UNIVERSIDADE ESTADUAL DE GOIÁS

SISTEMAS DE INFORMAÇÃO

MARCO ANTONIO BORGES NUNES

PROA – PROJETO DE APOIO A REGULARIZAÇÃO EM OPERAÇÕES AQUAVIÁRIAS

**MONOGRAFIA DE GRADUAÇÃO**

TRINDADE

2024

**MARCO ANTONIO BORGES NUNES**

DESENVOLVIMENTO DE SOFTWARE PARA GERAÇÃO E MANUTENÇÃO DE DOCUMENTOS UTILIZADOS EM SERVIÇOS DE ASSESORIA NAVAL

Monografia de Graduação a ser apresentada para a Coordenação do curso de Sistemas de Informação da Universidade Estadual de Goiás, como requisito para obtenção do título de “Bacharel em Sistemas de informação”.

**Orientador: Olegário Correa da Silva Neto**

TRINDADE

2024

**AGRADECIMENTOS**

**RESUMO**

**ABSTRACT**

SUMÁRIO

Sumário

[1. INTRODUÇÃO 9](#_Toc181382695)

[1.1 Objetivos Gerais 10](#_Toc181382696)

[1.2 Objetivos Específicos 10](#_Toc181382697)

[1.3 Justificativa 10](#_Toc181382698)

[1.4 Metodologia 11](#_Toc181382699)

[1.5 Estrutura do Trabalho 11](#_Toc181382700)

[2. REFERENCIAL TEÓRICO 12](#_Toc181382701)

[2.1 Gerenciamento de Requisitos 12](#_Toc181382702)

[2.2 Tecnologias utilizadas 13](#_Toc181382703)

[2.3 Padrões de Projeto 15](#_Toc181382704)

[3. MODELAGEM DO SOFTWARE 17](#_Toc181382705)

[3.1. Diagramas de Atividade 17](#_Toc181382706)

[3.2. Levantamento de Requisitos 20](#_Toc181382707)

[3.2.1. Requisitos Funcionais 20](#_Toc181382708)

[3.2.2. Requisitos Não Funcionais 25](#_Toc181382709)

[4. REFERÊNCIAS 32](#_Toc181382710)

Lista de Figuras

[Figura 1 - Diagrama de Atividade – Anexos Fonte: Autoria Própria 19](#_Toc181433518)

[Figura 2 - Diagrama de Atividade – Serviços Fonte: Autoria Própria 20](#_Toc181433519)

[Figura 3 – Caso de uso principal (CSU1, CSU2, CSU3) Fonte: Autoria própria 29](#_Toc181433520)

[Figura 4 – Caso de uso Emitir Anexos (CSU4) Fonte: Autoria própria 30](#_Toc181433521)

[Figura 5 – Caso de uso emitir anexos por Serviços (CSU5) – Serviços Pessoa Física Fonte: Autoria própria 31](#_Toc181433522)

[Figura 6 – Caso de uso emitir anexos por Serviços (CSU5) – Serviços Embarcação MA Fonte: Autoria própria 31](#_Toc181433523)

[Figura 7 - Caso de uso emitir anexos por Serviços (CSU5) – Serviços Embarcação ER Fonte: Autoria própria 32](#_Toc181433524)

[Figura 8 – MER – Modelo de Entidade Relacionamento Fonte: Autoria própria 33](#_Toc181433525)

[Figura 9 – Diagrama de Classe Fonte: Autoria própria 34](#_Toc181433526)

Lista de Tabelas

[Tabela 1 - Requisitos Funcionais do CSU1 – Manter Cliente Fonte: Autoria própria 21](#_Toc181382263)

[Tabela 2 – Requisitos Funcionais do CSU2 – Manter Embarcação Fonte: Autoria própria 22](#_Toc181382264)

[Tabela 3 – Requisitos Funcionais do CSU3 – Manter Empresa/Prestador Fonte: Autoria Própria 23](#_Toc181382265)

[Tabela 4 – Requisitos Funcionais CSU4 – Emitir Anexos Fonte: Autoria própria 24](#_Toc181382266)

[Tabela 5 – Requisitos Funcionais CSU5 – Emitir Anexos por Serviço Fonte: Autoria própria 25](#_Toc181382267)

[Tabela 6 – Requisitos de interface Homem – Computador Fonte: Autoria própria 26](#_Toc181382268)

[Tabela 7 – Requisitos de interface externa Fonte: Autoria própria 26](#_Toc181382269)

[Tabela 8 – Requisitos de plataforma de software Fonte: Autoria própria 27](#_Toc181382270)

[Tabela 9 – Requisitos de desempenho Fonte: Autoria própria 27](#_Toc181382271)

[Tabela 10 – Requisitos de segurança Fonte: Autoria própria 28](#_Toc181382272)

# INTRODUÇÃO

Com o avanço tecnológico temos visto uma rápida e crescente tendência de automação e informatização dos mais diversos meios. Amplos setores da sociedade avançam rápido com a tecnologia, fornecendo serviços inovadores para a população em geral, proporcionando assim uma significativa reorganização no modo de vida e nas expectativas das pessoas.

Contudo, há setores da sociedade que enfrentam uma miríade de problemas com essa rápida mudança nos paradigmas. No Brasil, setores governamentais e correlatos a muito buscam se adaptar a esse novo cenário, mas uma conjuntura de fatores como o alto nível de burocracia, a dificuldade de alteração de sistemas e processos legados, a intervenção de forças diversas e interesses políticos distintos geram uma prestação de serviço onerosa, burocrática, lenta e simplesmente incompatível com as demandas e expectativas da sociedade atual.

Um exemplo prático desse tema, que abordaremos no decorrer desse projeto, pode ser observado nos diversos serviços prestados pela Marinha do Brasil, onde ainda hoje são requeridos um conjunto de documentos impressos, assinados e com firma registrada para abertura de processos como habilitação ou regularização, até mesmo em processos de renovação onde todos os dados já existem junto a Marinha tal documentação ainda é exigida de forma física e presencial. Até certo tempo atrás sequer o preenchimento digital dos dados era aceito, forçando os usuários do serviço a preparar de próprio punho toda a documentação.

Dentre tantos desafios, como podemos utilizar dos meios de tecnologia para facilitar a vida de pessoas que muitas vezes buscam apenas uma forma rápida e eficiente de estar na legalidade?

OBJETIVOS  
  
A seguir serão apresentados os objetivos gerais e específicos do trabalho.

# Objetivos Gerais

Desenvolver um sistema de cadastro de usuários, embarcações e prestadores de serviço que possibilite a geração automatizada de documentos solicitados pela Marinha do Brasil para cada respectivo serviço prestado pelo órgão.

# Objetivos Específicos

* Analisar os requisitos legais e normativos estabelecidos pela Marinha do Brasil para a regularização de embarcações, identificando os documentos necessários e os procedimentos exigidos.
* Identificar as principais demandas e necessidades dos assessores náuticos no que diz respeito aos processos de cadastro de usuários e geração dos documentos para emissão de habilitação tipo Arrais amador e Motonauta.
* Projetar e desenvolver uma interface intuitiva e amigável para o sistema, com funcionalidade que permitam o cadastro eficiente de usuários e a geração automatizada de documentos conforme as exigências legais.
* Projetar e desenvolver uma base de dados robusta e segura que suporte o armazenamento e gerenciamento eficiente das informações dos usuários e demais dados necessários.
* Implementar recursos de segurança que permitam a validação automática dos dados inseridos, bem como a geração automática e precisa dos documentos a serem emitidos.

# Justificativa

A introdução apresentada contextualiza de maneira clara a problemática a ser abordada, com destaque para a lacuna entre a rapidez da evolução tecnológica e a lenta e morosa adaptação dos setores governamentais a essa evolução. O exemplo prático fornecido ilustra bem a incompatibilidade entre a demanda da sociedade por praticidade e eficiência em contraste com a burocracia adotada pelos órgãos competentes, em nosso caso específico a Marinha do Brasil. Tal cenário evidencia a necessidade de soluções que integrem a tecnologia aos processos administrativos, proporcionando uma experiência ágil e simplificada aos usuários.

Portanto, este projeto de T.C. propõe a investigação e desenvolvimento de um sistema capaz de permitir o cadastro e manutenção das informações dos usuários, de suas embarcações e demais dados correlatos e a partir desses dados preparar de forma fácil a documentação exigida pela Marinha do Brasil, de acordo com o serviço a ser solicitado. A relevância deste projeto reside na sua capacidade de promover uma significativa melhoria na eficiência dos serviços prestados, contribuindo para a modernização e desburocratização de um setor essencial para a navegação comercial e turística no país.

# Metodologia

As soluções propostas para resolução dos problemas apresentados foram obtidas através do processo de levantamento de requisitos, que consistiu no primeiro momento em ações junto ao potencial usuário a fim de se obter um melhor entendimento das regras de negócio envolvidas. Para isso foram realizadas entrevistas, onde um questionário foi elaborado a partir das informações colhidas.

Aos requisitos estão associados os principais problemas do desenvolvimento de software. Requisitos que não refletem as reais necessidades dos usuários, incompletos e/ou inconsistentes, mudanças em requisitos já acordados e a dificuldade para conseguir um entendimento comum entre usuários e desenvolvedores são as principais dificuldades relatadas, provocando retrabalho, atrasos no cronograma, custos ultrapassados e a insatisfação dos clientes e usuários de software (Blaschek, 2002)

Uma vez em posse dessas informações, foram iniciadas as demais etapas do processo de desenvolvimento, sendo construída a modelagem inicial a partir dos requisitos levantados anteriormente e na sequência iniciou-se as etapas de desenvolvimento e validação.

O presente trabalho pode ser classificado como um estudo de caso, se observado do ponto de vista técnico, ou como uma pesquisa exploratória aplicada, se baseado no ponto de vista da natureza de sua concepção. A metodologia utilizada consistiu na coleta e análise de informações sobre uma organização, com objetivo de identificar possíveis melhorias a serem realizadas por meio de um software a ser desenvolvido.

Com isso, buscou-se a aplicação prática dos conhecimentos obtidos em sala para obtenção de uma solução, que resultou no desenvolvimento de protótipos para um sistema para gerenciamento dos dados de clientes e geração automatizada de documentos a partir desses dados.

# Estrutura do Trabalho

PENDENTE. FAZER AO FINAL

# REFERENCIAL TEÓRICO

Neste capítulo será demonstrado o referencial teórico utilizado neste projeto, demonstrando as características e conceitos dos fundamentos de especificação do software bem como as tecnologias utilizadas no projeto.

# Gerenciamento de Requisitos

Segundo Sommerville (2011): "Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento.

Adicionalmente, podemos definir o Gerenciamento de Requisitos como um conjunto de atividades que ajuda a equipe do projeto a identificar, controlar e rastrear requisitos e mudanças nos requisitos em qualquer momento à medida que o projeto prossegue (Wiegers e Beatty, 2013). Quando a organização não dispõe deste processo formalmente definido e amplamente divulgado, os desenvolvedores elaboram as especificações de requisitos de uma forma empírica, executando atividades não padronizadas e definidas individualmente. Se isto ocorre, a qualidade da especificação dependerá exclusivamente da experiência e formação das pessoas, havendo assim uma elevada probabilidade de ocorrerem conflitos e retrabalho (BLASCHEK, 2014)

O uso de ferramentas que ajudam na gestão de requisitos de software traz diversos benefícios, criando um ambiente que favorece a colaboração e a responsabilidade. Organizações que adotam essas ferramentas e mantêm uma documentação precisa e detalhada dos requisitos conseguem dividir tarefas de forma mais eficaz, estimar prazos com maior exatidão e, assim, ganham tempo para gerenciar riscos e realizar manutenções necessárias devido às novas alterações. No gerenciamento de requisitos de software, existem várias técnicas estabelecidas para a coleta desses requisitos. Um levantamento eficaz é aquele que resulta em uma definição clara do projeto, na conclusão bem-sucedida do mesmo, na apresentação das informações essenciais para um diagnóstico preciso e na proposição de soluções inteligentes. Como afirma Moraes (2014):

“As técnicas de levantamento de requisitos têm por objetivo superar as dificuldades relativas a esta fase. Todas as técnicas possuem um conceito próprio e suas respectivas vantagens e desvantagens, que podem ser utilizadas em conjunto pelo analista.”

Estabelecidos os conceitos utilizados nos requisitos e em seu gerenciamento, segue agora algumas das técnicas que podem ser empregadas no levantamento de requisitos.

Entrevistas: Uma entrevista pode ser definida como uma reunião entre um Stakeholder e o analista de requisitos que tem como objetivo auxiliar ambas as partes na melhor compreensão dos problemas envolvido, nas necessidades do usuário e no esclarecimento dos objetivos envolvidos no processo. (SOMMERVILLE, 2011)

Questionário: Utilizado como complemento a outras formas de levantamento de requisitos, os questionários são especialmente importantes em análises que abrangem uma grande quantidade de usuários, pois permite coletar informações de forma eficiente e objetiva. São especialmente úteis quando não há viabilidade de entrevistar todos os potenciais usuários envolvidos no projeto. (WIEGERS & BEATTY, 2013)

Cenários: Composto por um conjunto de descrições do possível funcionamento do sistema em determinadas situações, os Cenários incluem uma sequência de passos descrevendo a interação entre o usuário e o sistema, bem como o contexto que envolve essa interação. Seu objetivo é que os stakeholders consigam visualizar como o sistema se comportará em diferentes situações. (SOMMERVILLE, 2011)

Prototipagem: Considerada uma técnica avançada, a construção de protótipos consiste em uma técnica para obtenção de informações específicas sobre os requisitos através da demonstração de um modelo relativamente funcional do sistema antes de realmente construí-lo por completo. ((PRESSMAN & MAXIM, 2016)

# Tecnologias utilizadas

**Java SpringBoot**

O Spring Boot tem se tornado uma plataforma fundamental para a criação de aplicações Java modernas, especialmente em cenários de microsserviços e sistemas RESTful. De acordo com Turnquist (2017), ele torna o processo de configuração e desenvolvimento de aplicações Java muito mais simples, oferecendo uma abordagem baseada em convenções, que elimina a necessidade de especificar manualmente um grande número de parâmetros e dependências que eram comuns anteriormente.

Essa ferramenta permite o desenvolvimento de aplicações autônomas que podem ser executadas sem a exigência de configurar um servidor de aplicação separado, como o Tomcat, pois já vem integrado. Sua popularidade no desenvolvimento de APIs RESTful se deve à facilidade na integração com bibliotecas e à sólida estrutura para injeção de dependências, além do suporte nativo ao JSON.

**Angular**

Angular é um framework voltado para o desenvolvimento front-end, bastante popular na criação de aplicações web modernas e eficientes. Criado e mantido pelo Google, ele utiliza uma arquitetura orientada a componentes, o que torna a modularização, o reuso de código e a manutenção das aplicações mais simples. Um dos grandes diferenciais do Angular é oferecer uma solução abrangente para o desenvolvimento de aplicações, incluindo recursos para gerenciamento de rotas, formulários reativos, injeção de dependências e interação com APIs externas (Freeman, 2019).

**PostGre**

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional (SGBD) de código aberto, conhecido por sua solidez, adaptabilidade e conformidade com os padrões SQL. Conforme indicado pelo PostgreSQL Global Development Group (2023), ele foi desenvolvido para processar grandes volumes de dados e é utilizado por empresas de variados setores devido à sua habilidade em realizar operações complexas, como transações ACID e consultas sofisticadas. Ademais, o PostgreSQL permite o uso de extensões, possibilitando a incorporação de funcionalidades personalizadas para atender a diversas demandas das aplicações (Documentação do PostgreSQL, 2023).

**Insomnia**

O Insomnia é uma ferramenta de código aberto bastante popular para realizar testes e desenvolver APIs REST e GraphQL. Com uma interface clara e intuitiva, essa aplicação permite que os desenvolvedores façam requisições HTTP, testem endpoints e monitorem as respostas, simplificando a integração entre front-end e back-end nos projetos de software. Ademais, o Insomnia disponibiliza suporte para autenticação, gerenciamento de cookies e variáveis de ambiente, o que contribui para a automação e controle em diversos ambientes de desenvolvimento e produção (Kanjilal, 2022).

**GitHub**

O GitHub é uma plataforma destinada ao armazenamento de código-fonte e ao gerenciamento de versões de forma distribuída, sendo amplamente utilizada para a colaboração em projetos de desenvolvimento de software. Baseado no sistema Git, ele possibilita que equipes trabalhem simultaneamente em diferentes seções de um projeto, facilitando o controle das alterações e a integração contínua do código. Além de funcionar como um repositório, o GitHub oferece várias funcionalidades, como rastreamento de problemas (issues), revisão de código e integração com serviços externos, tornando-se uma ferramenta indispensável para o desenvolvimento de software contemporâneo (Chacon & Straub, 2014). Segundo Loeliger e McCullough (2012), o GitHub melhora o fluxo de trabalho permitindo que os desenvolvedores façam pull requests, o que promove revisões antes da inclusão de novas funcionalidades no código principal.

**Adobe** **Acrobat**  
Adobe Acrobat é uma ferramenta amplamente utilizada para criar, visualizar, editar e assinar arquivos Portable Document Format (PDF). Segundo a própria Adobe, o Acrobat “permite converter documentos em arquivos PDF, editar, revisar e compartilhar, além de proteger documentos com criptografia” (Adobe, 2023). Essa versatilidade faz do Acrobat a escolha de empresas e indivíduos para gerenciar documentos digitais com segurança e eficiência.

# Padrões de Projeto

**MVC (Model-View-Controller)**

O padrão Model-View-Controller (MVC) é uma arquitetura de software bastante utilizada no desenvolvimento de aplicações web. De acordo com Gamma et al. (1994, p. 4), "o MVC divide o modelo do domínio, a apresentação e as ações baseadas na entrada do usuário em três classes distintas".

Reenskaug e Coplien (2009) comentam que o MVC foi inicialmente criado para aplicações desktop, mas com o passar do tempo passou por adaptações, tornando-se um dos pilares da arquitetura web. Os autores mencionam que "a finalidade do MVC é criar uma distinção clara entre a lógica de negócios e a lógica de apresentação em uma aplicação".

Em relação a uma aplicação web contemporânea, o MVC pode ser estruturado da seguinte maneira:

🡪 Model: Este componente representa os dados e a lógica de negócios. No caso de um sistema que utiliza Spring Boot e PostgreSQL, Walls (2022, p. 35) ressalta que "o Spring Data JPA facilita a camada de persistência, permitindo que os desenvolvedores se concentrem na definição das interfaces de repositório, enquanto o framework gera implementações em tempo real".

🡪 View: A camada de Visualização é responsável por apresentar os dados ao usuário. Freeman (2022, p. 12) esclarece que "o Angular, como um framework de front-end, opera principalmente na camada de View do padrão MVC, oferecendo uma estrutura sólida para desenvolver interfaces de usuário dinâmicas e responsivas".

🡪 Controller: A função do Controlador é gerenciar a transferência de dados entre o Modelo e a Visualização. De acordo com Johnson et al. (2022, p. 89), "No Spring Boot, os controladores costumam ser classes marcadas com @RestController, que tratam das solicitações HTTP e coordenam a interação entre o cliente (Angular) e o servidor".

Seshadri e Green (2021, p. 15) descrevem como a integração desses componentes em um sistema web contemporâneo acontece: "A junção do Angular no front-end com o Spring Boot no back-end proporciona uma clara divisão de responsabilidades, onde o Angular cuida da interface do usuário enquanto o Spring Boot se ocupa da lógica de negócios e da persistência dos dados".

# MODELAGEM DO SOFTWARE

A modelagem de software desempenha um papel fundamental no desenvolvimento de sistemas, pois proporciona uma visão clara e organizada da solução proposta antes de sua execução. Neste capítulo, iremos abordar os diferentes aspectos da modelagem do PROA.

Essa modelagem inclui desde a representação dos requisitos funcionais por meio de diagramas de casos de uso até a detalhada estruturação das classes e entidades do sistema. Empregamos a Linguagem de Modelagem Unificada (UML) para elaborar representações visuais que capturam a essência do sistema, suas interações e os fluxos de dados envolvidos.

## 3.1. Diagramas de Atividade

O Diagrama de Atividade é uma ferramenta essencial na modelagem de software, oferecendo uma representação visual dos fluxos de trabalho e processos dentro do sistema PROA. Este diagrama permite ilustrar a sequência de ações, decisões e fluxos que ocorrem durante a execução de uma funcionalidade específica.

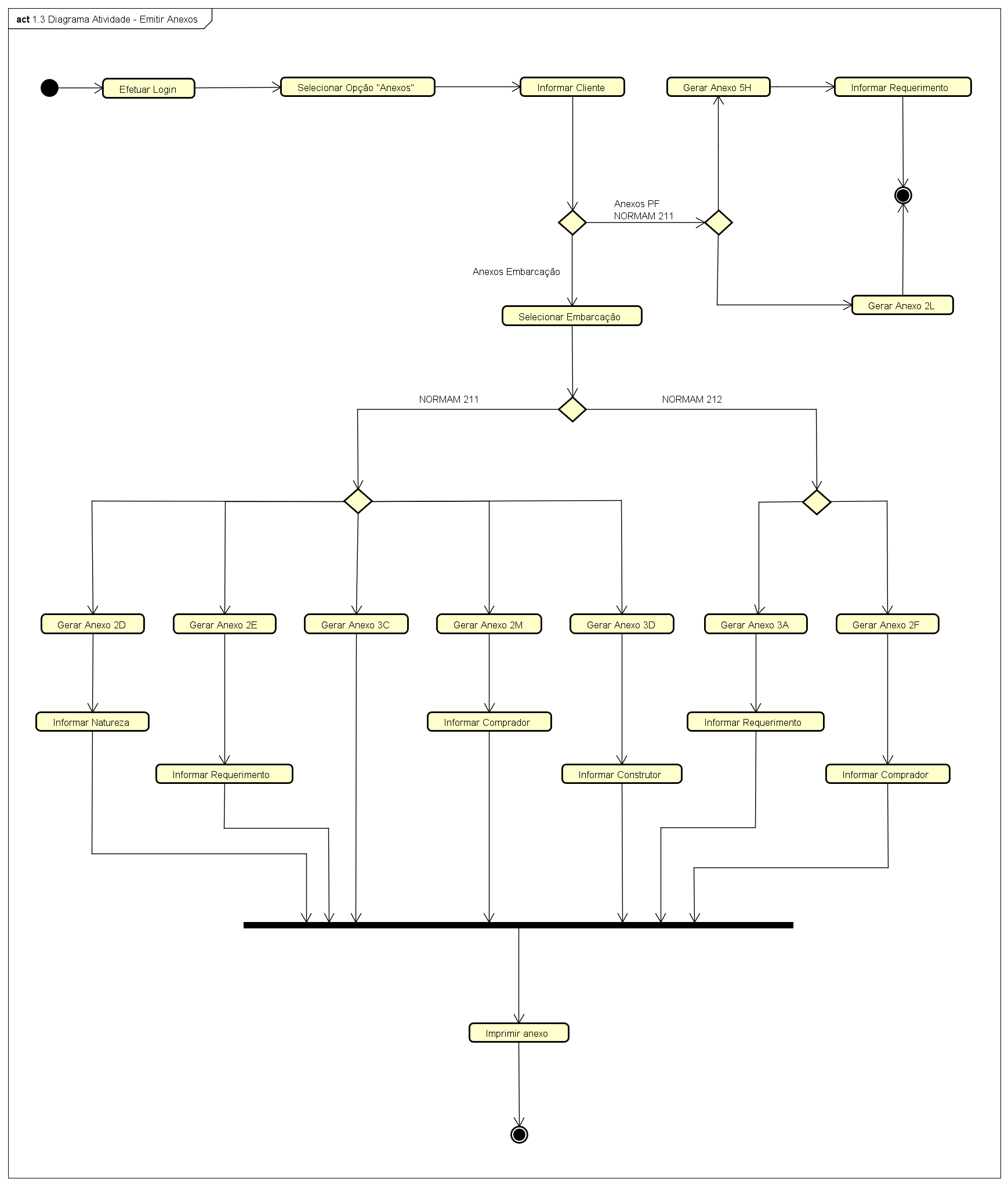


Figura 1 - Diagrama de Atividade – Anexos  
Fonte: Autoria Própria

O diagrama de atividade acima representa o fluxo de atividade a ser realizado para a emissão dos anexos a partir dos dados previamente cadastrados do cliente e da embarcação. Uma vez informado o cliente será possível emitir então os anexos relacionados somente a pessoa física. Se selecionada uma embarcação será então possível realizar a emissão dos demais anexos. Como essa emissão é detalhada, alguns anexos requerem informações adicionais para sua emissão. Essas informações serão utilizadas para compor corretamente o anexo solicitado de acordo com as NORMAM.

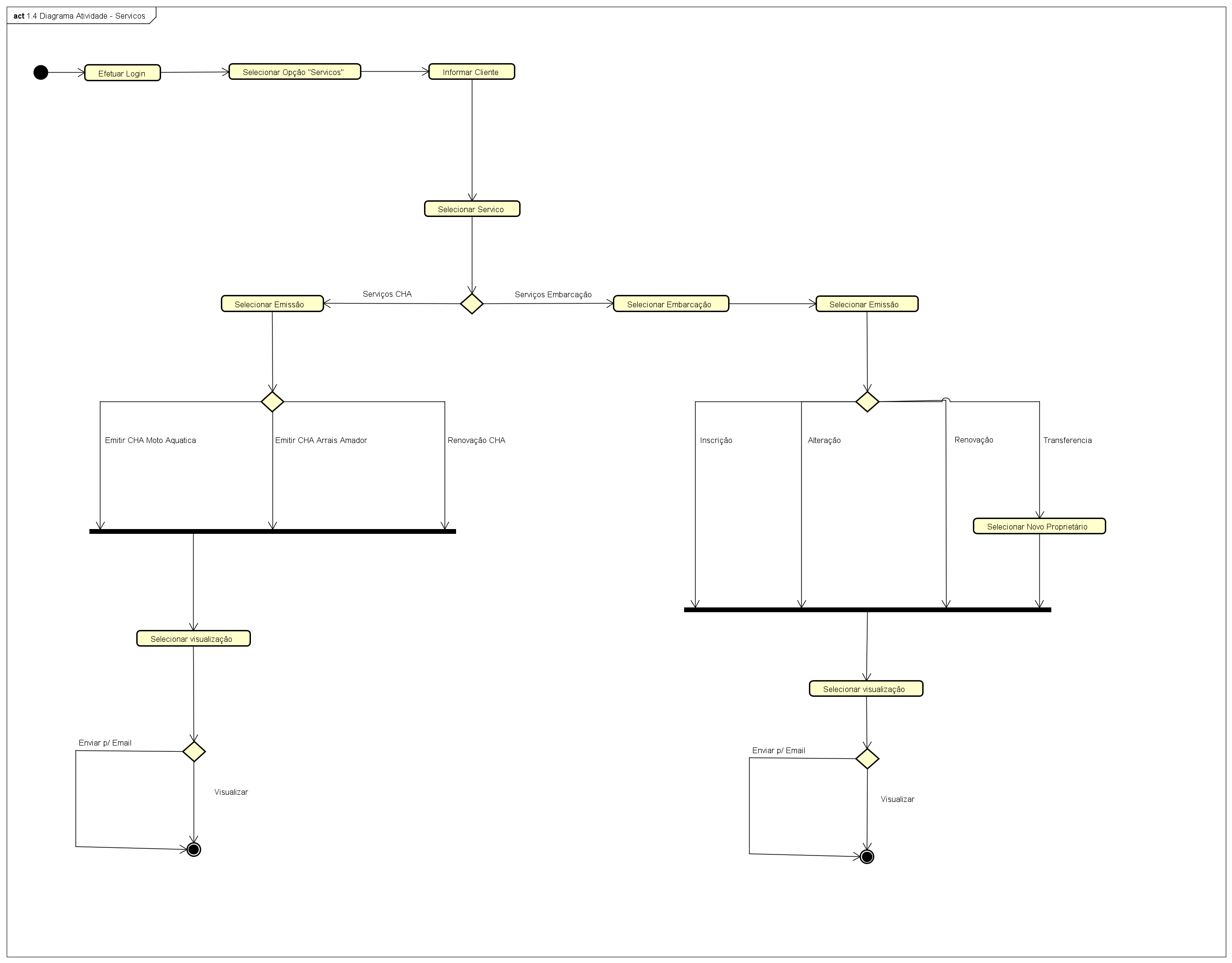


Figura 2 - Diagrama de Atividade – Serviços  
Fonte: Autoria Própria

Diferente do diagrama anterior, o diagrama de atividade – Serviços demonstra os serviços disponíveis para o usuário. O objetivo do serviço nesse contexto é agrupar toda a documentação necessária para um determinado objetivo, como a Procuração, Anexos relacionados ao serviço procurado e comprovante de endereço. A seleção do serviço também tem como objetivo fornecer uma emissão objetiva de um conjunto de anexos sem a necessidade de confirmação do usuário. Deve ser possível emitir a documentação gerada em tela ou realizar o respectivo envio por e-mail ao cliente solicitante.

## 3.2. Levantamento de Requisitos

### 3.2.1. Requisitos Funcionais

CSU1- Manter Cliente

Atores: Prestador de serviço  
Categoria: Primário

Descrição:

Este caso de uso ocorre quando um cliente solicita um determinado serviço para o prestador. Para cada tipo de serviço há um conjunto de dados que devem ser preenchidos, de forma que se deve então conduzir um cadastro completo do cliente em sistema, para que os dados estejam disponíveis quando forem necessários em etapas futuras.

Requisitos Funcionais:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref.** | **Descrição** | **Categoria** | **Prioridade** |
| RFUN 1.1 | Permitir aos Prestadores a manutenção completa do cadastro de seus clientes (Inclusão, alteração e exclusão) | Evidente | Alta |
| RFUN 1.2 | Validar os dados inseridos no cadastro de acordo com os padrões legais. | Evidente | Alta |
| RFUN 1.3 | Registrar Log com os dados do cadastro (Data, Hora, operador, etc) | Evidente | Alta |

Tabela 1 - Requisitos Funcionais do CSU1 – Manter Cliente  
Fonte: Autoria própria

CSU2- Manter Embarcação

Atores: Prestador de serviço  
Categoria: Primário

Descrição:

Este caso de uso ocorre quando um cliente solicita um serviço relacionado a embarcações para o prestador. Para cada tipo de serviço há um conjunto de dados que devem ser preenchidos, de forma que se deve então conduzir um cadastro completo do Cliente e da Embarcação em sistema, para que os dados estejam disponíveis quando forem necessários em etapas futuras.

Requisitos Funcionais:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref.** | **Descrição** | **Categoria** | **Prioridade** |
| RFUN 2.1 | Permitir aos Prestadores a manutenção completa do cadastro de Embarcações (Inclusão, alteração, exclusão e listagem) | Evidente | Alta |
| RFUN 2.2 | Caso a Embarcação já esteja motorizada, deve ser realizado o cadastro do motor. | Evidente | Alta |
| RFUN 2.3 | As informações referentes a motor podem ser alteradas por todos com acesso a embarcação | Evidente | Média |
| RFUN 2.4 | Validar os dados inseridos no cadastro de acordo com os padrões legais. | Evidente | Alta |
| RFUN 2.5 | Registrar Log com os dados do cadastro (Data, Hora, operador, etc) | Evidente | Média |

Tabela 2 – Requisitos Funcionais do CSU2 – Manter Embarcação  
Fonte: Autoria própria

CSU3- Manter Empresa/Prestador

Atores: Prestador de serviço  
Categoria: Primário

Descrição:

Este caso de uso ocorre quando um prestador deseja iniciar a utilização do sistema. Para isso, ele deverá realizar o cadastro da sua empresa com os dados pertinentes, realizar o próprio cadastro como cliente e o cadastro como cliente dos demais funcionários e sinalizar que estes atuam pela empresa.

Requisitos Funcionais:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref.** | **Descrição** | **Categoria** | **Prioridade** |
| RFUN 3.1 | Permitir o cadastro da empresa com seus respectivos dados. | Evidente | Alta |
| RFUN 3.2 | Permitir vincular colaboradores cadastrados como clientes para que componham os dados dos anexos onde forem pertinentes. | Evidente | Alta |
| RFUN 3.3 | Garantir que os clientes cadastrados por prestadores da empresa estejam vinculados a ela. | Evidente | Alta |
| RFUN 3.4 | Não permitir que usuários de uma empresa a visualização de dados de clientes de outras empresas. |  |  |

Tabela 3 – Requisitos Funcionais do CSU3 – Manter Empresa/Prestador  
Fonte: Autoria Própria

CSU4- Emitir Anexos

Atores: Prestador de serviço  
Categoria: Primário

Descrição:

Este caso de uso ocorre quando Cliente e Embarcação estão devidamente cadastradas, podendo então emitir os anexos necessários a partir de uma seleção individual. O cliente ou prestador deve acessar a respectiva página e selecionar qual anexo é necessário e as informações do cliente ou embarcação para consulta. Feito isso o sistema irá disponibilizar uma nova página contendo o anexo preenchido com os dados para impressão.

Requisitos Funcionais:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref.** | **Descrição** | **Categoria** | **Prioridade** |
| RFUN4.1 | Cada prestador somente pode visualizar dados de clientes vinculados a sua própria empresa | Evidente | Alta |
| RFUN 4.2 | O prestador somente pode emitir anexos de seus respectivos clientes | Evidente | Alta |
| RFUN 4.3 | Anexos emitidos não devem ser salvos e nem permanecer acessíveis por outros que não o emissor. | Evidente | Média |
| RFUN 4.4 | Registrar Emissor, Data e hora de cada anexo emissão. | Evidente | Baixa |
| RFUN 4.5 | Os anexos devem ser formatados seguindo estritamente os modelos disponíveis na regulamentação. | Evidente | Alta |
| RFUN 4.6 | Os anexos que requerem reconhecimento de firma na assinatura devem ser sinalizados. | Evidente | Média |

Tabela 4 – Requisitos Funcionais CSU4 – Emitir Anexos  
Fonte: Autoria própria

CSU5- Emitir anexos por Serviço

Atores: Prestador de serviço  
Categoria: Primário

Descrição:

Este caso de uso ocorre quando Cliente e Embarcação estão devidamente cadastradas, podendo então emitir os anexos necessários para atendimento a um determinado serviço. O prestador deve acessar a respectiva página e selecionar o cliente, a embarcação e qual serviço será prestado. A partir disso o sistema deverá disponibilizar a opção de envio dos respectivos anexos daquele serviço para o e-mail do cliente ou realizar uma impressão.

Requisitos Funcionais:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref.** | **Descrição** | **Categoria** | **Prioridade** |
| RFUN5.1 | Gerar os anexos necessários para um determinado serviço | Evidente | Alta |
| RFUN 5.2 | Permitir que os anexos gerados possam ser imediatamente impressos. | Evidente | Alta |
| RFUN 5.3 | Permitir que os anexos gerados possam ser encaminhados para o email cadastrado do cliente. | Evidente | Média |
| RFUN 5.4 | Anexos emitidos não devem ser salvos e nem permanecer acessíveis por outros que não o emissor. | Evidente | Média |
| RFUN 5.5 | Registrar Emissor, Data e hora de cada anexo emissão. | Evidente | Baixa |
| RFUN 5.6 | Os anexos que requerem reconhecimento de firma na assinatura devem ser sinalizados. | Evidente | Média |
| RFUN 5.7 | Os anexos devem ser formatados seguindo os modelos disponíveis na Regulamentação. | Evidente | Alta |

Tabela 5 – Requisitos Funcionais CSU5 – Emitir Anexos por Serviço  
Fonte: Autoria própria

3.2.2. Requisitos Não Funcionais

Este item descreve os requisitos e restrições não funcionais que o sistema deve atender para garantir desempenho, segurança, usabilidade, e outros aspectos importantes que não estão diretamente relacionados às funcionalidades específicas, mas que impactam a experiência geral e a eficácia do sistema.

Requisitos/Restrições de Interface Homem-Computador (RHIC)

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Descrição** | **Casos de Uso** |
| RIHC1 | A interface do sistema deve ser intuitiva e fácil de usar, com uma curva de aprendizado mínima, garantindo que novos usuários possam operar o sistema com eficácia após um breve treinamento. | Todos |
| RIHC2 | O sistema deve ser compatível com os principais navegadores (Google Chrome, Mozilla Firefox, Microsoft Edge e Safari), garantindo uma experiência de uso consistente em todos eles. | Todos |

Tabela 6 – Requisitos de interface Homem – Computador  
Fonte: Autoria própria

Requisitos e Restrições de Interface Externa (RIEX)

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Descrição** | **Casos de Uso** |
| RIEX1 | O sistema deverá utilizar a plataforma de autenticação FireBase para permitir o acesso. | CSU1 |
| RIEX2 | Nas telas de cadastro deverá ser disponibilizado a pesquisa de CEP através da API pública “Busca CEP” para facilitar o preenchimento dos campos de endereço. | CSU1, CSU2 |

Tabela 7 – Requisitos de interface externa  
Fonte: Autoria própria

Requisitos/Restrições de Plataforma de Software (RPSW)

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Descrição** | **Casos de Uso** |
| RPSW1 | O sistema deve ser desenvolvido com uma arquitetura modular, utilizando os recursos do Spring Boot para o backend e Angular para o frontend. Isso permite que componentes individuais sejam atualizados ou substituídos sem afetar outras partes do sistema, facilitando a manutenção e futuras expansões. | Todos |
| RPSW2 | O sistema deve ser projetado para suportar testes automatizados. | Todos |
| RPSW3 | A implementação deve garantir que a aplicação seja compátivel com o banco de dados PostgreSQL. | Todos |

Tabela 8 – Requisitos de plataforma de software  
Fonte: Autoria própria

Requisitos/Restrições de Desempenho (RDES)

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Descrição** | **Casos de Uso** |
| RDES1 | A geração de anexos PDF, incluindo a coleta de dados do banco de dados e a renderização dos documentos, deve ser concluída em no máximo 10 segundos, garantindo uma experiência fluida para o usuário. | Todos |
| RDES2 | O sistema deve suportar, sem degradação perceptível de desempenho, ao menos 30 conexões simultâneas ao servidor Spring Boot, garantindo que múltiplos usuários possam operar o sistema ao mesmo tempo. | Todos |
| RDES3 | O PostgreSQL deve ser otimizado para suportar um grande volume de dados, incluindo o uso de índices, e otimização de consultas para garantir o desempenho à medida que o volume de dados cresce. | Todos |

Tabela 9 – Requisitos de desempenho  
Fonte: Autoria própria

Requisitos/Restrições de Segurança de Acesso (RSEG)

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Descrição** | **Casos de Uso** |
| RSEG1 | O sistema utilizará o Firebase Authentication para gerenciar a autenticação dos usuários no front-end Angular. Todos os usuários devem se autenticar via Firebase antes de acessar qualquer funcionalidade protegida do sistema. | Todos |
| RSEG2 | Após a autenticação via Firebase, um token JWT (JSON Web Token) será gerado e utilizado para manter a sessão do usuário. Esse token será enviado em todas as requisições ao backend Spring Boot para validação da identidade do usuário. | Todos |
| RSEG3 | O sistema deve implementar mecanismos automáticos de renovação de token, garantindo que as sessões permaneçam ativas enquanto o usuário estiver utilizando o sistema. Tokens expirados devem exigir que o usuário faça login novamente. | Todos |
| RSEG4 | O sistema deve garantir que todos os usuários forneçam consentimento explícito para a coleta e processamento de seus dados pessoais, conforme exigido pela Lei Geral de Proteção de Dados (LGPD). | Todos |

Tabela 10 – Requisitos de segurança  
Fonte: Autoria própria

**3.3. Diagramas de Casos de uso**

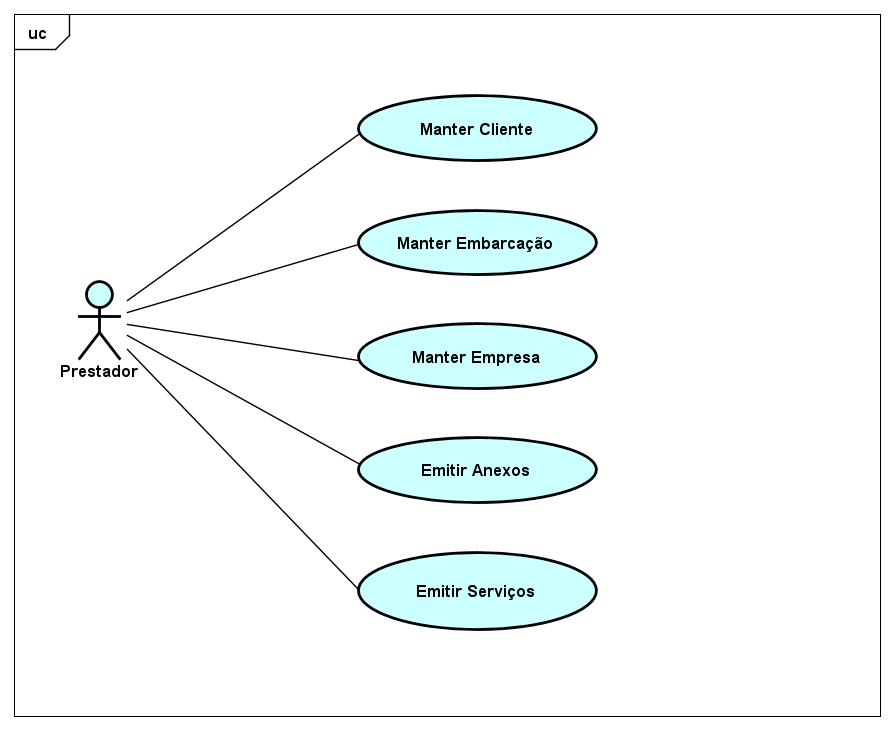


Figura 3 – Caso de uso principal (CSU1 – Manter Cliente, CSU2 – Manter Embarcação, CSU3 – Manter Empresa)  
Fonte: Autoria própria

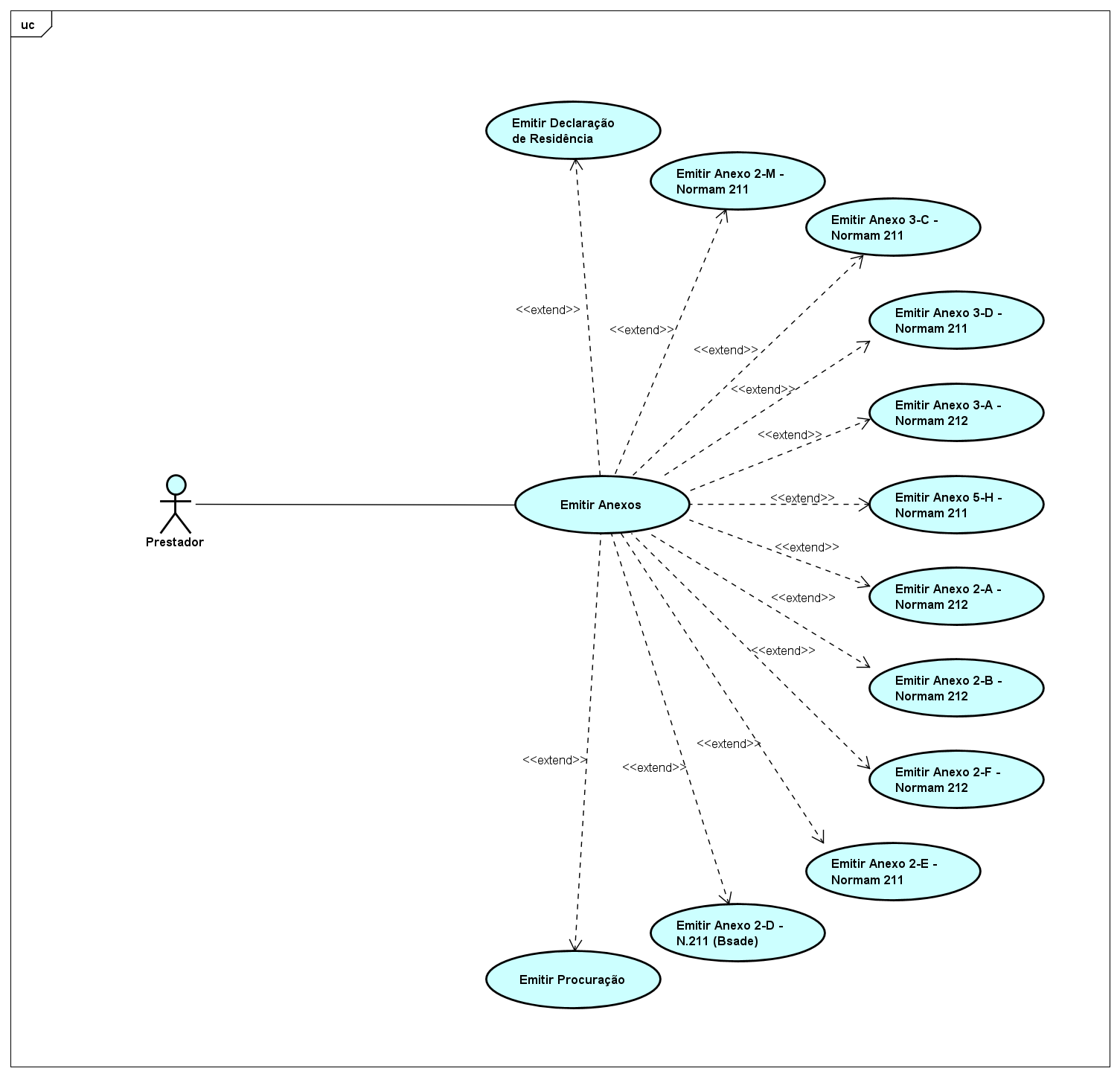


Figura 4 – Caso de uso Emitir Anexos (CSU4)  
Fonte: Autoria própria

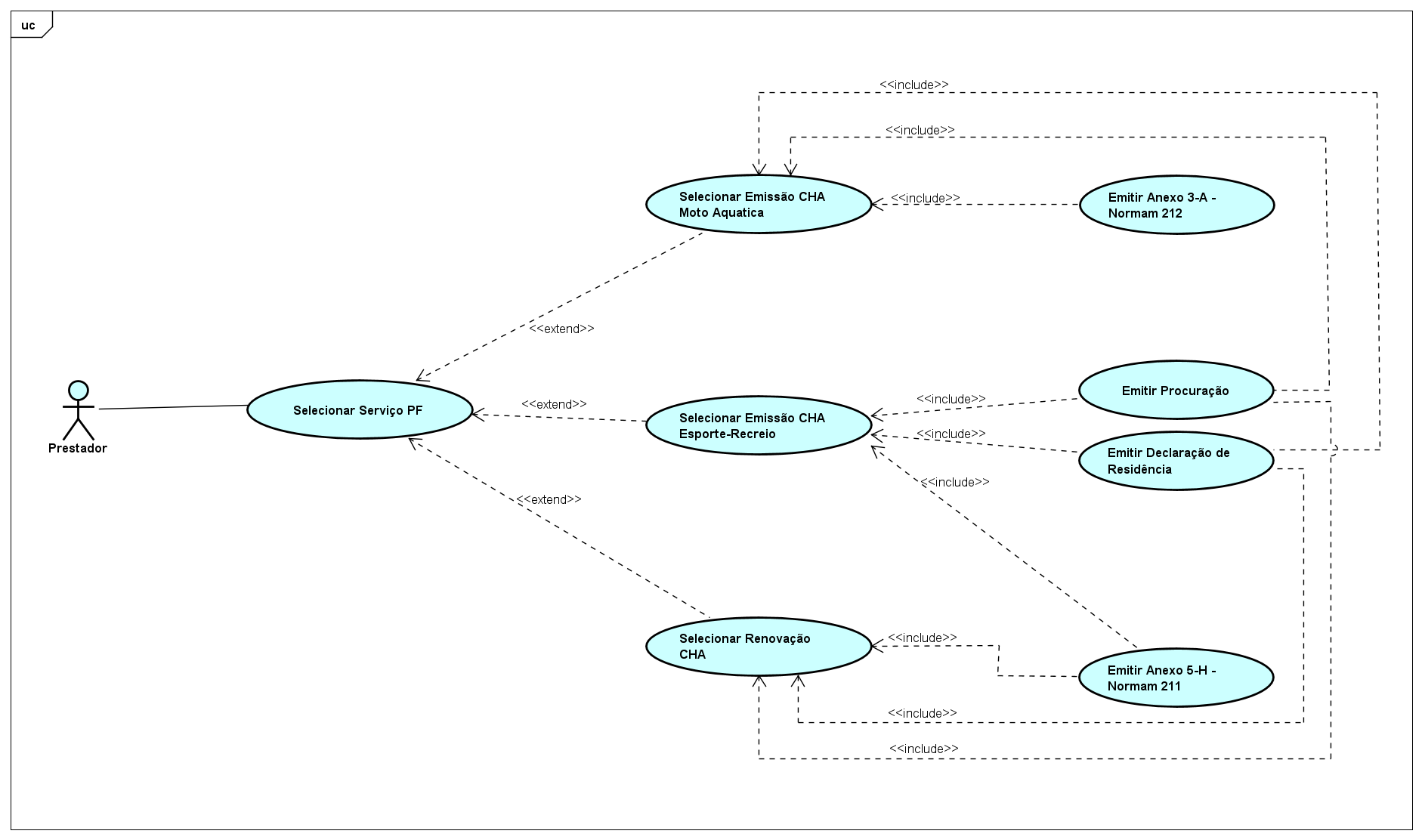


Figura 5 – Caso de uso emitir anexos por Serviços (CSU5) – Serviços Pessoa Física  
Fonte: Autoria própria

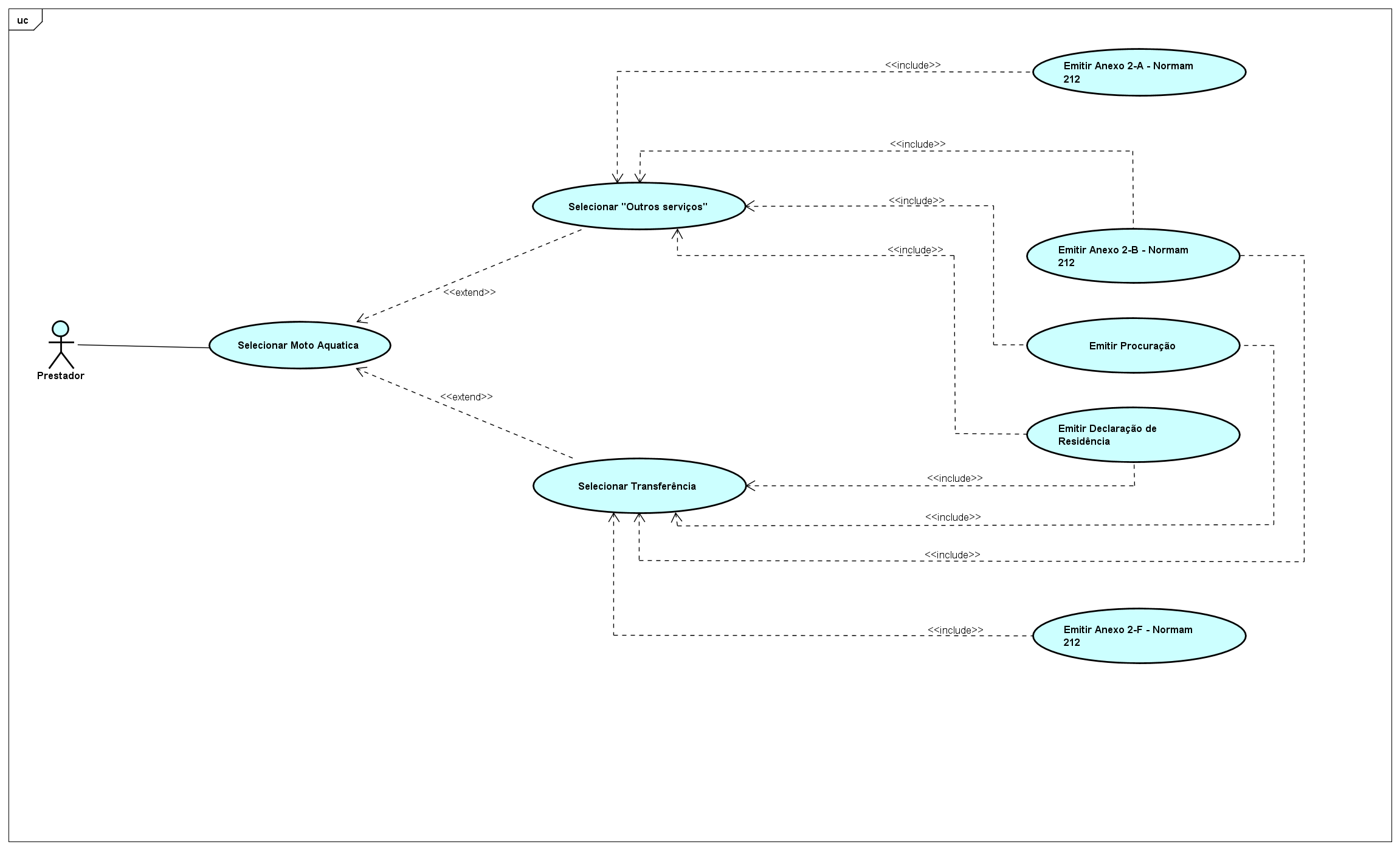


Figura 6 – Caso de uso emitir anexos por Serviços (CSU5) – Serviços Embarcação MA  
Fonte: Autoria própria

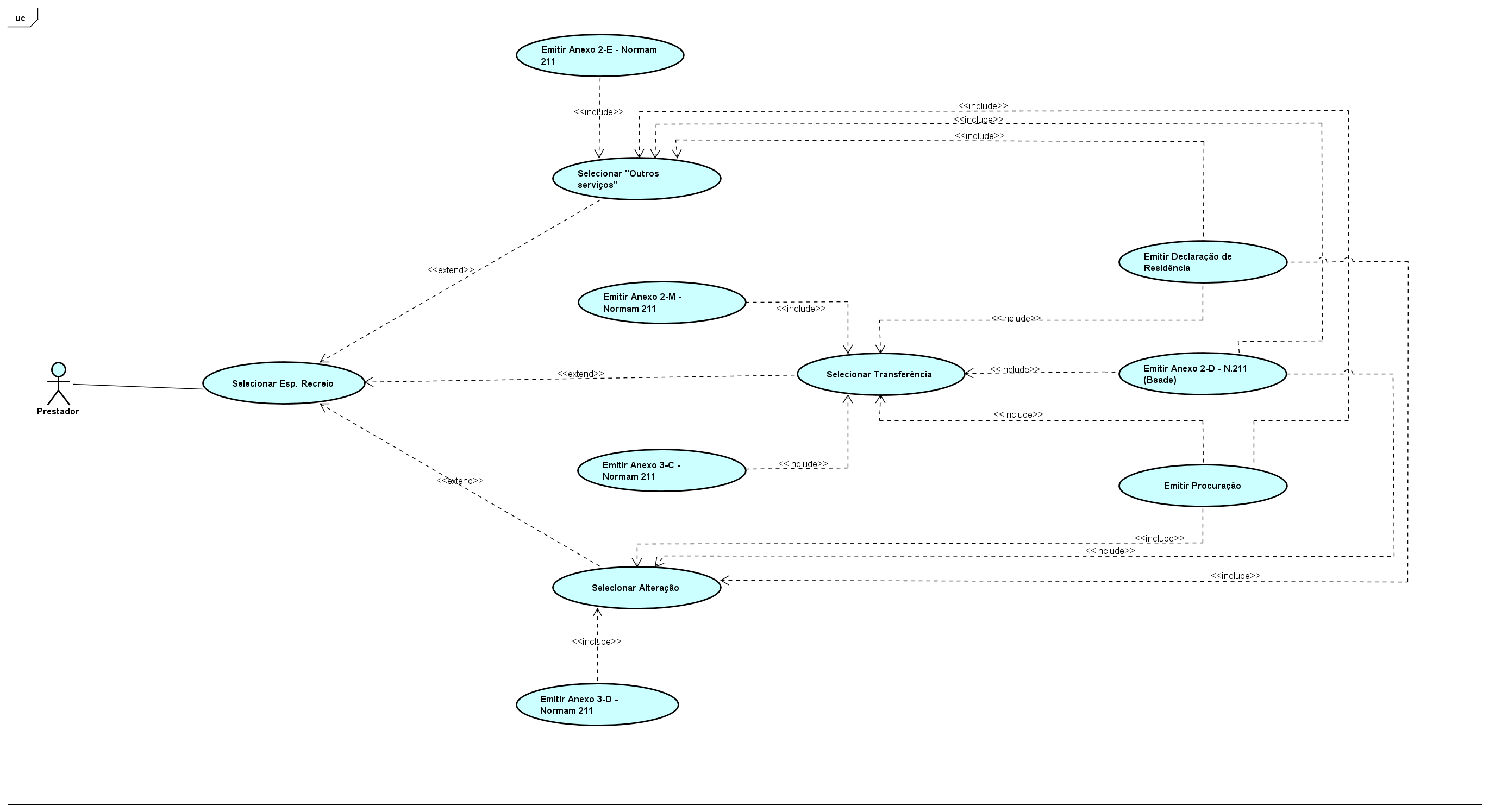


Figura 7 - Caso de uso emitir anexos por Serviços (CSU5) – Serviços Embarcação ER  
Fonte: Autoria própria

## 3.4. Modelo de Entidade-Relacionamento

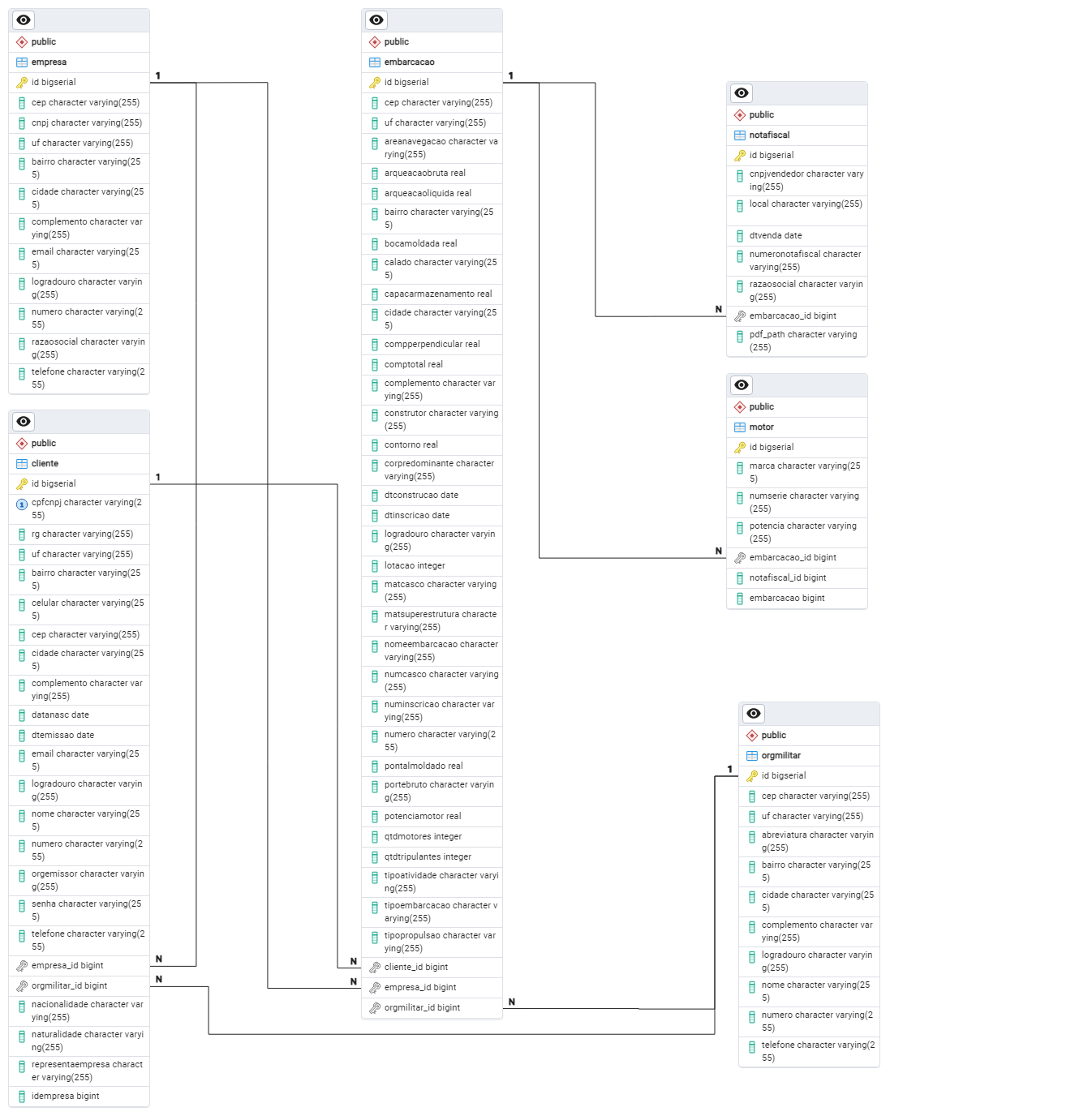


Figura 8 – MER – Modelo de Entidade Relacionamento  
Fonte: Autoria própria

## 3.5. Diagrama de Classe

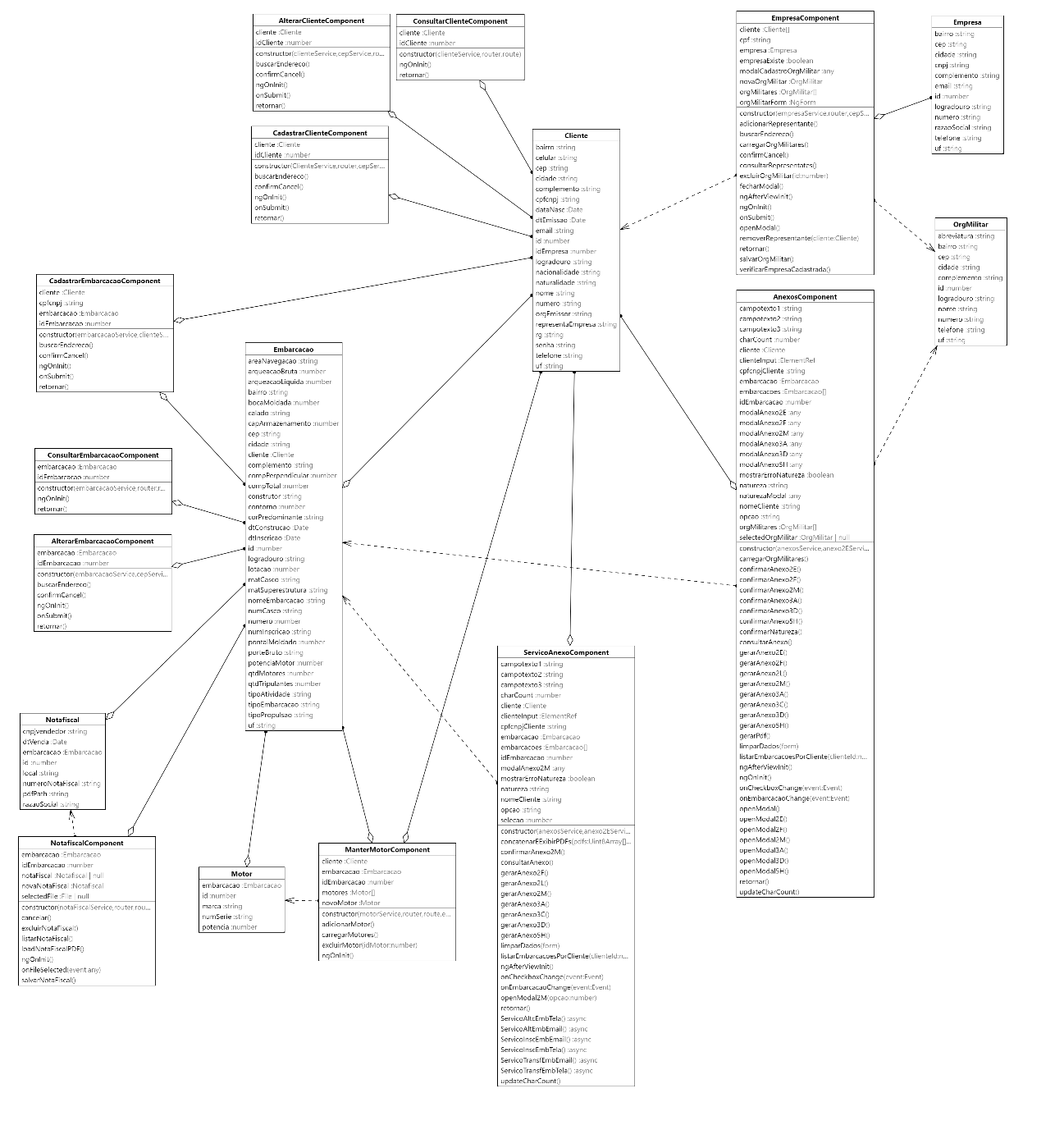


Figura 9 – Diagrama de Classe  
Fonte: Autoria própria

# REFERÊNCIAS

BLASCHEK, José Roberto. [O papel dos requisitos no insucesso dos projetos de software. 2002](https://wandersonwiller.wordpress.com/wp-content/uploads/2012/04/gerc3aancia-de-requisitos-o-principal-problema-dos-projetos-de-sw.pdf). Dissertação (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2002.

SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.  
  
WIEGERS, Karl; BEATTY, Joy. Software Requirements. 3. ed. Redmond: Microsoft Press, 2013.

MORAES, Janaina. **Técnicas para levantamento de requisitos**. DevMedia, 2014. Disponível em: <https://www.devmedia.com.br/tecnicas-para-levantamento-de-requisitos/9151>. Acesso em: 24 set. 2024.

(PRESSMAN, R. S.; MAXIM, B. R. Engenharia de Software: Uma Abordagem Profissional. 8ª ed. Porto Alegre: AMGH, 2016, p. 45)

[Spring Framework Documentation](https://docs.spring.io/spring-framework/docs/6.0.0/reference/pdf/spring-framework.pdf)

Turnquist, G. (2017). *Learning Spring Boot 2.0: Simplify the development of lightning-fast applications based on microservices and reactive programming*. 2nd ed. Packt Publishing.

Freeman, A. (2019). *Pro Angular 8*. Apress.

[Main Page/pt - PostgreSQL wiki](https://wiki.postgresql.org/wiki/Main_Page/pt#Funcionalidades)

Kanjilal, Joydip. TESTAR suas APIs REST usando o cliente REST do Insomnia. Code Magazine. Disponível em: [Testar suas APIs REST usando o cliente REST do Insomnia (codemag.com)](https://www.codemag.com/Article/2107051/Test-Your-REST-APIs-Using-Insomnia-REST-Client). Acesso em: 27 set. 2024.

Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.

Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development*. O'Reilly Media.

Adobe. (2023). **Adobe Acrobat Features**. Recuperado de <https://www.adobe.com/acrobat>.

GAMMA, E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. ([Design Patterns Elements of Reusable Object-Oriented Software (javier8a.com)](https://www.javier8a.com/itc/bd1/articulo.pdf))

REENSKAUG, T.; COPLIEN, J. O. The DCI Architecture: A New Vision of Object-Oriented Programming. 2009. Disponível em: <http://www.artima.com/articles/dci_vision.html>. Acesso em: 27 set. 2024.

WALLS, C. Spring in Action. 6th ed. Manning Publications, 2022.

FREEMAN, A. Pro Angular 9: Build Powerful and Dynamic Web Apps. Apress, 2022.

JOHNSON, R. et al. Spring Boot in Action. 2nd ed. Manning Publications, 2022.

SESHADRI, S.; GREEN, B. Angular Development with TypeScript. Manning Publications, 2021.