. Ele será um usuário com permissão para ver os dados de todos os restaurantes. Crie o novo usuário marcando como admin (crie uma nova coluna no banco) e exiba o conteúdo desta página somente para este usuário. Demais usuários deverão receber “Acesso negado” ao acessar a página. Acrescente nesta tela um botão ‘logout’. O admin não poderá fazer pedidos: não é necessário botão para voltar pra pedidos, mas é necessária a validação para que ele não acesse a tela de pedidos.

# Fluxo: Logout

O usuário irá retornar para a tela inicial, para que possa fazer login novamente.

GARANTIR que os dados do usuário anterior **NÃO SEJAM** acessíveis para um futuro usuário que fizer login.

# Entrega POO

Você deve publicar um zip contendo o código-fonte do projeto no AVA dentro do prazo definido. O projeto deve ser compatível com Python 3.12.

Não inclua no zip o banco de dados (apague o arquivo sqlite).

O projeto deve ser 100% autoral, sem uso de bibliotecas externas.

Evite problemas: plágio (de colegas ou IA's) pode invalidar a sua entrega.

# Entrega Banco de Dados

Entrega no AVA, individual, um *.zip* com os seguintes arquivos:

- Modelo Conceitual e Lógico do Banco de Dados
  - Uma imagem *.png* para cada modelo
- Modelo Físico do Banco de Dados
  - Scripts para criação (garanta 3FN) e alteração (se necessário) de tabelas
  - Scripts para inserção de dados
- Consultas utilizadas para geração dos Relatórios
  - Enunciado/Pergunta/Insight seguido do SELECT que responde
- Banco de Dados (arquivo *.sqlite*) com dados já inseridos
- Pontos extras:
  - Os relatórios (onde fizer sentido) podem ser visualizações de dados
  - Política de Segurança e Ficha Técnica de Dados (próximo slide)

Não está permitido o uso de AI em nenhuma parte do trabalho. Também **não** copie do coleguinha.



# Opcional: Governança dos Dados

- Crie a política de segurança para o banco de dados do seu sistema.
- Crie uma ficha técnica de metadados para cada banco. A ficha deve conter:
  - Nome do Banco de Dados
  - Descrição resumida do conteúdo e finalidade
  - Proprietário/Responsável (setor ou indivíduo, com informações de contato)
  - Data de criação e frequência de atualização
  - Fonte dos dados (sistemas internos, dados de terceiros, usuários, etc)
  - Estrutura do Banco de Dados:
    - Lista de tabelas e conteúdo armazenado (inclui colunas e tipos de dados)
    - Obrigatoriedade dos dados, relacionamentos (chaves) e índices
    - Observações sobre dados sensíveis ou dados que exigem tratamento especial
  - Regras de retenção/exclusão de dados
  - Data e responsável pelo documento (com histórico de atualizações)

# Avaliação - POO

- 05% código fonte está comentado?
- 10% as classes foram organizadas em múltiplos arquivos?
- 05% código fonte está indentado?
- 05% funções foram criadas para NÃO duplicar código?
- 05% banco de dados NÃO foi incluído?
- 20% aplicação NÃO quebrou durante os testes?
- 50% as operações funcionam como esperado?
  - Login (05%), Logout (05%)
  - Listagem de pedidos (10%), Alterações de status (15%),
  - Informações do relatório (15%)

# Avaliação - Banco

- (20%) Modelo Conceitual e Lógico do Banco de Dados
  - Validação de cada modelagem (10%)
- (30%) Modelo Físico do Banco de Dados
  - (05%) Tradução do modelo lógico
  - (15%) Relacionamentos e Chaves
  - (05%) Formas normais
  - (05%) Qualidade dos dados inseridos
- (45%) Consultas utilizadas para geração dos Relatórios
  - (20%) Validação e Organização dos Enunciado/Select para Relatório de Vendas
  - (20%) Validação e Organização dos Enunciado/Select para Relatório Administrativo
  - (05%) Criatividade e Legitimidade do Insight
- (05%) Banco de Dados (arquivo *.sqlite*) com dados já inseridos