Diagramas de Sequência

Este documento apresenta os diagramas de sequência para os principais fluxos da aplicação Simple. Os diagramas ilustram as interações entre os diferentes componentes do sistema e ajudam a entender como os processos são executados.

1. Registro de Novo Pedido

Este diagrama mostra o fluxo completo de registro de um novo pedido de serviço, desde a interação do atendente até a persistência no banco de dados.

```
sequenceDiagram
    actor Atendente
    participant Frontend
    participant AuthController
    participant RequestController
    participant RequestService
    participant CitizenService
    participant ServiceTypeService
    participant RequestRepository
    participant Database

    Atendente->>Frontend: Preenche formulário de pedido
    Frontend->>AuthController: Verifica autenticação
    AuthController-->>Frontend: Token válido

    Frontend->>RequestController: POST /api/requests
    RequestController->>RequestService: createRequest(requestDTO)

    RequestService->>CitizenService: getCitizen(citizenId)
    CitizenService->>Database: SELECT FROM cidadaos
    Database-->>CitizenService: Dados do cidadão
    CitizenService-->>RequestService: Objeto Cidadão

    RequestService->>ServiceTypeService: getServiceType(typeId)
    ServiceTypeService->>Database: SELECT FROM tipos_servicos
    Database-->>ServiceTypeService: Dados do tipo de serviço
    ServiceTypeService-->>RequestService: Objeto TipoServico

    RequestService->>RequestService: Gera código de acompanhamento
    RequestService->>RequestService: Calcula data prevista
    RequestService->>RequestService: Define etapa inicial

    RequestService->>RequestRepository: save(request)
    RequestRepository->>Database: INSERT INTO pedidos
    Database-->>RequestRepository: ID do pedido criado
```

```
    RequestService->>RequestService: Registra histórico inicial
    RequestService->>Database: INSERT INTO historico_pedidos
    Database-->>RequestService: Confirmação

    RequestService-->>RequestController: Pedido criado com sucesso
    RequestController-->>Frontend: Resposta 201 Created com dados do pedido
    Frontend-->>Atendente: Exibe confirmação e código de acompanhamento
```

2. Consulta de Status de Pedido pelo Cidadão

Este diagrama ilustra como um cidadão pode consultar o status de seu pedido usando o código de acompanhamento.

```
sequenceDiagram
    actor Cidadão
    participant Frontend
    participant TrackingController
    participant RequestService
    participant HistoryService
    participant Database

    Cidadão->>Frontend: Insere código de acompanhamento
    Frontend->>TrackingController: GET /api/tracking/{codigo}

    TrackingController->>RequestService: getRequestByTrackingCode(codigo)
    RequestService->>Database: SELECT FROM pedidos WHERE codigo_acompanhamento = ?
    Database-->>RequestService: Dados do pedido

    alt Pedido não encontrado
        RequestService-->>TrackingController: Pedido não encontrado
        TrackingController-->>Frontend: 404 Not Found
        Frontend-->>Cidadão: Exibe mensagem de erro
    else Pedido encontrado
        RequestService-->>TrackingController: Dados do pedido

        TrackingController->>HistoryService: getRequestHistory(pedidoId)
        HistoryService->>Database: SELECT FROM historico_pedidos WHERE pedido_id = ?
        Database-->>HistoryService: Histórico do pedido
        HistoryService-->>TrackingController: Lista de eventos do histórico

        TrackingController-->>Frontend: 200 OK com dados do pedido e histórico
        Frontend-->>Cidadão: Exibe informações do pedido e status atual
    end
```

3. Fluxo de Aprovação de Documentos

Este diagrama mostra o processo de upload, análise e aprovação de documentos em um pedido.

```
sequenceDiagram
    actor Usuário
    participant Frontend
    participant DocumentController
    participant DocumentService
    participant RequestService
    participant StorageService
    participant Database

    Usuário->>Frontend: Seleciona documento e faz upload
    Frontend->>DocumentController: POST /api/documents/upload

    DocumentController->>StorageService: storeFile(file)
    StorageService-->>DocumentController: Caminho do arquivo salvo

    DocumentController->>DocumentService: createDocument(documentDTO)
    DocumentService->>RequestService: getRequest(requestId)
    RequestService->>Database: SELECT FROM pedidos
    Database-->>RequestService: Dados do pedido
    RequestService-->>DocumentService: Objeto Pedido

    DocumentService->>Database: INSERT INTO documentos_pedido
    Database-->>DocumentService: ID do documento criado
    DocumentService-->>DocumentController: Documento registrado
    DocumentController-->>Frontend: 201 Created com dados do documento
    Frontend-->>Usuário: Confirmação de upload

    actor Técnico
    Técnico->>Frontend: Acessa lista de documentos pendentes
    Frontend->>DocumentController: GET /api/documents/pending
    DocumentController->>DocumentService: getPendingDocuments()
    DocumentService->>Database: SELECT FROM documentos_pedido WHERE aprovado IS NULL
    Database-->>DocumentService: Lista de documentos pendentes
    DocumentService-->>DocumentController: Lista de documentos
    DocumentController-->>Frontend: 200 OK com lista de documentos
    Frontend-->>Técnico: Exibe documentos para análise

    Técnico->>Frontend: Aprova/rejeita documento
    Frontend->>DocumentController: PUT /api/documents/{id}/approve
    DocumentController->>DocumentService: approveDocument(id, approved, comments)
    DocumentService->>Database: UPDATE documentos_pedido SET aprovado = ?, observacao = ?
```

```
    Database-->>DocumentService: Confirmação

    DocumentService->>RequestService: updateRequestStatus(requestId)
    RequestService->>Database: UPDATE pedidos SET status_id = ?, etapa_atual_id = ?
    Database-->>RequestService: Confirmação
    RequestService-->>DocumentService: Status atualizado

    DocumentService-->>DocumentController: Documento processado
    DocumentController-->>Frontend: 200 OK com resultado
    Frontend-->>Técnico: Confirmação da ação
```

4. Fluxo de Pagamento

Este diagrama ilustra o processo de geração, pagamento e confirmação de uma taxa municipal.

```
sequenceDiagram
    actor Atendente
    participant Frontend
    participant PaymentController
    participant PaymentService
    participant RequestService
    participant Database
    actor Cidadão

    Atendente->>Frontend: Gera cobrança para pedido
    Frontend->>PaymentController: POST /api/payments

    PaymentController->>PaymentService: createPayment(paymentDTO)
    PaymentService->>RequestService: getRequest(requestId)
    RequestService->>Database: SELECT FROM pedidos
    Database-->>RequestService: Dados do pedido
    RequestService-->>PaymentService: Objeto Pedido

    PaymentService->>PaymentService: Calcula valor e gera código de barras
    PaymentService->>Database: INSERT INTO pagamentos
    Database-->>PaymentService: ID do pagamento criado

    alt Pagamento parcelado
        PaymentService->>PaymentService: Calcula parcelas
        PaymentService->>Database: INSERT INTO parcelas_pagamento
        Database-->>PaymentService: Confirmação
    end

    PaymentService-->>PaymentController: Pagamento registrado
    PaymentController-->>Frontend: 201 Created com dados do pagamento
```

```
    Frontend-->>Atendente: Exibe informações de pagamento

    Atendente-->>Cidadão: Informa código de barras/link

    Cidadão->>Cidadão: Realiza pagamento externamente

    Atendente->>Frontend: Registra pagamento
    Frontend->>PaymentController: PUT /api/payments/{id}/confirm

    PaymentController->>PaymentService: confirmPayment(id, paymentDate, receipt)
    PaymentService->>Database: UPDATE pagamentos SET status = 'PAGO', data_pagamento = ?
    Database-->>PaymentService: Confirmação

    PaymentService->>RequestService: updateRequestStatus(requestId)
    RequestService->>Database: UPDATE pedidos SET status_id = ?, etapa_atual_id = ?
    Database-->>RequestService: Confirmação

    PaymentService-->>PaymentController: Pagamento confirmado
    PaymentController-->>Frontend: 200 OK com resultado
    Frontend-->>Atendente: Confirmação da ação
```

5. Agendamento e Realização de Vistoria

Este diagrama mostra o processo de agendamento, realização e registro de resultado de uma vistoria.

```
sequenceDiagram
    actor Atendente
    participant Frontend
    participant InspectionController
    participant InspectionService
    participant RequestService
    participant Database
    actor Fiscal

    Atendente->>Frontend: Agenda vistoria
    Frontend->>InspectionController: POST /api/inspections

    InspectionController->>InspectionService: scheduleInspection(inspectionDTO)
    InspectionService->>RequestService: getRequest(requestId)
    RequestService->>Database: SELECT FROM pedidos
    Database-->>RequestService: Dados do pedido
    RequestService-->>InspectionService: Objeto Pedido

    InspectionService->>Database: INSERT INTO vistorias
    Database-->>InspectionService: ID da vistoria criada
```

```
InspectionService->>RequestService: updateRequestStatus(requestId)
RequestService->>Database: UPDATE pedidos SET status_id = ?
Database-->>RequestService: Confirmação

InspectionService-->>InspectionController: Vistoria agendada
InspectionController-->>Frontend: 201 Created com dados da vistoria
Frontend-->>Atendente: Confirmação do agendamento

Fiscal->>Frontend: Acessa lista de vistorias agendadas
Frontend->>InspectionController: GET /api/inspections/scheduled
InspectionController->>InspectionService: getScheduledInspections()
InspectionService->>Database: SELECT FROM vistorias WHERE status = 'AGENDADA'
Database-->>InspectionService: Lista de vistorias
InspectionService-->>InspectionController: Lista de vistorias
InspectionController-->>Frontend: 200 OK com lista
Frontend-->>Fiscal: Exibe vistorias agendadas

Fiscal->>Frontend: Registra resultado da vistoria
Frontend->>InspectionController: PUT /api/inspections/{id}/complete

InspectionController->>InspectionService: completeInspection(id, result, observations, p
InspectionService->>Database: UPDATE vistorias SET status = 'REALIZADA', resultado = ?,
Database-->>InspectionService: Confirmação

alt Com fotos
    InspectionService->>Database: INSERT INTO fotos_vistoria
    Database-->>InspectionService: Confirmação
end

InspectionService->>RequestService: updateRequestStatus(requestId)
RequestService->>Database: UPDATE pedidos SET status_id = ?
Database-->>RequestService: Confirmação

InspectionService-->>InspectionController: Vistoria concluída
InspectionController-->>Frontend: 200 OK com resultado
Frontend-->>Fiscal: Confirmação do registro
```

Estes diagramas representam os principais fluxos da aplicação Simple. Eles servem como referência para entender como os diferentes componentes interagem e como os processos são executados no sistema.