Trabalho 1

Disciplina: DM-112

Alunos: Marcio Tulio Aiex Taier Filho, Samuel Kenzo Umezawa.

Fig 1.0: Diagrama do serviço de entrega.

Requisitos

- Consultar a lista de pedidos a serem entregues
- Registrar a entrega de um pedido
- Enviar um e-mail para o cliente quando o pedido for entregue

Fronteiras de análise

- O entregador consulta a lista de pedidos a serem entregues
- O entregador registra a entrega de um pedido
- O sistema acessa o servidor de e-mails

Partes envolvidas: O entregador, a transportadora, loja e o cliente.

Partes afetadas da corporação: Logística.

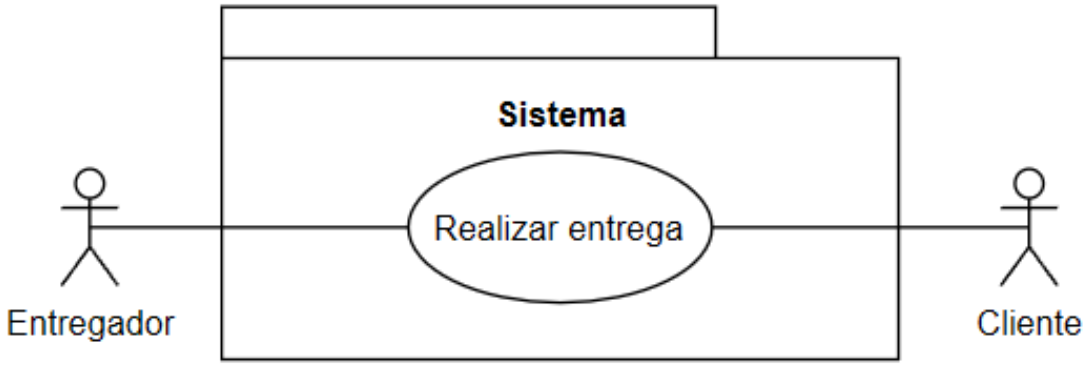


Figura 1.1: Caso de uso de entrega de pedido.

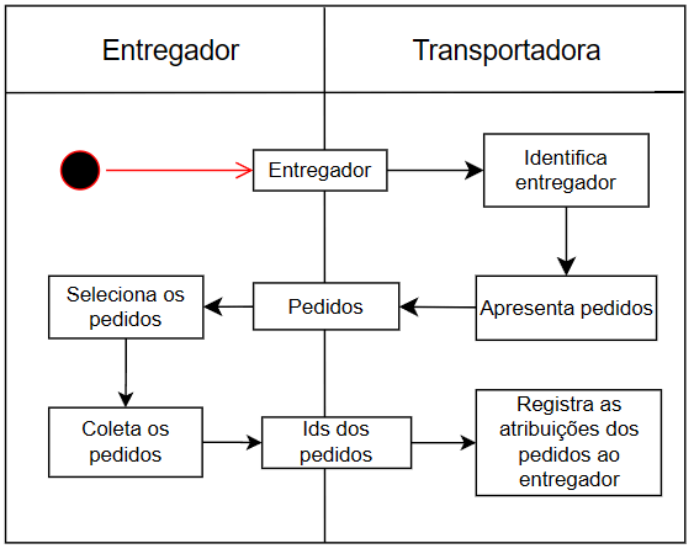


Figura 2: Modelo de processo de negócio do pagamento.

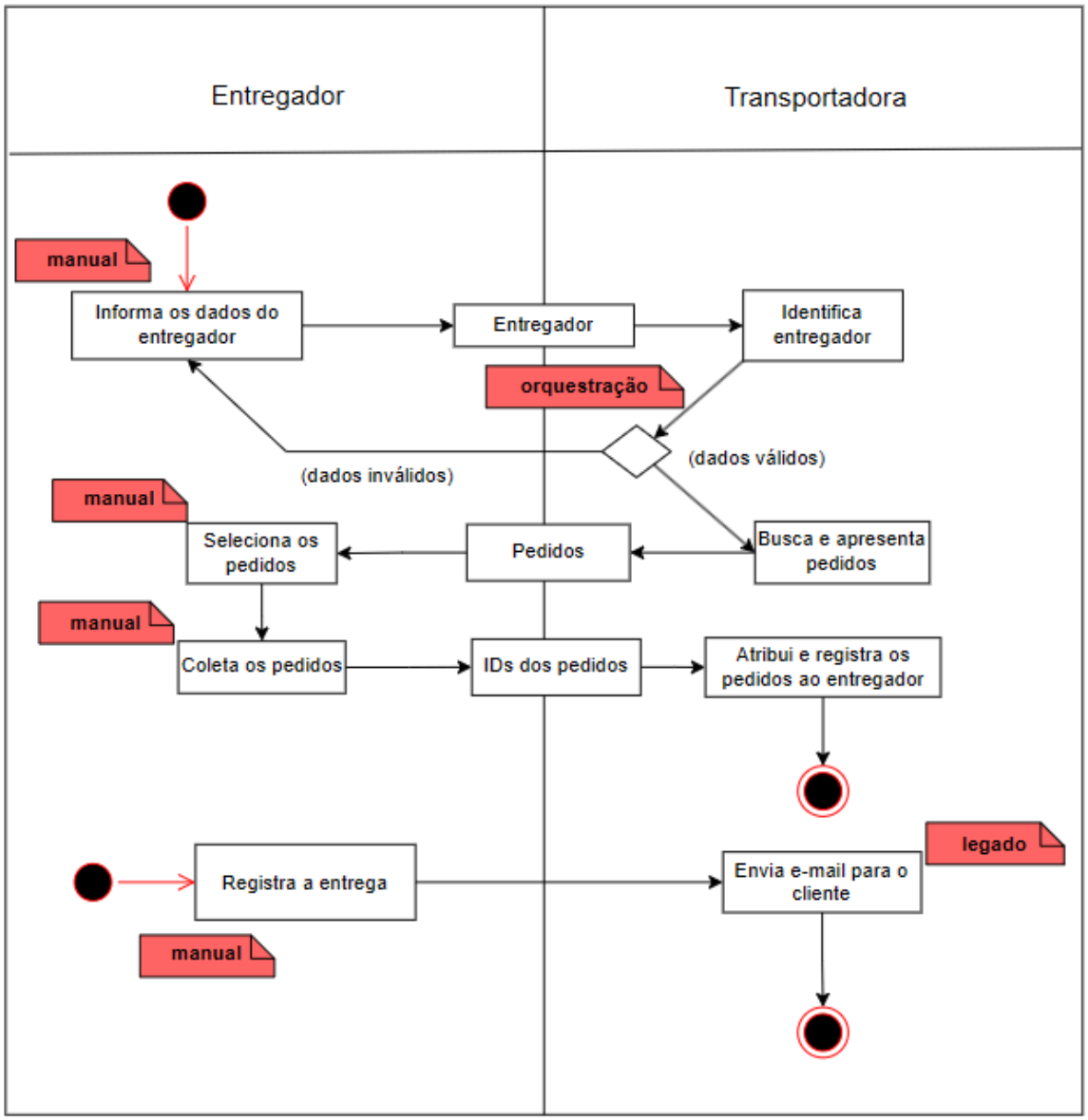


Fig 3: Marcação dos serviços - Registro de entrega do pedido.

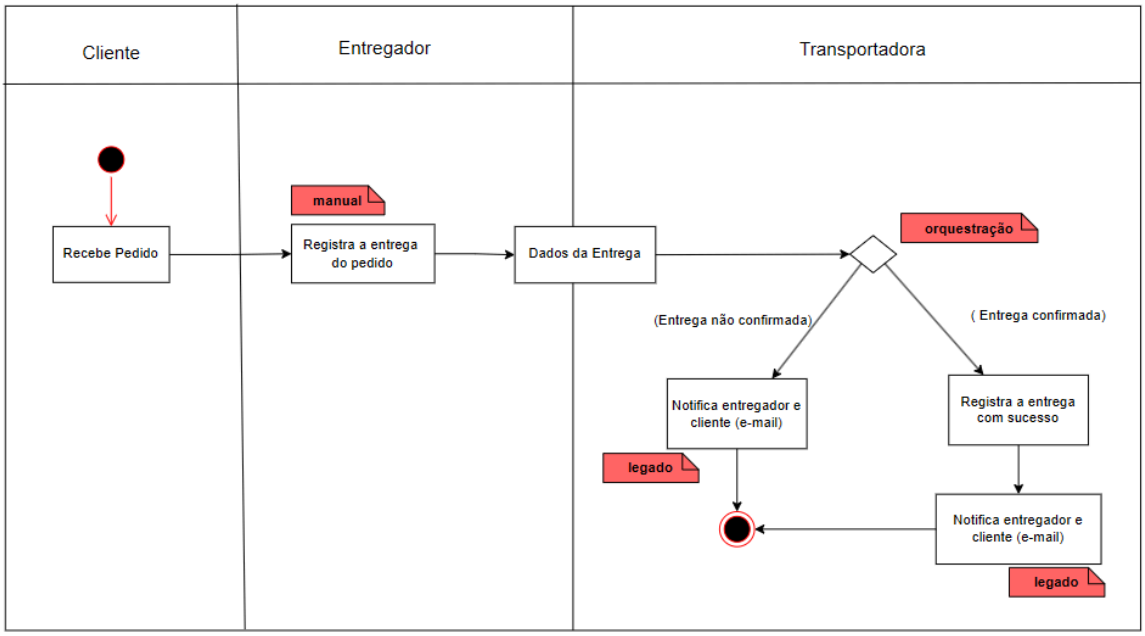


Fig 4: Marcação dos serviços - Acesso ao servidor de e-mails.

Serviços Candidatos e Aplicação dos Princípios de Orientação a Serviços

1. Pedido

- Responsável por fornecer informações sobre os pedidos que precisam ser entregues.
- **Serviços:**
  - `fetchOrders(deliveryPersonID)`: Retorna a lista de pedidos atribuídos ao entregador.
- **Princípios aplicados:**
  - **Reusabilidade:** Pode ser utilizado por diferentes clientes (aplicação web, mobile, API externa da transportadora).
  - **Autonomia:** Independente dos outros serviços; pode funcionar isoladamente.
  - **Statelessness:** A resposta depende apenas da requisição feita, sem armazenar estado da consulta no servidor.

2. Entrega

- Gerencia o registro das entregas feitas pelos entregadores.
- **Serviços:**
  - `registerDelivery(orderID, recipientCPF, deliveryDateTime)`: Registra a entrega de um pedido, armazenando as informações do recebedor e o horário da entrega.
- **Princípios aplicados:**
  - **Reusabilidade:** Pode ser utilizado por qualquer transportadora ou empresa que precise registrar entregas.
  - **Autonomia:** Não depende diretamente de outros serviços; apenas recebe dados e os armazena.
  - **Statelessness:** Cada requisição é independente; não mantém estado entre chamadas.

3. Mensageria

- Gerencia o envio de notificações automáticas relacionadas às entregas.
- **Serviços:**
  - `sendDeliveryConfirmationEmail(orderID, customerEmail)`: Envia um e-mail de confirmação de entrega para o cliente.
- **Princípios aplicados:**
  - **Reusabilidade:** Pode ser usado por outros serviços da empresa, como faturamento ou atendimento ao cliente.
  - **Autonomia:** Pode operar independentemente dos outros serviços.
  - **Baixo Acoplamento:** Os serviços que chamam este serviço não precisam esperar a confirmação imediata do envio do e-mail.
  - **Statelessness:** Cada requisição de envio de e-mail é tratada separadamente.

4. Autenticação

- Permite que os entregadores e administradores acessem os serviços de forma segura.
- **Serviços:**
  - `authenticateDeliveryPerson(username, password)`: Valida as credenciais e gera um token de autenticação.
- **Princípios aplicados:**
  - **Autonomia:** Pode ser integrado com qualquer outro serviço que precise de autenticação.
  - **Loose Coupling:** Outros serviços não precisam conhecer os detalhes da autenticação, apenas validar o token.
  - **Statelessness:** Após a autenticação, um token é gerado, e o servidor não mantém o estado do usuário.

5. Auditoria e Monitoramento

- Registra todas as operações do sistema para fins de auditoria e análise.
- **Serviços:**
  - `logEvent(operation, user, timestamp)`: Registra um evento importante no sistema, como o registro de uma entrega ou envio de um e-mail.
- **Princípios aplicados:**
  - **Reusabilidade:** Pode ser utilizado por qualquer módulo do sistema que precise de rastreamento.
  - **Autonomia:** Funciona de maneira independente, sem afetar a operação dos outros serviços.
  - **Statelessness:** Cada requisição de log é armazenada sem dependência de chamadas anteriores.

Serviço de Orquestração: Gestão de Entregas

- **Função:** Coordena as operações de consulta, registro de entrega e envio de notificações

- **Operações principais:**
  - `retrieveOrders(deliveryPersonID) → fetchOrders(deliveryPersonID)`
  - `confirmDelivery(orderID, recipientCPF, deliveryDateTime)`
    - Chama `registerDelivery(orderID, recipientCPF, deliveryDateTime)`
    - Depois, chama `sendDeliveryConfirmationEmail(orderID, customerEmail)`

Agrupamento de Serviços

1. Pedido

- `fetchOrders(deliveryPersonID)`
- **Camada:** Serviço de Entidade

2. Gestão de Entregas (Orquestração)

- `confirmDelivery(orderID, recipientCPF, deliveryDateTime)`
- **Camada:** Serviço de Orquestração

3. Entrega

- `registerDelivery(orderID, recipientCPF, deliveryDateTime)`
- **Camada:** Serviço de Tarefa

4. Mensageria

- `sendDeliveryConfirmationEmail(orderID, customerEmail)`
- **Camada:** Serviço de Utilidade

5. Autenticação

- `authenticateDeliveryPerson(username, password)`
- **Camada:** Serviço de Utilidade

6. Auditoria

- `logEvent(operation, user, timestamp)`

- **Camada:** Serviço de Utilidade

Identificação da composição de serviços

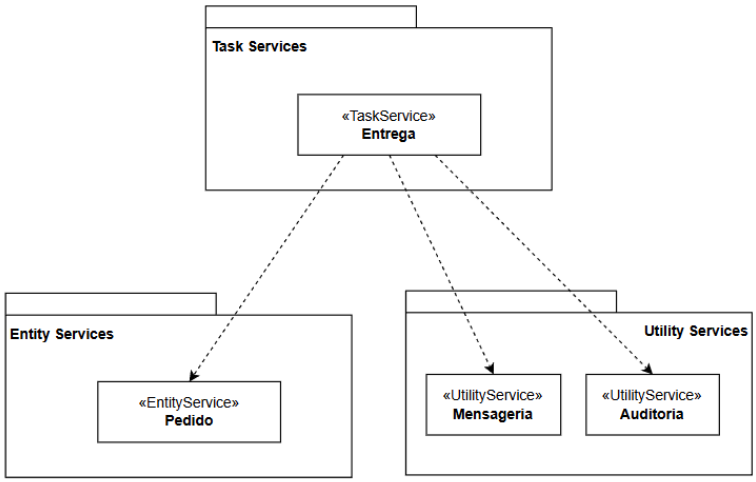
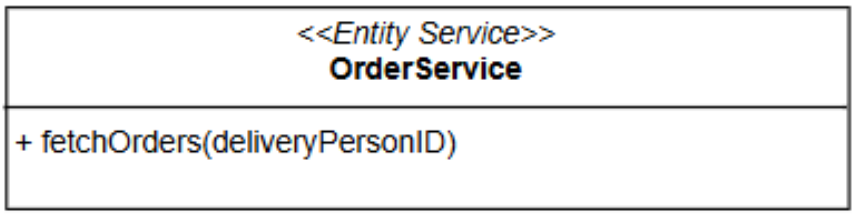
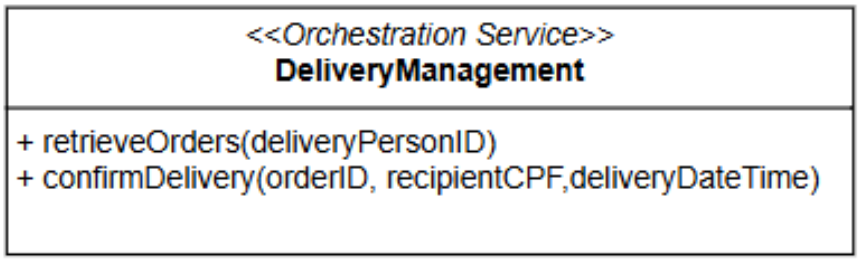


Fig 5: Relacionamento entre os serviços





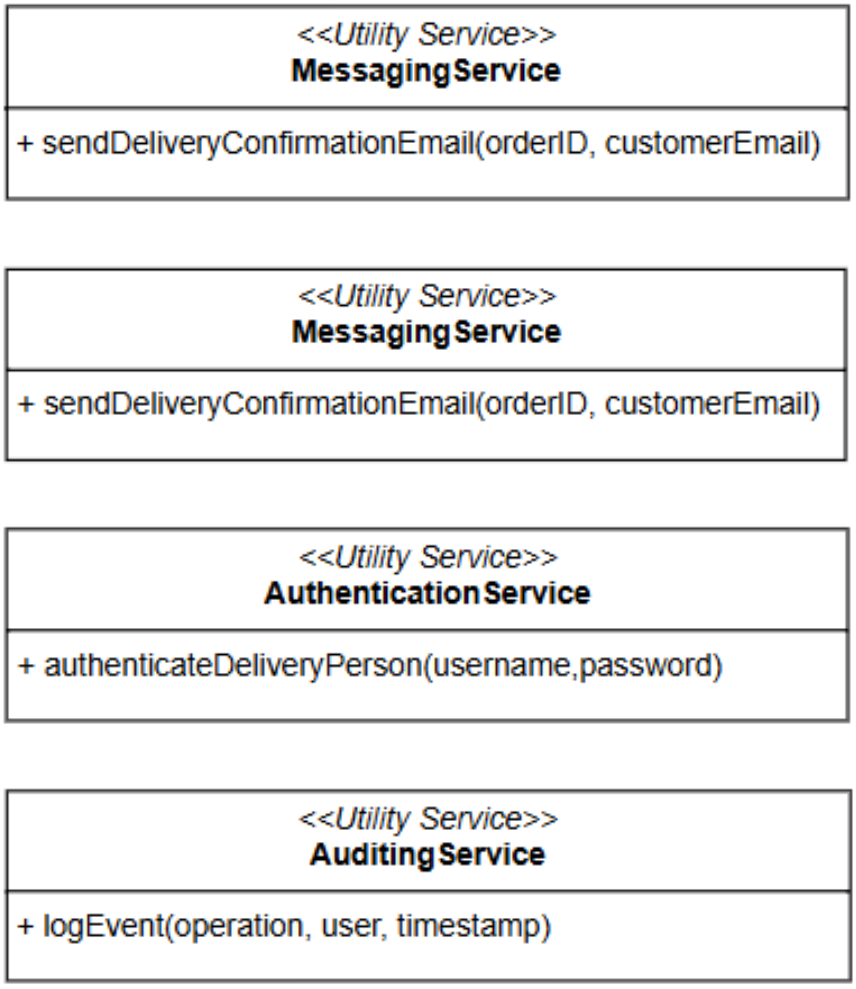


Fig 6: Diagrama de classes com os serviços detalhados.