**Helps**

**Descrição do Projeto e Análise de Requisitos**

|  |  |
| --- | --- |
| Autores: Jean-Luc Bonnet, Lucas da Silva Carrasco, Victor Alexandre dos Santos, Pedro Bazaluk Machado Videira | Data de emissão:  16/08/23 |
| Revisor: | Data de revisão |

**FOLHA DE CONTROLE DE REVISÕES**

|  |  |  |
| --- | --- | --- |
| **Número da versão** | **Data de emissão** | **Registro de modificações** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Objetivo do projeto

O projeto tem como objetivo desenvolver um sistema para conectar os clientes que precisam de algum serviço domiciliar com prestadores autônomos dessas atividades. Proporcionando uma comunicação mais ágil e rápida, além da possibilidade de analisar as avaliações de cada prestador, possibilitando uma forma fácil de encontrar o profissional mais adequado para o problema encontrado.

# Requisitos do Software

Nesta seção serão detalhados os requisitos funcionais e não funcionais.

## Requisitos Funcionais

RF01: O sistema deve permitir que o cliente solicite um serviço.

RF02: O sistema deve permitir que o prestador receba e aceite um serviço.

RF03: O sistema deve permitir um contato entre o cliente e o prestador.

RF04: O sistema deve permitir marcar a data do serviço.

RF05: O sistema deve permitir o pagamento do serviço.

RF06: O sistema deve permitir a avaliação do serviço.

RF07: O sistema deve permitir o cadastro de clientes.

RF08: O sistema deve permitir o cadastro de prestadores.

RF09: O sistema deve permitir que o cliente receba e aceite um prestador.

## Requisitos não funcionais

RNF01: Os usuários do sistema devem se autenticar com login e senha.

RNF02: O sistema deve estar disponível 99,99%.

RNF03: O sistema deve armazenar as informações em um banco de dados SQL.

RNF04: O sistema deve ser compatível apenas com dispositivos móveis.

RNF05: O sistema deve atender à LGPD.

# Casos de Uso

## Diagrama de Casos de Uso

## 

Diagrama

Descrição gerada automaticamente

## Especificação dos Casos de Uso

|  |  |
| --- | --- |
| Identificação | UC01 – Solicitar serviço |
| Função | Este caso de uso descreve as etapas necessárias para o cliente solicitar um serviço. |
| Atores | Cliente, prestador. |
| Pré-condição | O cliente deve estar autenticado como cliente e o prestador deve estar autenticado como prestador. |
| Pós-condição | A lista de serviços será atualizada. |
| Fluxo principal – solicitar serviço | 1. O cliente seleciona a opção nova solicitação.  2. O sistema exibe um formulário de solicitações em branco.  3. O cliente preenche o formulário com as informações do serviço.  4. O sistema registra a solicitação.  5. O prestador seleciona a opção visualizar serviços.  6. O sistema exibe as áreas de atuação do prestador.  7. O prestador seleciona a área de atuação desejada.  8. O sistema exibe as solicitações.  9. O prestador aceita a solicitação.  10. O sistema informa o cliente. |
| Fluxo Alternativo – solicitação recusada | Após a etapa 8 do fluxo o principal:  9. O prestador recusa a solicitação. |
| Fluxo Exceção |  |

|  |  |
| --- | --- |
| Identificação | UC02 – Detalhar o serviço |
| Função | Este caso de uso descreve as etapas necessárias para o cliente detalhar um serviço. |
| Atores | Cliente, prestador. |
| Pré-condição | O cliente deve estar autenticado como cliente e o prestador deve estar autenticado como prestador. |
| Pós-condição |  |
| Fluxo principal – detalhar serviço | 1. O cliente seleciona o serviço desejado.  2. O sistema disponibiliza um chat entre cliente e prestador.  3. O cliente detalha o serviço iniciando uma conversa.  4. O sistema atualiza as informações do chat para o prestador.  5. O prestador analisa as informações e conversa com o cliente. |
| Fluxo Alternativo |  |
| Fluxo Exceção |  |

|  |  |
| --- | --- |
| Identificação | UC03 – Marcar visita |
| Função | Estabelecer uma comunicação entre Prestador e Cliente para definir a data da visita e marca-la |
| Atores | Prestador e Cliente |
| Pré-condição | O Prestador e o Cliente terem aceitado o serviço |
| Pós-condição | A data do serviço deve ser atualizada |
| Fluxo principal | 1. O sistema deve disponibilizar o chat para a conversa entre os atores, Prestador e Cliente.  2. O Cliente e o Prestador utilizam o chat para escolher a data.  3. O sistema deve disponibilizar uma opção para o Prestador escolher uma data.  4. O Prestador deve informar a data definida.  5. O sistema deve confirmar a data definida pelo Prestador.  6. O sistema deve confirmar a data definida com o Cliente.  7. O Cliente confirma a data.  8. O sistema deve salvar a data marcada para este serviço. |
| Fluxo Alternativo – Cliente não confirma a data | A partir do passo 6 do fluxo principal:  7. O sistema deve permitir o Cliente definir uma data  8. O Cliente deve informar a data escolhida  9. O sistema deve confirmar a data definida com o Cliente  10. O sistema deve confirmar a data definida com o Prestador  11. O sistema deve salvar a data marcada para este serviço |
| Fluxo Alternativo – O Cliente e o Prestador não utilizam o chat | 1. O sistema deve disponibilizar o chat para a conversa entre os atores, Prestador e Cliente.  2. O sistema deve disponibilizar uma opção para o Prestador escolher uma data.  3. O Prestador deve informar a data definida.  4. O sistema deve confirmar a data definida pelo Prestador.  5. O sistema deve confirmar a data definida com o Cliente.  6. O Cliente confirma a data.  7. O sistema deve salvar a data marcada para este serviço. |

|  |  |
| --- | --- |
| Identificação | UC04 – Efetuar Pagamento |
| Função | Pagar o serviço realizado |
| Atores | Cliente |
| Pré-condição | O serviço deve ter sido concluído |
| Pós-condição | Atualizar a condição atual do serviço |
| Fluxo principal | 1. O sistema deve mostrar os serviços ativos do Cliente.  2. O Cliente deve escolher o serviço que deseja pagar.  3. O sistema deve permitir o cliente marcar o serviço como finalizado.  4. O Cliente deve marcar o serviço como finalizado.  5. O sistema deve mostrar as opções de pagamento.  6. O Cliente deve escolher uma opção.  7. O sistema deve permitir o pagamento.  8. O Cliente deve pagar o serviço.  9. O sistema valida as informações do pagamento.  10. O sistema salva o serviço como pago e finalizado. |
| Fluxo Exceção – Pagamento recusado | A partir do passo 9: o sistema não valida as informações do pagamento.  10. O sistema deve permitir o Cliente escolher novamente a opção de pagamento.  11. O Cliente deve escolher uma opção.  12. O sistema deve permitir o pagamento.  13. O Cliente deve pagar o serviço.  14. O sistema salva o serviço como pago e finalizado. |

|  |  |
| --- | --- |
| Identificação | UC05 – Avaliar serviço |
| Função | Avaliar o serviço realizado |
| Atores | Cliente |
| Pré-condição | O serviço deve ter sido concluído |
| Pós-condição | Atualizar a avaliação média do Prestador |
| Fluxo principal | 1. O sistema deve mostrar os valores possíveis para a avaliação e disponibilizar um espaço para o comentário.  2. O Cliente deve escolher um valor de avaliação e escrever um comentário.  3. O sistema deve salvar a avaliação e o comentário relacionando com o Prestador do serviço. |
| Fluxo Alternativo – Sem comentário | 1. O sistema deve mostrar os valores possíveis para a avaliação e disponibilizar um espaço para o comentário.  2. O Cliente deve escolher um valor de avaliação e não escrever nenhum comentário.  3. O sistema deve salvar apenas a avaliação relacionando com o Prestador do serviço. |

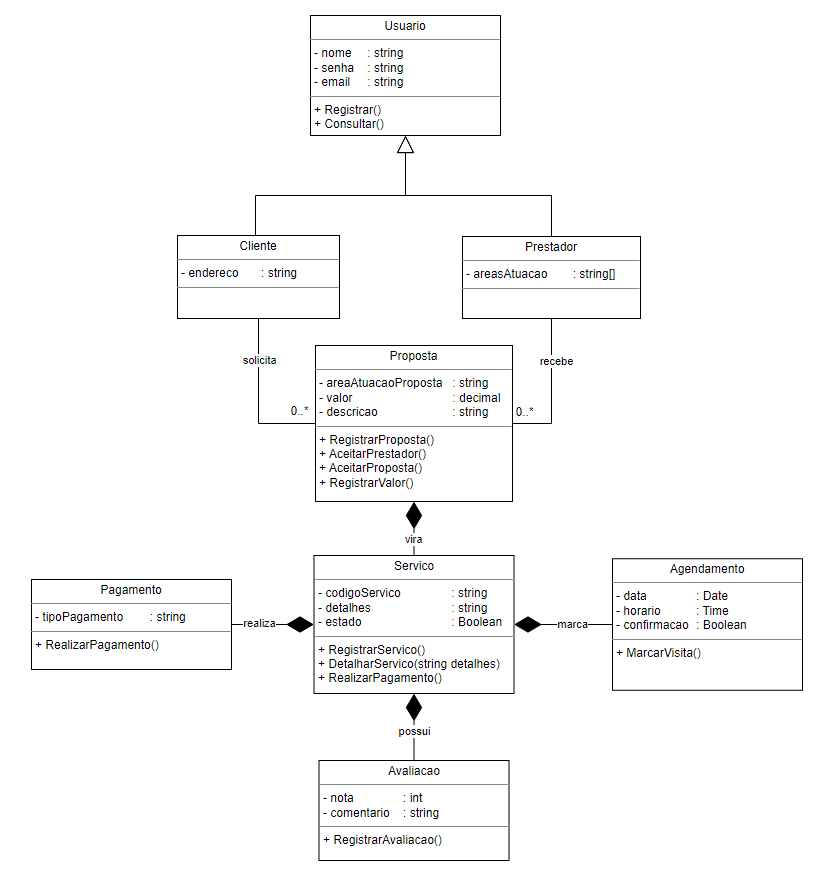
|  |  |
| --- | --- |
| Identificação | UC06 – Gerenciar conta cliente |
| Função | Este caso de uso descreve as etapas necessárias para o cliente registrar, visualizar, atualizar e excluir sua conta. |
| Atores | Cliente. |
| Pré-condição |  |
| Pós-condição | A lista de clientes será atualizada. |
| Fluxo principal – registrar cliente | 1. O cliente seleciona a opção fazer cadastro.  2. O sistema exibe um formulário de cadastro em branco.  3. O cliente preenche o formulário.  4. O sistema registra o cliente. |
| Fluxo Alternativo – atualizar conta do cliente | 1. O cliente se autentica como cliente.  2. O cliente seleciona o seu perfil.  3. O sistema exibe as informações do cliente.  4. O cliente seleciona a opção editar.  5. O sistema exibe um formulário com as informações já preenchidas.  6. O cliente altera as informações necessárias.  7. O sistema atualiza as informações do cliente. |
| Fluxo Alternativo – visualizar conta do cliente | 1. O cliente se autentica como cliente.  2. O cliente seleciona o seu perfil.  3. O sistema exibe as informações do cliente. |
| Fluxo Alternativo – excluir conta do cliente | 1. O cliente se autentica como cliente.  2. O cliente seleciona o seu perfil.  3. O sistema exibe as informações do cliente.  4. O cliente seleciona a opção excluir conta.  5. O sistema exibe uma mensagem de confirmação.  6. O cliente confirma a ação.  7. O sistema exclui a conta do cliente. |
| Fluxo Exceção |  |

|  |  |
| --- | --- |
| Identificação | UC07 – Gerenciar conta prestador |
| Função | Este caso de uso descreve as etapas necessárias para o prestador registrar, visualizar, atualizar e excluir sua conta. |
| Atores | Prestador. |
| Pré-condição |  |
| Pós-condição | A lista de prestadores será atualizada. |
| Fluxo principal – registrar prestador | 1. O prestador seleciona a opção fazer cadastro.  2. O sistema exibe um formulário de cadastro em branco.  3. O prestador preenche o formulário.  4. O sistema registra o prestador. |
| Fluxo Alternativo – atualizar conta do prestador | 1. O prestador se autentica como prestador.  2. O prestador seleciona o seu perfil.  3. O sistema exibe as informações do prestador.  4. O prestador seleciona a opção editar.  5. O sistema exibe um formulário com as informações já preenchidas.  6. O prestador altera as informações necessárias.  7. O sistema atualiza as informações do prestador. |
| Fluxo Alternativo – visualizar conta do prestador | 1. O prestador se autentica como prestador.  2. O prestador seleciona o seu perfil.  3. O sistema exibe as informações do prestador. |
| Fluxo Alternativo – excluir conta do prestador | 1. O prestador se autentica como prestador.  2. O prestador seleciona o seu perfil.  3. O sistema exibe as informações do prestador.  4. O prestador seleciona a opção excluir conta.  5. O sistema exibe uma mensagem de confirmação.  6. O prestador confirma a ação.  7. O sistema exclui a conta do prestador. |
| Fluxo Exceção |  |

|  |  |
| --- | --- |
| Identificação | UC08 – Aceitar prestador |
| Função | Este caso de uso descreve as etapas necessárias para o cliente aceitar o prestador para o serviço solicitado. |
| Atores | Cliente. |
| Pré-condição | O cliente deve ter realizado a solicitação de um serviço e um prestador deve ter aceito essa solicitação. |
| Pós-condição | O prestador do serviço será atualizado. |
| Fluxo principal – aceitar prestador | 1. O cliente seleciona a opção ver solicitações.  2. O sistema exibe as solicitações feitas pelo cliente.  3. O cliente escolhe a solicitação desejada.  4. O sistema exibe os prestadores que aceitaram a solicitação.  5. O cliente aceita o prestador desejado.  6. O sistema atualiza o prestador do serviço. |
| Fluxo Exceção |  |

## 4. Classes

## 4.1. Diagrama de Classes



## 4.2. Especificações

|  |  |
| --- | --- |
| Nome Classe | Usuario |
| Descrição | Esta classe define informações referentes aos usuários em geral. |
| Atributos | Seus atributos são:  - **nome** (nome do usuário) do tipo String.  - **senha** (senha do usuário) do tipo String.  - **email** (e-mail do usuário) do tipo String. |
| Métodos | - **Registrar()**: responsável por instanciar um novo usuário. |
| - **Consultar()**: consulta todas as informações de um usuário. |
| Associação | Associação de Herança entre as classes Usuario, Cliente e Prestador. |

|  |  |
| --- | --- |
| Nome Classe | Cliente |
| Descrição | Esta classe define informações referentes aos usuários que são clientes. |
| Atributos | Seus atributos são:  - **endereco** (endereço do cliente) do tipo String. |
| Métodos |  |
| Associação | Associação de Herança entre as classes Cliente e Usuario. E associação binária entre as classes Cliente e Proposta; isto significa que um cliente pode solicitar no mínimo nenhuma e no máximo muitas propostas, e uma proposta deve ser solicitada somente por um cliente. |
| Nome Classe | Prestador |
| Descrição | Esta classe define informações referentes aos usuários que são prestadores. |
| Atributos | Seus atributos são:  - **areasAtuacao** (áreas de atuação do prestador) do tipo String[], é uma lista de string que armazena as áreas em que um prestador atua. |
| Métodos |  |
| Associação | Associação de Herança entre as classes Prestador e Usuario. E associação binária entre as classes Prestador e Proposta; isto significa que um prestador pode receber no mínimo nenhuma e no máximo muitas propostas, e uma proposta deve ser recebida somente por um prestador. |

|  |  |
| --- | --- |
| Nome Classe | Proposta |
| Descrição | Esta classe define informações referentes às propostas. |
| Atributos | Seus atributos são:  - **areaAtuacaoProposta** (área de atuação do serviço proposto) do tipo String.  - **valor** (valor do serviço à ser feito) do tipo decimal.  - **descricao** (breve descrição com no máximo 200 caracteres do serviço proposto) do tipo String. |
| Métodos | - **RegistrarProposta()**: responsável por instanciar uma nova proposta. |
| - **AceitarPrestador()**: responsável pela aceitação do prestador por parte do cliente, gerando um serviço. |
| - **AceitarProposta()**: responsável pela aceitação da proposta por parte do prestador. |
| - **RegistrarValor()**: responsável por registrar o valor da proposta, definido pelo prestador. |
| Associação | Associação binária entre as classes Proposta e Cliente; isto significa que uma proposta deve ser solicitada somente por um cliente, e um cliente pode solicitar no mínimo nenhuma e no máximo muitas propostas.  Associação binária entre as classes Proposta e Prestador; isto significa que uma proposta deve ser recebida somente por um prestador, e um prestador pode receber no mínimo nenhuma e no máximo muitas propostas.  Associação binária de composição entre as classes Proposta (todo) e Servico (parte); isto significa que uma proposta se torna no mínimo nenhum e no máximo um serviço, e um serviço é o resultado de somente uma proposta. |

|  |  |
| --- | --- |
| Nome Classe | Servico |
| Descrição | Esta classe define informações referentes aos serviços. |
| Atributos | Seus atributos são:  - **codigoServico** (número de identificação do serviço) do tipo String.  - **detalhes** (descrição detalhada do serviço à ser feito) do tipo String.  - **estado** (estado atual do serviço) do tipo Boolean, que define se um serviço está em andamento (verdadeiro) ou já foi finalizado (falso). |
| Métodos | - **RegistrarServico()**: responsável por instanciar um novo serviço. |
| - **DetalharServico()**: responsável por registrar os detalhes do serviço, por parte do cliente. |
| Associação | Associação binária de composição entre as classes Servico (todo) e Agendamento (parte); isto significa que um serviço deve marcar somente um agendamento, e um agendamento deve ser marcado por somente um serviço.  Associação binária de composição entre as classes Servico (todo) e Avaliacao (parte); isto significa que um serviço possui somente uma avaliação, e uma avaliação deve ser somente de um serviço.  Associação binária de composição entre as classes Servico (parte) e Proposta (todo); isto significa que um serviço é o resultado de somente uma proposta, e uma proposta se torna no mínimo nenhum e no máximo um serviço.  Associação binária de composição entre as classes Servico (todo) e Pagamento (parte); isto significa que um serviço possui somente um pagamento, e um pagamento deve ser somente de um serviço. |

|  |  |
| --- | --- |
| Nome Classe | Pagamento |
| Descrição | Esta classe define informações referentes ao pagamento do serviço. |
| Atributos | Seus atributos são:  - **tipoPagamento** (tipo do pagamento) do tipo String. |
| Métodos | - **RealizarPagamento()**: responsável pelo pagamento do serviço por parte do cliente. |
| Associação | Associação binária de composição entre as classes Servico (todo) e Pagamento (parte); isto significa que um serviço possui somente um pagamento, e um pagamento deve ser somente de um serviço. |

|  |  |
| --- | --- |
| Nome Classe | Avaliacao |
| Descrição | Esta classe define informações referentes à avaliação do serviço. |
| Atributos | Seus atributos são:  - **nota** (nota do serviço) do tipo int.  - **comentario** (comentário sobre a execução do serviço) do tipo String. |
| Métodos | - **RegistrarAvaliacao()**: responsável por instanciar uma nova avaliação de um serviço. |
| Associação | Associação binária de composição entre as classes Servico (todo) e Avaliacao (parte); isto significa que um serviço possui somente uma avaliação, e uma avaliação deve ser somente de um serviço. |
| Nome Classe | Agendamento |
| Descrição | Esta classe define informações referentes ao agendamento do serviço. |
| Atributos | Seus atributos são:  - **data** (data do serviço) do tipo Date.  - **horario** (horário em que o serviço será realizado) do tipo Time.  - **confirmacao** (confirmação da data proposta pelo prestador, por parte do cliente) do tipo String. |
| Métodos | - **MarcarVisita()**: responsável por instanciar um novo agendamento de um serviço. |
| Associação | Associação binária de composição entre as classes Servico (todo) e Agendamento (parte); isto significa que um serviço deve marcar somente um agendamento, e um agendamento deve ser marcado por somente um serviço. |