1. OBJETIVO:

Este documento tem como principal objetivo, descrever a arquitetura e detalhes de consumo e comunicação entre aplicação cliente, orquestrador e operadora.

2. O QUE É

A **API Orquestrador** funciona como um **Hub de Integração (Facade/Adapter)** centralizado. O seu principal objetivo é abstrair a complexidade de comunicação com diversas operadoras de planos de saúde e odontológicos (Bradesco Saúde, SulAmérica, Unimed, Amil, etc.).
Em vez de o sistema cliente (como um ERP Protheus, ou um portal de RH) precisar conhecer as regras, URLs, tokens e formatos de payload específicos de cada operadora de saúde, ele se comunica **apenas com o Orquestrador**, utilizando um modelo de dados único e padronizado (RequestBase).
Principais finalidades:

1. **Padronizar Movimentações Cadastrais:** Inclusão (I), Alteração (A), Exclusão (E), Troca de Plano (T) e Reativação (R) de vidas (beneficiários titulares e dependentes).
2. **Consultar Dados:** Buscar beneficiários, movimentações e redes cadenciadas/locais na operadora.
3. **Automação de Faturas:** Intermediar chamadas para robôs (RPA) para download e interpretação de faturas quando a operadora não possui APIs públicas.
4. **Tratamento de Retornos:** Receber os status das operadoras de forma síncrona ou assíncrona (via WebHooks) e traduzir os erros/sucessos para um formato comum.

3. MODELO CONCEITUAL E ARQUITETURA

O modelo conceitual do sistema é baseado em **Roteamento Inteligente (Mediation)** e no princípio de **Adapters** (Tradutores).

3.1. AUTENTICAÇÃO E MULTI-TENANCY (INQUILINOS)

A plataforma atende múltiplos clientes. A entrada exige *ClientId*, *ClientSecret* e *TenantId* gerando um token JWT. O appsettings.json armazena as URLs e credenciais de acesso para cada operadora atreladas àquele *TenantId*.

3.2. ENTRADAENTRADA PADRONIZADA (REQUESTBASE)

O sistema cliente monta um JSON padronizado com os dados do funcionário/dependente acompanhado da intenção (Movimento) e do destino (CNPJProvedor).

3.3. ROTEAMENTO (PADRÃO MEDIATOR)

Quando o OrquestradorController recebe a requisição, ele a delega para a classe Mediator. O Mediador atua como uma **chave de roteamento**:

- *Qual é a operadora?* (Olha o CNPJ: Bradesco, SulAmérica, Unimed...)
- *Qual é a ação?* (Olha o Movimento: Inclusão, Exclusão...)

Com base nisso, o Mediador direciona a execução para o *Handler* (Tratador) correto.

3.4. HANDLERS E CONVERSÕES (O CORE BUSINESS)

Para cada combo (Operadora + Movimento), existe um **Handler** específico (ex: HandlerPostMovimentacoesVidasInclusoesBradesco). O papel principal do Handler é:

- **Conversão de Request:** Transformar o objeto genérico RequestBase nas classes exatas que a operadora de destino espera (ex: RequestPostMovimentacoesVidasInclusoesBradesco).
- **Integração:** Realizar chamadas HTTP externas REST/SOAP para os barramentos das operadoras (ex: APIs na plataforma Sensedia da SulAmérica ou Endpoints do Bradesco).

3.5. RETORNO PADRONIZADO (RESPONSEBASESUCCESS)

Independente de como o provedor devolve os dados e os formatos de erro, a resposta final trafega por outro Adapter voltando ao formato comum ResponseBaseSuccess, mapeando:

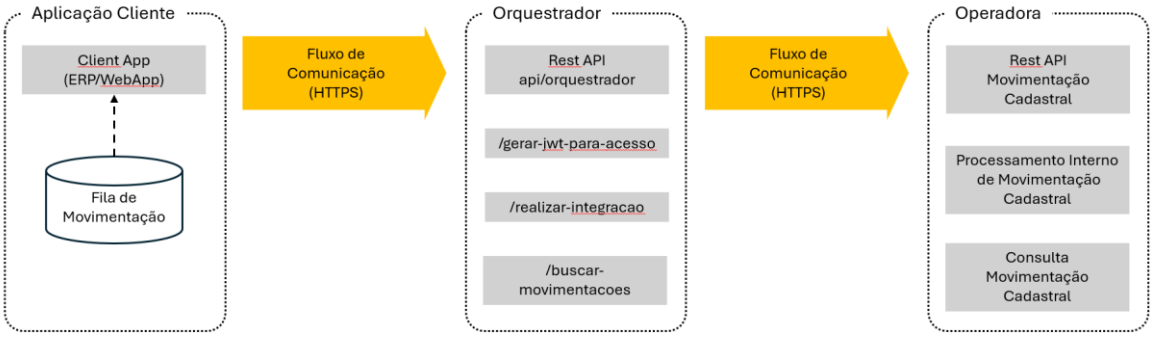
- O status HTTP simulado.
- Uma lista padronizada de validações ou erros que a operadora acusou (ex: "CPF inválido", "Beneficiário inativo").
- O payload original (RequestJson), para fins de rastreabilidade (auditoria).

3.6. RESUMO DO FLUXO DO ORQUESTRADOR

Sistema/Portal ➔ RequestBase Único ➔ Controller ➔ Mediator Roteia por CNPJ ➔ Handler Converte para Dialeto da Operadora ➔ API Operadora ➔ Converte para Resposta Padrão ➔ Sistema/Portal

3.7. MODELO DE COMUNICAÇÃO

A aplicação cliente deve disparar a movimentação cadastral em intervalos regulares executando a autenticação inicial para a criação de um token de acesso. Com esse token é possível realizar as outras ações de envio de movimentação cadastral e consulta movimentações na operadora. Atenção: A operadora requer um tempo de processamento da movimentação então é aconselhado executar a movimentação cadastral e busca por movimentações de forma intercalada.



4. AUTENTICAÇÃO

A autenticação na API do Orquestrador é baseada no padrão **JWT (JSON Web Token)**. Antes de consumir qualquer endpoint de negócio (como o de integrações), o cliente precisa comprovar sua identidade e obter um token de acesso temporário.

A porta de entrada é o endpoint **POST /api/orquestrador/gerar-jwt-para-acesso**. Ele possui a anotação **[AllowAnonymous]**, o que significa que é o único local da API que não exige um token prévio para ser acessado.

4.1. PAYLOAD DE ENTRADA (RequestSignIn)

O cliente (ex: um portal de RH ou ERP) envia um JSON contendo três informações principais:

- **ClientId**: O identificador único do cliente.
- **ClientSecret**: A chave secreta (senha) do cliente.
- **TenantId**: O identificador do inquilino (cliente/empresa de destino).

4.2. RETORNO AO CLIENTE (ResponseSuccessCreateJsonWebToken)

O endpoint responde um **200 OK** devolvendo um objeto JSON para o cliente contendo:

- **token:** A string JWT codificada (Header, Payload e Signature).
- **dataCriacao**: A data/hora (DateTime) em que o token foi gerado.
- **dataExpiracao**: A data/hora exata em que o token perderá a validade (criação + 2 horas).

4.3. COMO USAR O TOKEN NA PRÁTICA

De posse do token gerado, o cliente deve incluí-lo no **Header HTTP** de todas as chamadas subsequentes para os endpoints protegidos (**[Authorize]**, como o **/api/orquestrador/realizar-integracao**), utilizando o formato padrão de autorização do tipo Bearer: **Authorization: Bearer <TOKEN_AQUI>**

5. REALIZAR A MOVIMENTAÇÃO CADASTRAL

O endpoint **/api/orquestrador/realizar-integracao** é o núcleo operacional da API. Ele atua como um tradutor universal, recebendo um modelo de dados padrão (agnóstico de operadora) e devolvendo uma resposta também padronizada, independentemente de qual operadora de saúde (Bradesco, SulAmérica, Unimed, etc.) foi consultada internamente.

5.1. REQUEST (RequestBase)

A requisição é feita enviando um JSON no corpo (Body) que representa a classe RequestBase. O objetivo desse modelo é concentrar todos os dados possíveis de uma movimentação cadastral.

Estrutura principal da Entrada: Para que o Orquestrador saiba o que fazer, dois campos são **obrigatórios** na raiz do JSON:

1. **cnpjProvedor**: Define para qual operadora a requisição será roteada (ex: Bradesco, SulAmérica).

2. **movimento**: Define o tipo de ação. Pode ser:
    - **I** (Inclusão)
    - **A** (Alteração)
    - **E** (Exclusão)
    - **T** (Troca de Plano)
    - **R** (Reativação)

Nós Condicionais: Dependendo do movimento escolhido, você preenche blocos de dados específicos na raiz do JSON:

- Se **movimento = "I"**, preenche-se o array **"inclusao": [...]** (com dados do titular e dependentes, endereço, etc.).
- Se **movimento = "A"**, preenche-se o objeto **"alteracao": { ... }**.
- Se **movimento = "E"**, preenche-se o objeto **"exclusao": { ... }**.

Exemplo de Payload de Entrada (Inclusão):

```
{
  "cnpjProvedor": "33055146000193",
  "movimento": "I",
  "inclusao": [
    {
      "nome": "João da Silva",
      "cpf": "12345678900",
      "dataNascimento": "1990-01-01T00:00:00",
      "grauParentesco": "0",
      "endereco": { ... },
      "contato": { ... },
      "produto": { ... }
    }
  ]
}
```

5.2. A SAÍDA (ResponseBaseSuccess)

Independentemente de como a operadora de destino formata suas respostas (seja um XML legado, um JSON complexo da SulAmérica ou do Bradesco), o Orquestrador sempre converte o resultado para um modelo único chamado ResponseBaseSuccess.

Isso facilita a vida do sistema cliente, que não precisa programar tratamentos de erro específicos para cada operadora.

Estrutura principal da Saída:

- **status:** O código de status HTTP resultante do processamento na operadora (ex: 200 sucesso, 400 erro de negócio na operadora).
- **movimentacoes:** Uma lista consolidada com o resultado da ação (ex: número da carteirinha gerada, ID da transação).
- **validacoes:** Uma lista de mensagens críticas ou erros devolvidos pela regra de negócio da operadora (ex: "CPF inválido", "Data de adesão fora da vigência").
- **requestURI:** A URL exata da operadora que o Orquestrador chamou por baixo dos panos.
- **requestJson:** O payload *exato* que o Orquestrador traduziu e enviou para a operadora (ótimo para auditoria e troubleshooting).
- **responseContent:** O retorno cru (raw string) devolvido pela API da operadora.

Exemplo de payload de Saída:


```
{
  "status": 200,
  "movimentacoes": [
    {
      "nome": "João da Silva",
      "carteirinha": "8888999900001111",
      "statusMovimentacao": "Processado com Sucesso"
    }
  ],
  "validacoes": [],
  "requestURI": "https://api.operadora.com.br/v1/inclusoes",
  "requestJson": { "nomeBeneficiario": "João da Silva", "doc_cpf": "12345678900" },
  "responseContent": "{ \"success\": true, \"id\": 9988 }"
}
```


Em resumo: Você entrega o RequestBase padronizado. O Orquestrador se vira para traduzir, autenticar (gerar token OAuth na operadora correspondente) e chamar a API de destino, devolvendo tudo mastigado no ResponseBaseSuccess.

6. CONSULTAR O PROCESSAMENTO DAS MOVIMENTAÇÕES

O endpoint **/api/orquestrador/buscar-movimentacoes** tem como objetivo centralizar a consulta do **status e histórico de movimentações** (inclusões, alterações, exclusões) enviadas previamente para as operadoras (atualmente suportando SulAmérica e Bradesco).

Assim como o endpoint de integração, ele adota o princípio de unificar a comunicação através de um modelo padrão de entrada e saída.

6.1. REQUEST (RequestBase)

A requisição é feita enviando o mesmo JSON padrão (RequestBase), mas para este endpoint de busca, a atenção é voltada para o nó **consulta**. O campo movimento normalmente é preenchido com **"C"** (Consulta).

Campos obrigatórios principais:

- cnpjProvedor: Define qual operadora será consultada (ex: CNPJ da SulAmérica ou Bradesco).
- movimento: "C" (Consulta).
- consulta: O objeto contendo os filtros de busca.

O objeto consulta (Filtros): A classe Consulta_RequestBase possui dezenas de filtros possíveis. A operadora de destino determina quais filtros são válidos, mas os mais utilizados para buscar movimentações incluem:

- **apolice**: Número da apólice/contrato.
- **empresa / cnpj**: Dados da empresa estipulante.
- **cpf / cpfDependente**: Para filtrar as movimentações de um beneficiário específico.
- **dataMovimentacaoApolice**: Para buscar movimentações realizadas em uma data ou período específico.
- **pagina e quantidadePorPagina**: Parâmetros de paginação (muito comuns no retorno da SulAmérica, por exemplo).

Exemplo de Payload de Entrada:

```
{
  "cnpjProvedor": "01685053000156",
  "movimento": "C",
  "consulta": {
    "apolice": "12345",
    "empresa": "999999",
    "cpf": "12345678900",
    "dataMovimentacaoApolice": "2026-04-01",
    "pagina": "1",
    "quantidadePorPagina": "50"
  }
}
```


6.2. **RESPONSE (Response)**

A saída deste endpoint é um retorno padronizado que traduz o status atual das solicitações na base da operadora. Em vez de lidar com os diferentes formatos de rastreio de cada plano de saúde, o cliente recebe uma lista unificada de ocorrências.

A resposta geralmente contém, para cada vida pesquisada:

- **Status da Movimentação:** Se a inclusão/exclusão está em análise, foi processada com sucesso, ou se foi rejeitada (crítica).
- **Protocolo ou ID da Transação:** O número de rastreio gerado pela operadora.
- **Motivo/Crítica:** Em caso de erro ou rejeição, o detalhe preenchido pela operadora do porquê a movimentação não foi concluída (ex: "Falta documento X", "Fora do prazo de vigência").
- **Dados Básicos do Beneficiário:** Nome, CPF e número da carteirinha (se já tiver sido gerada).

Exemplo conceitual de Payload de Saída:


```
{
  "status": 200,
  "movimentacoes": [
    {
      "nome": "João da Silva",
      "cpf": "12345678900",
      "tipoMovimento": "Inclusão",
      "dataMovimentacao": "2026-04-01T10:30:00",
      "statusMovimentacao": "Processado com Sucesso",
      "numeroCarteirinha": "8888999900001111",
      "criticas": []
    },
    {
      "nome": "Maria da Silva (Dependente)",
      "cpf": "09876543211",
      "tipoMovimento": "Inclusão",
      "dataMovimentacao": "2026-04-01T10:30:00",
      "statusMovimentacao": "Rejeitado",
      "numeroCarteirinha": null,
      "criticas": [
        "Data de nascimento incompatível com o grau de parentesco informado."
      ]
    }
  ],
  "requestURI": "https://apisulamerica.../movimentacoes-vidas",
  "requestJson": { ... }
}
```


Em resumo: O endpoint **/api/orquestrador/buscar-movimentacoes** permite que o seu sistema realize um *tracking* (acompanhamento) passivo. Você passa quem você quer procurar (nó consulta) e a API do Orquestrador se encarrega de ir até a operadora, consultar o portal/API deles e trazer o status traduzido.

7. CONCLUSÃO: O ECOSSISTEMA DA API ORQUESTRADOR

A **API Orquestrador** foi arquitetada para ser o "coração" das integrações sistêmicas de benefícios corporativos, atuando como um poderoso **Hub e Tradutor Universal (Facade/Mediator)** entre sistemas clientes (como o Protheus ou Hub RH) e as complexas (e muitas vezes distintas) APIs das operadoras de saúde (Bradesco, SulAmérica, Unimed, etc.).

Em suma: O Orquestrador encapsula a complexidade, padroniza as entradas/saídas e fornece resiliência. Para quem o consome, não importa se amanhã o Bradesco mudar a versão da sua própria API ou se uma nova operadora for adicionada; o contrato (RequestBase e ResponseBaseSuccess) e a forma de consumo permanecerão simples e inalterados para o cliente final.