Diretrizes:

Revisão do projeto: Reescreva os casos de uso detalhados até aqui para refletir as mudanças pedidas pelo cliente e escreva novos casos de uso, caso sejam necessários. Baseado nessas mudanças, você irá atualizar o Diagrama de Classes e o Dicionário de Dados. Faça a revisão das partes implementadas para refletir todas as mudanças pedidas.

- **Especificação**: Prepare os diagramas de sequência para os casos de uso previamente especificados.
- **Projeto**: Forneça uma lista de todos os cabeçalhos de métodos que você adicionaria a cada uma de suas classes com base nos diagramas de sequência e diagramas de comunicação que você preparou. Além disso, indique quaisquer novas classes, atributos ou relações que seriam necessárias.
- **Implementação**: Desenvolva os métodos de classes baseados nos diagramas produzidos para as classes projetadas.

➢ Dicionário de dados:

1. Usuário

Tabela que armazena as informações dos usuários do sistema.

Coluna	Tipo de Dados	Descrição
Id	int	Identificador único do usuário (PK).
Nome	varchar(100)	Nome completo do usuário.
Email	varchar(100)	Endereço de e-mail do usuário.
Telefone	varchar(15)	Número de telefone do usuário.
Senha	varchar(255)	Senha do usuário para login, devidamente criptografada.

2. PedidoReembolso

Tabela que armazena os pedidos de envio enviados pelos usuários.

Coluna	Tipo de Dados	Descrição
Id	int	Identificador único do pedido de reembolso (PK).
DataSubmissao	datetime	Dados e hora em que o pedido foi enviado.
ValorTotal	decimal(10,2)	Valor total do pedido de reembolso.

Coluna	Tipo de Dados	Descrição
StatusPedido	int	Referência ao status atual do pedido (FK para StatusPedido).
UsuarioId	int	Identificador do usuário que enviou o pedido (FK para Usuario).

3. DespesaMedica

Tabela que armazena as despesas médicas associadas a um pedido de reembolso.

Coluna	Tipo de Dados	Descrição
Id	int	Identificador único de despesa médica (PK).
TipoDespesa	varchar(50)	Tipo de despesa médica (ex.: consulta, exame, cirurgia, etc.).
Valor	decimal(10,2)	Valor da despesa médica.
DataDespesa	date	Dados relativos a despesas médicas realizadas.
PedidoReembolsoId	int	Identificador do pedido de reembolso associado (FK para PedidoReembolso).

4. Documento

Tabela que armazena documentos anexados ao pedido de envio.

Coluna	Tipo de Dados	Descrição
Id	int	Identificador único do documento (PK).
TipoDocumento	varchar(50)	Tipo do documento (ex.: recibo, laudo médico, etc.).
CaminhoArquivo	varchar(255)	Caminho onde o arquivo está armazenado.
PedidoReembolsoId	int	Identificador do pedido de reembolso associado (FK para PedidoReembolso).

5. StatusPedido

Tabela que define os diferentes estados pelos quais um pedido de reembolso pode passar.

Coluna	Tipo de Dados	Descrição
IdStatus	int	Identificador único de status (PK).
DescricaoStatus	varchar(50)	Descrição do status (ex.: Pendente, Aprovado, Indeferido, etc.).

6. Notificação

Tabela que armazena notificações enviadas ao usuário sobre o status dos pedidos.

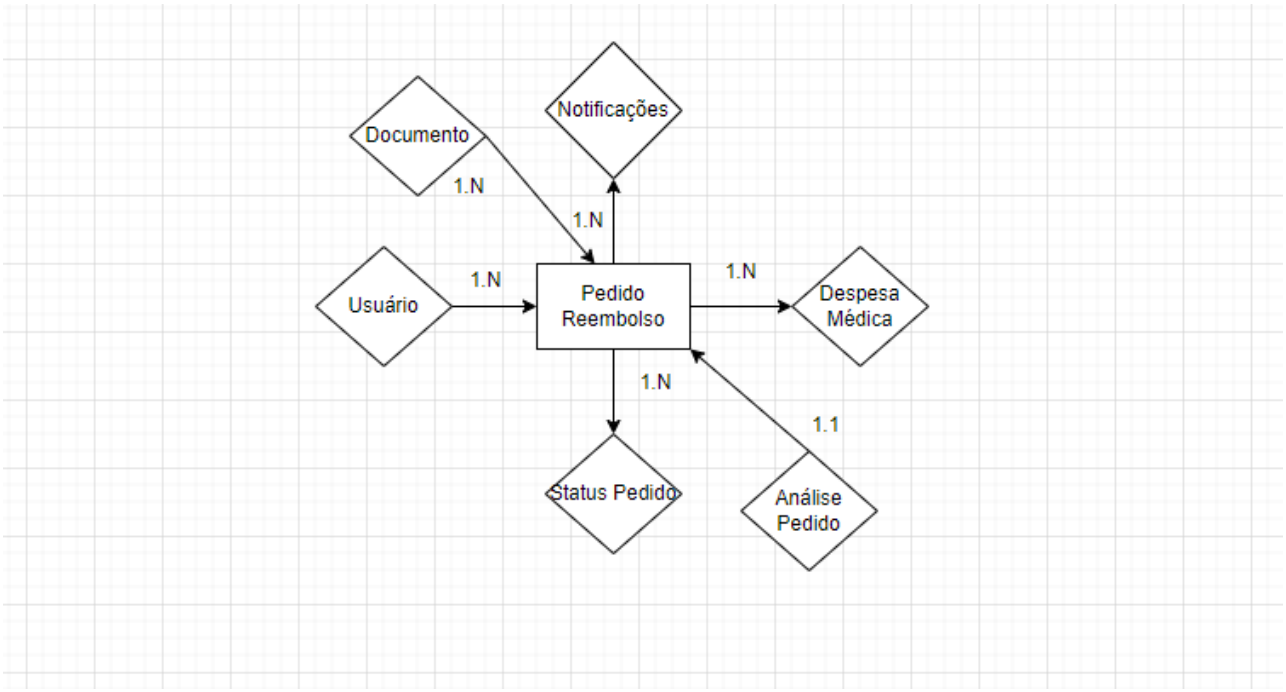
Coluna	Tipo de Dados	Descrição
Id	int	Identificador único da notificação (PK).
Mensagem	varchar(255)	Mensagem enviada ao usuário.
DataEnvio	datetime	Dados e hora em que a notificação foi enviada.
UsuarioId	int	Identificador do usuário que recebeu uma notificação (FK para Usuario).
TipoNotificacao	varchar(50)	Tipo de notificação (ex.: E-mail, SMS).

7. AnalisePedido

Tabela que armazena as análises feitas pela equipe de suporte ao cliente para um pedido de reembolso.

Coluna	Tipo de Dados	Descrição
Id	int	Identificador único da análise (PK).
Comentario	varchar(255)	Comentário detalhado sobre a análise do pedido.
DataAnalise	datetime	Dados e hora em que a análise foi feita.
PedidoReembolsoId	int	Identificador do pedido de reembolso detalhado (FK para PedidoReembolso).

- Diagrama de classe:



https://app.diagrams.net/#G1bKh0qU9QtryNsBd3tzdNkgLrKPh1MtaL#%7B%22pageId%22%3A%22LkH4FCj_PupRZ8XCfTFf%22%7D

- Diagrama de Sequência:

Casos de uso:

- **Caso de Uso 1**: Submissão de Pedido de Reembolso
- **Caso de Uso 2**: Acompanhamento do Pedido de Reembolso
- **Caso de Uso 3**: Análise e Processamento do Pedido de Reembolso
- **Caso de Uso 4**: Notificação do Status do Pedido

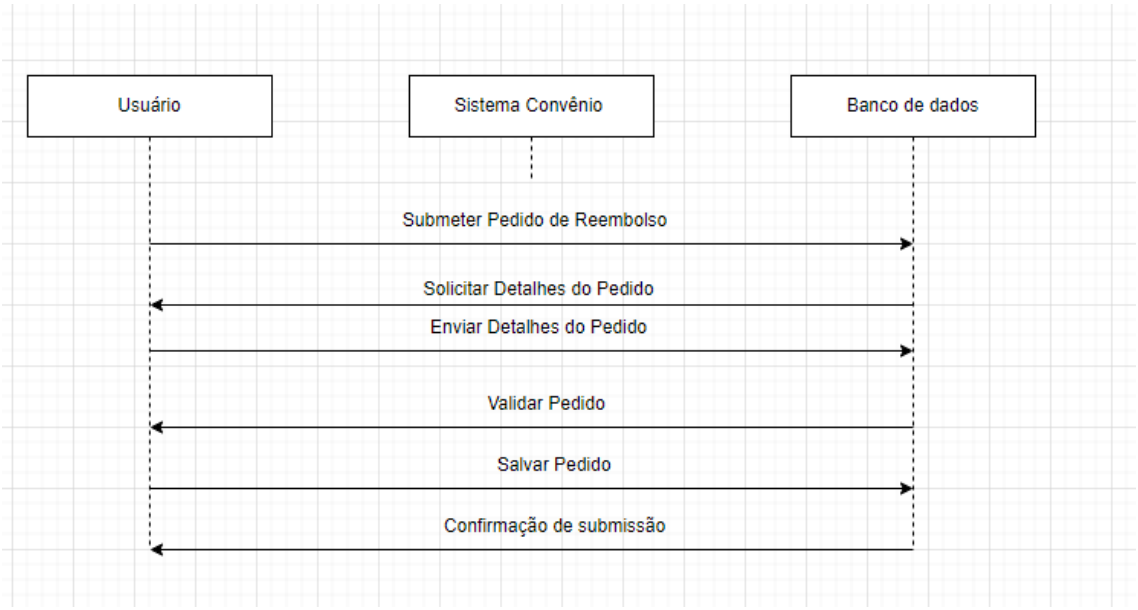
Caso de uso 1: Submissão de Pedido de Reembolso

Atores:

- **Usuário (Titular do Plano de Saúde)**
- **Sistema de Convênio (do Plano de Saúde)**
- **Banco de Dados**

Fluxo:

1. O **usuário** acessa o sistema e seleciona a opção de envio de um pedido de reembolso.
2. O sistema solicita os detalhes do pedido (documentos, despesas médicas, valor total).
3. O **usuário** preencha os detalhes e envie a solicitação.
4. O **Sistema de Convênio de Plano de Saúde** valida os dados.
5. O **Sistema** salva o pedido no **Banco de Dados.**
6. O sistema exibe uma confirmação para o **usuário.**



https://app.diagrams.net/#G1bKh0qU9QtryNsBd3tzdNkgLrKPh1MtaL#%7B%22pageId%22%3A%22Sn9oRUTqcVAlEyNHzdA5%22%7D

Caso de uso2: Acompanhamento do Pedido de Reembolso

Atores:

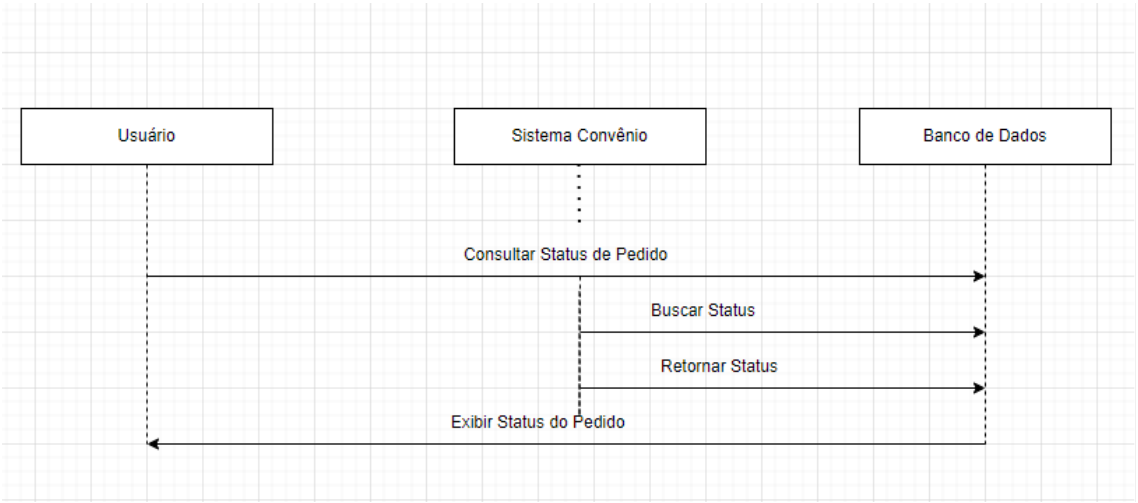
- **Usuário**
- **Sistema de Convênio de Plano de Saúde**
- **Banco de Dados**

Fluxo:

Fluxo:

1. O **usuário** solicita o status do pedido.
2. O **Sistema** recupera os detalhes do pedido do **Banco de Dados** .
3. O **Sistema** apresenta o status atual do pedido ao **Usuário** .

Diagrama de Sequência



https://app.diagrams.net/#G1bKh0qU9QtryNsBd3tzdNkgLrKPh1MtaL#%7B%22pageId%22%3A%22teLOjlEkx2PsKB4XdogQ%22%7D

Caso de uso3:Análise e Processamento do Pedido de Reembolso

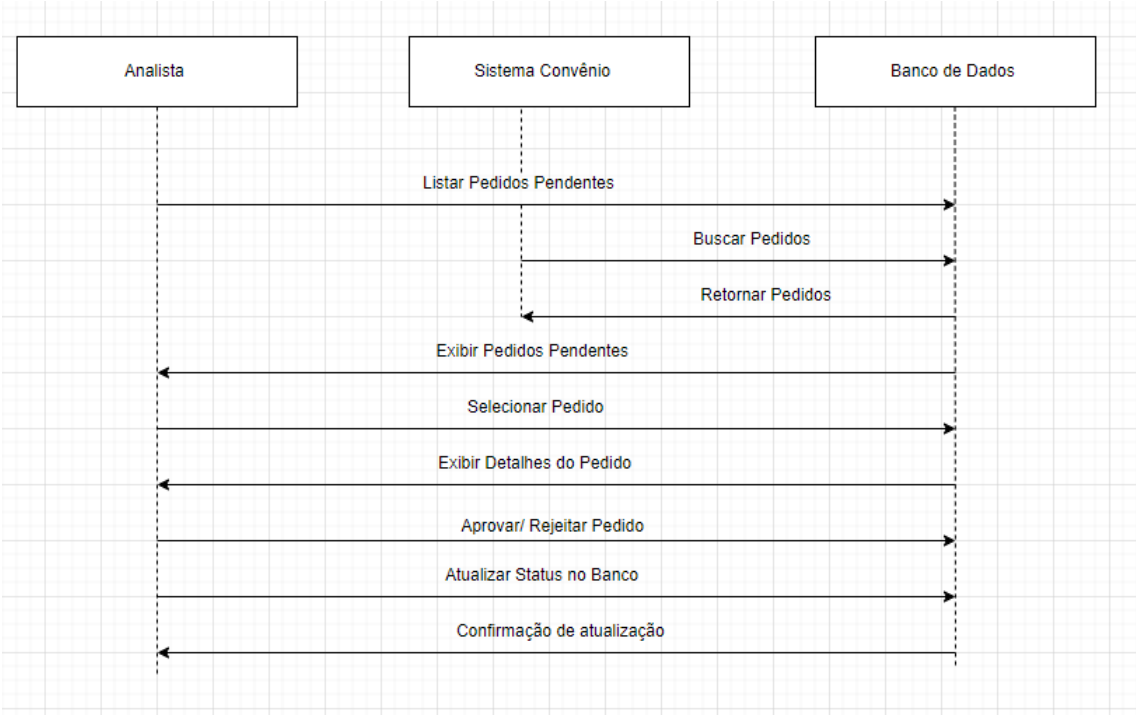
Atores:

- **Administrador (Analista de Pedidos)**
- **Sistema de Convênio de Plano de Saúde**
- **Banco de Dados**

Fluxo:

1. O **Analista** acessa o sistema e solicita uma lista de pedidos pendentes.
2. O **Sistema** busca os pedidos pendentes no **Banco de Dados** .
3. O **Analista** seleciona um pedido para análise.
4. O **Sistema** exibe os detalhes do pedido.
5. O **Analista** decide aprovar ou rejeitar o pedido.
6. O **Sistema** atualiza o status do pedido no **Banco de Dados** .

Diagrama de Sequência:



https://app.diagrams.net/#G1bKh0qU9QtryNsBd3tzdNkgLrKPh1MtaL#%7B%22pageId%22%3A%229gYkUYrbYS0N24C2Vjhw%22%7D

Caso de uso 4: Notificação do Status do Pedido

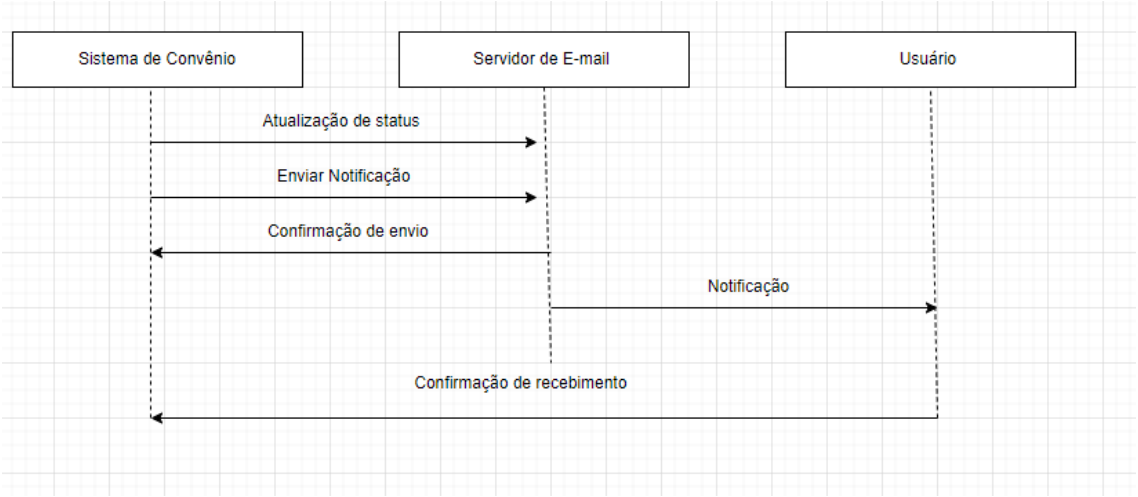
Atores:

- **Sistema de Convênio de Plano de Saúde**
- **Servidor de E-mail**
- **Usuário**

Fluxo:

4. O status do pedido é atualizado (aprovado ou rejeitado).
5. O **Sistema** envia uma notificação para o **Usuário** via **Servidor de E-mail**.
6. O **Servidor de E-mail** entrega a notificação ao **Usuário**.

Diagrama de Sequência:



https://app.diagrams.net/#G1bKh0qU9QtryNsBd3tzdNkgLrKPh1MtaL#%7B%22pageId%22%3A%224_y0rWH56R9_OjO9Vjy7%22%7D

- Lista de todos os cabeçalhos:

1. Classe Usuario

O usuário submete e acompanha os pedidos de reembolso. Uma classe já existe, mas pode incluir novos métodos.

Métodos:

public PedidoReembolso SubmeterPedidoReembolso(PedidoReembolso pedido);
public PedidoReembolso ConsultarStatusPedido(int pedidoId);

Novos Atributos:

- int UsuarioId
- string Nome
- string Email
- List<PedidoReembolso> PedidosReembolso

2. ClassePedidoReembolso

A classe PedidoReembolsoarmazena os dados relacionados a um pedido de reembolso.

Métodos:

public void Validar();
public void AtualizarStatus(StatusPedido novoStatus);

Novos Atributos:

- int PedidoId
- int UsuarioId
- DateTime DataSubmissao

- decimal Valor
- StatusPedido Status *(enum: Pendente, Aprovado, Reprovado)*
- string MotivoRejeicao *(Caso seja rejeitado)*
- string JustificativaAprovacao

3. ClasseAnalista

O analista revisa os pedidos pendentes e aprova ou rejeita.

Métodos;

public List<PedidoReembolso> ListarPedidosPendentes();
public PedidoReembolso AnalisarPedido(int pedidoId);
public void AprovarPedido(PedidoReembolso pedido);
public void RejeitarPedido(PedidoReembolso pedido, string motivoRejeicao);

Novos Atributos:

- int AnalistaId
- string Nome
- string Setor

4. ClasseSistemaConvenio

Esta classe representa o sistema de convênio de plano de saúde e interação com o sistema para processar e gerenciar os pedidos de envio.

Métodos:

public PedidoReembolso ReceberPedidoReembolso(PedidoReembolso pedido);
public void ProcessarPedidoReembolso(PedidoReembolso pedido);
public void EnviarNotificacaoStatus(PedidoReembolso pedido, Usuario usuario);

Novos Atributos:

- string NomeConvenio
- List<PedidoReembolso> PedidosProcessados

5. ClasseFornecedorInformacao

O fornecedor fornece informações para o sistema de reembolso.

Métodos:

public InformacaoReembolso FornecerInformacoes(int pedidoId);

Novos Atributos:

- int FornecedorId
- string Nome
- List<InformacaoReembolso> InformacoesReembolso

6. ClasseServidorEmail (Nova Classe)

Esta classe seria responsável por enviar notificações de status aos usuários.

Métodos:

public void EnviarNotificacao(string email, string assunto, string mensagem);

Novos Atributos:

- string ServidorSMTP
- int Porta
- string Credenciais

7. EnumeraçãoStatusPedido

Enumeração para representar o status de um pedido de reembolso.

Valores:

public enum StatusPedido {
Pendente,
Aprovado,
Reprovado
}

Relações Entre Classes:

- Usuario**->**PedidoReembolso: Um usuário pode submeter vários pedidos de reembolso.
    - **Relacionamento**: Um-para-Muitos (Usuariocontém uma lista de PedidoReembolso)
- Analista**->**PedidoReembolso : Um analista revisa e decide o status de múltiplos pedidos.
    - **Relacionamento** : Um-para-Muitos (Analistaanalisa vários PedidoReembolso)
- SistemaConvenio**->**PedidoReembolso : O sistema de convênio gerencia e processa os pedidos.
    - **Relacionamento**: Um-para-Muitos (SistemaConvenio contém múltiplos PedidoReembolso)
- FornecedorInformacao→**I**nformacaoReembolso: O fornecedor fornece informações adicionais sobre o pedido.
    - **Relacionamento**: Um-para-Muitos (FornecedorInformacaocontém várias InformacaoReembolso)
- PedidoReembolso→**S**ervidorEmail: O servidor de e-mail notifica o usuário após o processamento.
    - **Relacionamento**: O Servidor E-mail envio de notificações sobre o PedidoReembolso

➢ Desenvolva os métodos de classes baseados em diagramas produzidos para as classes projetadas:

1. ClasseUsuario

```
public class Usuario
{
    public int UsuarioId { get; set; }
    public string Nome { get; set; }
    public string Email { get; set; }
    public List<PedidoReembolso> PedidosReembolso { get; set; }

    public Usuario()
    {
        PedidosReembolso = new List<PedidoReembolso>();
    }

    public PedidoReembolso SubmeterPedidoReembolso(PedidoReembolso pedido)
    {
        pedido.Status = StatusPedido.Pendente;
        PedidosReembolso.Add(pedido);
        // Lógica para enviar o pedido ao sistema
        SistemaConvenio.ReceberPedidoReembolso(pedido);
        return pedido;
    }

    public PedidoReembolso ConsultarStatusPedido(int pedidoId)
    {
        return PedidosReembolso.FirstOrDefault(p => p.PedidoId == pedidoId);
    }
}
```

2. ClassePedidoReembolso

```
public class PedidoReembolso
{
    public int PedidoId { get; set; }
    public int UsuarioId { get; set; }
    public DateTime DataSubmissao { get; set; }
    public decimal Valor { get; set; }
    public StatusPedido Status { get; set; }
    public string MotivoRejeicao { get; set; }
    public string JustificativaAprovacao { get; set; }

    public void Validar()
    {
        if (Valor <= 0)
        {
            throw new Exception("O valor do pedido deve ser maior que zero.");
        }
    }

    public void AtualizarStatus(StatusPedido novoStatus)
```

```
    {
        Status = novoStatus;
    }
}
```

3. ClasseAnalista

```
public class Analista
{
    public int AnalistaId { get; set; }
    public string Nome { get; set; }
    public string Setor { get; set; }

    public List<PedidoReembolso> ListarPedidosPendentes()
    {
        // Lógica para listar pedidos pendentes
        return SistemaConvenio.PedidosProcessados
                        .Where(p => p.Status == StatusPedido.Pendente)
                        .ToList();
    }

    public PedidoReembolso AnalisarPedido(int pedidoId)
    {
        var pedido = SistemaConvenio.PedidosProcessados.FirstOrDefault(p =>
p.PedidoId == pedidoId);
        if (pedido == null)
        {
            throw new Exception("Pedido não encontrado.");
        }
        return pedido;
    }

    public void AprovarPedido(PedidoReembolso pedido)
    {
        pedido.AtualizarStatus(StatusPedido.Aprovado);
        pedido.JustificativaAprovacao = "Pedido aprovado.";
        // Notificar o usuário
        SistemaConvenio.EnviarNotificacaoStatus(pedido,
UsuarioRepository.GetById(pedido.UsuarioId));
    }

    public void RejeitarPedido(PedidoReembolso pedido, string motivoRejeicao)
    {
        pedido.AtualizarStatus(StatusPedido.Reprovado);
        pedido.MotivoRejeicao = motivoRejeicao;
        // Notificar o usuário
        SistemaConvenio.EnviarNotificacaoStatus(pedido,
UsuarioRepository.GetById(pedido.UsuarioId));
    }
}
```

4. ClasseSistemaConvenio

```
public static class SistemaConvenio
{
   public static List<PedidoReembolso> PedidosProcessados { get; set; } = new
List<PedidoReembolso>();

   public static PedidoReembolso ReceberPedidoReembolso(PedidoReembolso
pedido)
   {
      pedido.DataSubmissao = DateTime.Now;
      PedidosProcessados.Add(pedido);
      // Processar o pedido
      ProcessarPedidoReembolso(pedido);
      return pedido;
   }

   public static void ProcessarPedidoReembolso(PedidoReembolso pedido)
   {
      // Simula o processamento do pedido
      if (pedido.Valor > 1000)
      {
         // Aprova se for menor que o valor limite
         pedido.AtualizarStatus(StatusPedido.Aprovado);
      }
      else
      {
         pedido.AtualizarStatus(StatusPedido.Reprovado);
         pedido.MotivoRejeicao = "Valor abaixo do mínimo exigido.";
      }
   }

   public static void EnviarNotificacaoStatus(PedidoReembolso pedido, Usuario
usuario)
   {
      string mensagem = $"Olá {usuario.Nome}, o status do seu pedido de reembolso
foi atualizado para {pedido.Status}.";
      if (pedido.Status == StatusPedido.Reprovado)
      {
         mensagem += $" Motivo: {pedido.MotivoRejeicao}";
      }
      ServidorEmail.EnviarNotificacao(usuario.Email, "Status do Pedido de
Reembolso", mensagem);
   }
}
```

5. ClasseFornecedorInformacao

```
public class FornecedorInformacao
{
```

```
    public int FornecedorId { get; set; }
    public string Nome { get; set; }
    public List<InformacaoReembolso> InformacoesReembolso { get; set; }

    public FornecedorInformacao()
    {
        InformacoesReembolso = new List<InformacaoReembolso>();
    }

    public InformacaoReembolso FornecerInformacoes(int pedidoId)
    {
        // Busca e fornece informações do pedido
        return InformacoesReembolso.FirstOrDefault(info => info.PedidoId == pedidoId);
    }
}
```

6. ClasseServidorEmail

```
public static class ServidorEmail
{
    public static string ServidorSMTP { get; set; } = "smtp.exemplo.com";
    public static int Porta { get; set; } = 587;
    public static string Credenciais { get; set; } = "credenciais_email";

    public static void EnviarNotificacao(string email, string assunto, string mensagem)
    {
        // Lógica para enviar email
        Console.WriteLine($"Enviando email para: {email}");
        Console.WriteLine($"Assunto: {assunto}");
        Console.WriteLine($"Mensagem: {mensagem}");
        // Aqui entraria a lógica real de envio de email
    }
}
```

7. EnumeraçãoStatusPedido

```
public enum StatusPedido
{
    Pendente,
    Aprovado,
    Reprovado
}
```

Novas entidades introduzidas:

1. ServidorEmail: Para enviar notificações aos usuários.
2. StatusPedido(Enum): Para representar os estados dos pedidos de reembolso.

3. InformacaoReembolso(Não detalhado): Provavelmente armazena dados fornecidos pelo FornecedorInformacao.