Uma imagem com texto

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

Uma imagem com arte, Gráficos, clipart, design

Descrição gerada automaticamente

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Relatório de Projeto**

Bruno Rafael Mendes Oliveira – a15566

Ana Margarida Maia Pinto – a23548

Diogo Carvalho Pinheiro – a24016

Diogo Mário Sá Fernandes – a24017

**Janeiro de 2025**

Esta página foi deixada em branco propositadamente.

Uma imagem com texto

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Relatório de Projeto**

***MedStock* - Aplicação de Farmácias Hospitalares**

**Unidades Curriculares**

Engenharia De *Software*

Gestão de Sistemas de Informação

Integração de Sistemas Clínicos

**Nome dos Alunos**

Bruno Rafael Mendes Oliveira

Ana Margarida Maia Pinto

Diogo Carvalho Pinheiro

Diogo Mário Sá Fernandes

**Docentes das Unidades Curriculares:**

Profª. Margarida Portela

Profª. Patrícia Leite

Profº. João Pedro Silva

**Janeiro de 2025**

Esta página foi deixada em branco propositadamente.

**Resumo**

Este relatório detalha a elaboração de um projeto na unidade curricular de Engenharia de Software, centrado no desenvolvimento de uma aplicação *Web* destinado à gestão de uma farmácia hospitalar. O projeto foi conduzido com o objetivo de integrar conhecimentos teóricos com a prática de desenvolvimento de software, pretendendo melhorar o armazenamento dos consumíveis e a adesão aos mesmos. (incompleto)

**Abstract**

(quando tiver o resumo completamente feito)

**Índice**

[Índice de Figuras 9](#_Toc187250363)

[Índice de Tabelas 10](#_Toc187250364)

[Lista de siglas e acrónimos 11](#_Toc187250365)

[1. Introdução 12](#_Toc187250366)

[1.1. Enquadramento 12](#_Toc187250367)

[1.2. Estrutura do Documento 12](#_Toc187250368)

[2. Instigadores do Projeto 13](#_Toc187250369)

[2.1. Problema / Contexto 13](#_Toc187250370)

[2.2. Objetivos / Benefícios 13](#_Toc187250371)

[2.3. Identificação dos Intervenientes 14](#_Toc187250372)

[2.3.1. Roda de Intervenientes 15](#_Toc187250373)

[2.3.2. Intervenientes Internos 15](#_Toc187250374)

[2.3.3. Intervenientes Externos 16](#_Toc187250375)

[2.4. Matriz de Interesse 17](#_Toc187250376)

[2.5. Utilizadores do Sistema 18](#_Toc187250377)

[3. Análise de Requisitos 19](#_Toc187250378)

[3.1. Requisitos Funcionais - RF 19](#_Toc187250379)

[3.2. Requisitos Não Funcionais – *RNF* 24](#_Toc187250380)

[*3.3.* Casos de Uso *MedStock* 26](#_Toc187250381)

[3.4. Diagrama Entidade Relação 29](#_Toc187250382)

[3.5. Diagrama de Contexto 33](#_Toc187250383)

[3.6. Diagrama *Business Process Modeling Notation* 34](#_Toc187250384)

[3.7. Diagrama de Estados 38](#_Toc187250385)

[4. Implementação 40](#_Toc187250386)

[4.1. Metodologia 41](#_Toc187250387)

[4.2. Arquitetura 42](#_Toc187250388)

[4.3. Tecnologias 43](#_Toc187250389)

[4.4. Integrações 44](#_Toc187250390)

[4.4.1. *FireBase* 44](#_Toc187250391)

[4.4.2. *Gmail* 45](#_Toc187250392)

[4.4.3. *MedReader* 46](#_Toc187250393)

[4.4.4. *MedOcorrências* 47](#_Toc187250394)

[4.4.5. *MedSupply* 48](#_Toc187250395)

[4.5. API 49](#_Toc187250396)

[4.5.1. Estrutura 49](#_Toc187250397)

[4.5.2. Servidor *Render* 49](#_Toc187250398)

[4.5.3. Documentação 49](#_Toc187250399)

[4.5.4. Padronização desenvolvida 50](#_Toc187250400)

[4.6. Aplicação 52](#_Toc187250401)

[4.6.1. *MedStock* 52](#_Toc187250402)

[4.6.1.1. Administrador 53](#_Toc187250403)

[4.6.1.2. Requerentes 56](#_Toc187250404)

[4.6.1.3. Gestor de Ala Hospitalar 56](#_Toc187250405)

[4.6.1.4. Farmacêutico 56](#_Toc187250406)

[4.6.1.5. Algoritmo de Gestão *de stock*s 57](#_Toc187250407)

[4.6.2. *MedReader* 59](#_Toc187250408)

[4.6.3. *MedOcorrencias* 64](#_Toc187250409)

[4.6.4. *MedSupply* 67](#_Toc187250410)

[5. Testes 67](#_Toc187250411)

[6. Repositório Projeto 67](#_Toc187250412)

[7. Trabalho Futuro 67](#_Toc187250413)

[8. Conclusão 67](#_Toc187250414)

[9. Bibliografia 69](#_Toc187250415)

# Índice de Figuras

[Figura 1 - Roda dos Intervenientes 15](#_Toc187250449)

[Figura 2 - Matriz de Interesse 17](#_Toc187250450)

[Figura 3 - Diagrama de Casos de Uso 28](#_Toc187250451)

[Figura 4 - Diagrama Entidade Relação 32](#_Toc187250452)

[Figura 5 - Diagrama de Contexto 33](#_Toc187250453)

[Figura 6 - Diagrama BPMN 37](#_Toc187250454)

[Figura 7 – Diagrama de Estados 38](#_Toc187250455)

[Figura 8 - Arquitetura do Sistema 42](#_Toc187250456)

[Figura 9 - Esquema Integração com *FireBase* 45](#_Toc187250457)

[Figura 10 - Esquema Integração com *Gmail* 45](#_Toc187250458)

[Figura 11 - Esquema Integração com *MedReader* 46](#_Toc187250459)

[Figura 12 - Esquema Integração com *MedOcorrências* 47](#_Toc187250460)

[Figura 13 - Esquema Integração com *MedSupply* 48](#_Toc187250461)

[Figura 14 - Resposta da *API* do método GET – Sucesso 50](#_Toc187250462)

[Figura 15 - Resposta da *API* do método GET – Erro 51](#_Toc187250463)

[Figura 16 - 53](#_Toc187250464)

[Figura 17 - 54](#_Toc187250465)

[Figura 18 - 54](#_Toc187250466)

[Figura 19 - 55](#_Toc187250467)

[Figura 20 - 55](#_Toc187250468)

[Figura 21 - 56](#_Toc187250469)

[Figura 22 - *MedReader* - Login 59](#_Toc187250470)

[Figura 23 - Lista Requerimentos - Não Urgentes 60](#_Toc187250471)

[Figura 24 - Lista de Requerimentos - Urgentes 60](#_Toc187250472)

[Figura 25 - Consumíveis Requerimento Selecionado 61](#_Toc187250473)

[Figura 26 - Leitura de Consumíveis 62](#_Toc187250474)

[Figura 27 - Atualização Estado Requerimento - *MedStock* 63](#_Toc187250475)

[Figura - Página de Introdução 64](#_Toc187250476)

[Figura - Login 65](#_Toc187250477)

[Figura - Registo de uma ocorrência 65](#_Toc187250478)

[Figura - Envio da ocorrência 66](#_Toc187250479)

[Figura - Visualização da ocorrência - *MedStock* 66](#_Toc187250480)

# Índice de Tabelas

[Tabela 1 - Requisitos Funcionais 22](#_Toc187250481)

[Tabela 2 - Requisitos Funcionais - Prioridade 23](#_Toc187250482)

[Tabela 3 - Requisitos Não Funcionais 25](#_Toc187250483)

# Lista de siglas e acrónimos

* *API* Application Programming Interface
* BPMN Business Process Model and Notation
* CPU Central Processing Unit
* DER Diagrama Entidade Relação
* ER Entidade Relação
* HTTP Hypertext Transfer Protocol
* ID Identificador
* JSON JavaScript Object Notation
* POO Programação Orientada a Objetos
* POS Programação Orientada a Serviços
* REST Representational State Transfer
* SOA Service-Oriented Architecture
* UC Unidade Curricular
* UI User Interface
* UML Unified Modeling Language
* UX User Experience

# Introdução

## Enquadramento

O *MedStock* surge como resposta à necessidade crescente de uma gestão de medicamentos mais eficiente e um acompanhamento rigoroso do armazenamento dos mesmos. Além disso, com o aumento da complexidade do controlo *do stock* e, consequentemente, a rutura do mesmo, tornou-se fundamental desenvolver uma ferramenta que ofereça suporte eficaz e confiável.

## Estrutura do Documento

(fazer quando estiver finalizado o relatório)

# Instigadores do Projeto

Durante este capítulo serão especificados os instigadores do projeto, que motivam a realização deste projeto, assim como os objetivos e a identificação dos intervenientes e os utilizadores do sistema.

## Problema / Contexto

A necessidade de desenvolver este sistema surgiu como resposta às dificuldades enfrentadas no modelo de gestão atual. Atualmente, as informações sobre os *stock*s encontram-se dispersas, os consumíveis são registados em formato de papel. Além disso, os processos de comunicação e gestão dos pedidos de consumíveis são realizados manualmente, o que resulta em erros, atrasos e perda de eficiência.

Todo este cenário motivou a idealização de um sistema que visa digitalizar este processo, centralizar informações e melhorar a comunicação entre os requerentes e a farmácia, promovendo uma gestão mais eficiente dos recursos hospitalares.

## Objetivos / Benefícios

O sistema proposto tem como principal objetivo a digitalização dos processos e a otimização da comunicação entre a farmácia hospitalar e os diferentes requerentes. Para além deste, existem outros objetivos principais, nomeadamente:

* **Centralizar informações *do stock*:** Desenvolver uma base de dados que permita à farmácia hospitalar consultar em tempo real os níveis *do stock* disponíveis.
* **Digitalizar processos de comunicação e pedidos:** Substituir os registos em papel e as comunicações telefónicas por um sistema digital que permita a realização de requisições e o envio de notificações de forma automatizada.
* **Aumentar a eficiência operacional:** Automatizar o processamento de pedidos e implementar mecanismos para a reposição de consumíveis.
* **Reduzir riscos de falhas e erros:** Minimizar as perdas de informação, atrasos e erros durante o fluxo de trabalho interno da farmácia hospitalar e nas interações com os requerimentos.
* **Promover interoperabilidade:** Facilitar a comunicação e a troca de informações entre farmácias hospitalares e fornecedores externos, assegurando a reposição mais rápida dos *stocks*.
* **Implementar validação dupla:** Introduzir mecanismos de validação intermédia durante a preparação dos pedidos e uma validação final pelo requerente, assegurando que os consumíveis entregues correspondem integralmente às requisições efetuadas.
* **Desenvolver um algoritmo de gestão *do stock***: Implementar um algoritmo capaz de monitorizar os níveis do *stock*, identificar consumíveis abaixo do nível *do stock* mínimo, redistribuir recursos entre requerimentos e priorizar requisições urgentes.

## Identificação dos Intervenientes

A identificação dos intervenientes é uma etapa do projeto necessária para compreender quais as partes envolvidas no sistema e os diferentes níveis de interesse e contribuição. Este processo permite reconhecer quem são os participantes e entidades relacionados com o funcionamento do sistema.

Os intervenientes identificados são:

* **Farmacêuticos**
* **Gestores Hospitalares**
* **Profissionais de Saúde (Enfermeiros, Médicos, Secretários Clínicos e Assistentes)**
* **Fornecedores**
* **Clínicas e Centros de Saúde**
* **Reguladores e Órgãos de Saúde**
* **Utentes**

### Roda de Intervenientes

A roda dos intervenientes, apresentada na Figura 1, divide os intervenientes em dois grupos, os intervenientes internos e externos. A definição destes dois grupos, auxilia na definição de estratégias garantindo, assim, que as necessidades e expectativas de todos os envolvidos sejam atendidas e alinhadas com os objetivos traçados para o projeto.

Uma imagem com texto, captura de ecrã, Tipo de letra, círculo

Descrição gerada automaticamente

Figura 1 - Roda dos Intervenientes

### Intervenientes Internos

Os intervenientes internos são aqueles que desempenham papéis diretos no funcionamento e na utilização do sistema. Estes estão envolvidos diretamente na gestão, operação e tomada de decisões relacionadas aos processos suportados pelo sistema.

Abaixo estão descritos os principais intervenientes internos:

* **Farmacêuticos:** Responsáveis pela gestão do *stock*, processamento de pedidos, preparação de consumíveis e comunicação com fornecedores.
* **Gestor Responsável:** Responsáveis por tomar decisões relacionadas ao nível de requerimentos de consumíveis.
* **Profissionais de Saúde (Enfermeiros, Médicos, Secretários Clínicos e Assistentes):** Utilizam o sistema para realizar pedidos de consumíveis, garantindo a continuidade dos tratamentos e procedimentos clínicos.

### Intervenientes Externos

Os intervenientes externos são aqueles que estão fora da organização e que, embora não desempenhem papéis diretos no funcionamento do sistema, são afetados de forma indireta pelas suas decisões e resultados.

Abaixo estão descritos principais intervenientes externos:

* **Fornecedores:** Entidades responsáveis por fornecer os consumíveis necessários ao funcionamento da farmácia hospitalar, assegurando a disponibilidade contínua dos consumíveis.
* **Clínicas e Centros de Saúde:** Potenciais utilizadores do sistema para gerir os seus próprios *stocks*.
* **Reguladores e Órgãos de Saúde:** Entidades responsáveis por garantir a conformidade do sistema com as normas e regulamentações, assegurando a legalidade e qualidade dos processos e operações realizadas.
* **Utentes**: Os utentes são intervenientes externos, uma vez que a eficácia do sistema reflete na qualidade do tratamento que recebem.

## Matriz de Interesse

A matriz de poder e interesse apresentada na Figura 2 é uma ferramenta de análise que categoriza os intervenientes do sistema com base no seu nível de interesse e poder de influência que possuem sobre o desenvolvimento e funcionamento do sistema. Esta abordagem permite identificar as prioridades de cada grupo de intervenientes e orientar estratégias para garantir que os seus interesses e necessidades sejam atendidos [12].

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 2 - Matriz de Interesse

* **Alto Poder, Alto Interesse:** Este grupo é composto por administradores e gestores responsáveis, tem um interesse direto e significativo no sucesso do sistema. A sua função está diretamente relacionada à gestão estratégica e operacional da farmácia hospitalar, sendo essencial o envolvimento contínuo em todas as etapas de desenvolvimento e execução do projeto.
* **Alto Poder, Baixo Interesse:** Este grupo inclui os médicos, farmacêuticos e enfermeiros, que desempenham um papel essencial nas operações clínicas e logísticas. Apesar de terem um elevado poder de influência, o seu interesse direto no sistema pode ser reduzido, pelo que a estratégia deve focar-se em mantê-los informados e envolvidos sem sobrecarregá-los com detalhes técnicos.
* **Baixo Poder, Alto Interesse:** Este grupo abrange os secretários clínicos, que apesar de não terem uma alta influência nas decisões estratégicas, demonstram elevado interesse no sistema devido ao impacto direto que irá existir nos seus processos diários. É necessário manter este grupo bem informado e assegurar que as suas necessidades sejam atendidas.
* **Baixo Poder, Baixo Interesse:** Este grupo inclui os assistentes, cuja participação no sistema é menos significativa. Embora não sejam uma prioridade, é importante manter um canal de comunicação para garantir a sua colaboração sempre que necessário.

## Utilizadores do Sistema

Neste subtópico são identificados os utilizadores potenciais do sistema.

Primeiramente, é-nos apresentado um farmacêutico que trata dos consumíveis nos pedidos; em seguida, em cada ala hospitalar, para a gestão deles, temos um gestor responsável e, por fim, o médico, enfermeiro, o assistente e o secretário clínico têm a oportunidade de requisitar os materiais necessários.

# Análise de Requisitos

Neste capítulo, será apresentado o processo de levantamento e análise de requisitos que levou ao desenvolvimento da solução apresentada. Este levantamento inclui a definição detalhada dos requisitos funcionais, que descrevem as funcionalidades que o sistema fornece para atender às necessidades dos utilizadores, bem como os requisitos não funcionais, que estabelecem as características de desempenho, segurança e qualidade do sistema [17].

Além disso, será apresentada a priorização das funcionalidades, que é essencial para guiar o desenvolvimento do software. Os requisitos identificados abrangem desde a autenticação de utilizadores até a finalização de pedidos de consumíveis pelo requerente.

## Requisitos Funcionais - RF

Na Tabela 1 encontra-se identificados os requisitos funcionais, que detalham as funcionalidades do software e também os instigadores beneficiados. Na Tabela 2, apresenta-se a priorização das funcionalidades a serem desenvolvidas.

|  |  |  |  |
| --- | --- | --- | --- |
| Requisitos Funcionais | Tipo | Descrição | Instigadores Beneficiados |
| RF1 | **Autenticação** | O sistema deve fornecer uma funcionalidade que permita aos utilizadores autenticarem-se para aceder às funcionalidades da aplicação, através de email e palavra-passe. | Todos os utilizadores |
| RF2 | **Criar Utilizadores** | O sistema deve fornecer a criação de utilizadores, associando-os a papéis específicos (médicos, enfermeiros, gestores, etc.). | Administradores do Sistema |
| RF3 | **Associar Gestores Responsável** | O sistema deve fornecer que gestores responsáveis sejam associados a setores específicos, como enfermarias ou blocos operatórios. | Administradores do Sistema |
| RF4 | **Criar Setores Hospitalares** | O sistema deve fornecer criar setores hospitalares. | Administradores do Sistema |
| RF5 | **Criar Novos Consumíveis** | O sistema deve permitir criar consumíveis não existentes no sistema. | Administradores do Sistema |
| RF6 | **Gerar Relatórios** | O sistema deve permitir gerar relatórios detalhados em formato PDF do *stock* existente, realocações dos consumíveis e detalhes dos requerimentos | Farmacêuticos, Médicos, Enfermeiros, Secretários Clínicos, Assistentes e Gestores Responsáveis |
| RF7 | **Visualizar Quantidade Consumíveis** | O sistema deve permitir verificar a quantidade de consumíveis em *stock* e os consumíveis alocados para os pedidos dos requerimentos. | Administradores do Sistema, Farmacêuticos |
| RF8 | **Alterar Quantidade Mínima Consumíveis** | O sistema deve permitir a alteração da quantidade mínima que deve existir daquele consumível em *stock* e quantidade do pedido no caso dos níveis *do stock* foram inferiores a quantidade mínima. | Administradores do Sistema, Farmacêuticos |
| RF9 | **Validar Requerimentos** | O sistema deve fornecer uma funcionalidade que permita validar ou rejeitar os requerimentos. | Gestores Responsáveis |
| RF10 | **Submeter Requerimentos** | O sistema deve permitir aos utilizadores de criar requerimentos, onde são pedidos de medicamentos, vacinas e/ou materiais hospitalares. | Farmacêuticos, Médicos, Enfermeiros, Secretários Clínicos, Assistentes |
| RF11 | **Visualizar Requerimentos** | O sistema deve permitir que os utilizadores visualizem os detalhes dos requerimentos, incluindo detalhes do pedido, estado atual do requerimento, data do pedido, data de validação, data de preparação e data de entrega e o utilizador responsável por cada ação. | Farmacêuticos, Médicos, Enfermeiros, Secretários Clínicos, Assistentes e Gestores Responsáveis |
| RF12 | **Requerimentos Urgentes** | O sistema deve fornecer uma funcionalidade visual de perceber quais ou qual o requerimento que seja com maior urgência. | Farmacêuticos, Médicos, Enfermeiros, Secretários Clínicos, Assistentes |
| RF13 | **Realocar Consumíveis** | O sistema deve fornecer uma funcionalidade de realocar consumíveis para alocar os consumíveis aos requerimentos para prosseguir com o processo. | Farmacêuticos |
| RF14 | **Notificações do Utilizador** | O sistema deve enviar notificações aos utilizadores sobre alterações dos estados dos requerimentos. | Médicos, Enfermeiros, Secretários Clínicos, Assistentes e Gestores Responsáveis |
| RF15 | **Requerimentos Externos** | O sistema deve fornecer uma funcionalidade para obter requerimentos externos ao hospital. | Farmacêuticos, Serviços Externos |
| RF16 | **Preparar Consumíveis** | O sistema deve fornecer uma funcionalidade de enviar os requerimentos para preparação dos consumíveis. | Farmacêuticos |
| RF17 | **Entregar Consumíveis** | O sistema deve fornecer uma funcionalidade de entrega dos consumíveis preparados. | Farmacêuticos |
| RF18 | **Definir Consumíveis Requerimento Externo** | O sistema deve fornecer uma opção de definir os consumíveis necessários para satisfazer o requerimento externo. | Farmacêuticos |
| RF19 | **Pedir Consumíveis Fornecedor** | O sistema deve fornecer uma opção de pedir consumíveis aos fornecedores para reabastecer o *stock*. | Farmacêuticos |
| RF20 | **Utilizar *MedReader*** | O sistema deve ter um sistema adicional para validar os consumíveis que serão enviados para o requerente. | Farmacêuticos |
| RF21 | **Selecionar Urgência** | O sistema deve fornecer uma opção de conseguir definir o tipo de urgência do requerimento. | Médicos, Enfermeiros |
| RF22 | **Selecionar Consumíveis** | O sistema deve fornecer uma funcionalidade de selecionar os consumíveis para efetuar o pedido do requerimento. | Médicos, Enfermeiros, Secretários Clínicos, Assistentes e Farmacêuticos |
| RF23 | **Validar Consumíveis Entregues** | O sistema deve fornecer uma funcionalidade para validar os consumíveis entregues do requerimento. | Médicos, Enfermeiros, Secretários Clínicos, Assistentes |
| RF24 | **Finalizar Requerimento** | O sistema deve fornecer uma funcionalidade de finalizar o requerimento quando está concluído. | Médicos, Enfermeiros, Secretários Clínicos, Assistentes |
| RF25 | **Reportar Erro nos Consumíveis Entregues** | O sistema deve fornecer uma funcionalidade de conseguir reportar erro nos consumíveis entregues. | Médicos, Enfermeiros, Secretários Clínicos, Assistentes |
| RF26 | **Requerimentos Prioritários** | O sistema deve fornecer uma funcionalidade de apenas apresentar os requerimentos urgentes. | Farmacêuticos |
| RF27 | **Redistribuir Consumíveis** | O sistema deve automaticamente efetuar a redistribuição de consumíveis entre requerimentos para atender os requerimentos urgentes rapidamente. | Todos os utilizadores |
| RF28 | **Registar Quantidades Alocadas** | O sistema deve registar automaticamente as quantidades alocadas de consumíveis em todos os requerimentos. | Todos os utilizadores |
| RF29 | ***Stock* Após Redistribuição** | O sistema deve recalcular automaticamente as quantidades de consumíveis disponíveis e alocadas após a redistribuição. | Todos os utilizadores |
| RF30 | **Verificar *Stock* para Novos Requerimentos** | O sistema deve verificar automaticamente a disponibilidade dos consumíveis ao registar um novo requerimento. | Todos os utilizadores |
| RF31 | **Visualizar Realocações de Consumíveis** | O sistema deve permitir visualizar as realocações que foram efetuadas. | Farmacêuticos |
| RF32 | **Pedidos Fornecedor Automático** | O sistema deve verificar automaticamente o *stock* existente, caso seja menor que a quantidade mínima efetua um pedido de consumíveis ao fornecedor. | Todos os utilizadores |

Tabela 1 - Requisitos Funcionais

|  |  |
| --- | --- |
| Requisitos Funcionais | Prioridade |
| RF1 | Alta |
| RF2 | Alta |
| RF3 | Média |
| RF4 | Média |
| RF5 | Alta |
| RF6 | Média |
| RF7 | Alta |
| RF8 | Alta |
| RF9 | Alta |
| RF10 | Alta |
| RF11 | Alta |
| RF12 | Alta |
| RF13 | Alta |
| RF14 | Média |
| RF15 | Baixa |
| RF16 | Alta |
| RF17 | Alta |
| RF18 | Média |
| RF19 | Média |
| RF20 | Alta |
| RF21 | Média |
| RF22 | Alta |
| RF23 | Alta |
| RF24 | Alta |
| RF25 | Alta |
| RF26 | Média |
| RF27 | Alta |
| RF28 | Alta |
| RF29 | Alta |
| RF30 | Alta |
| RF31 | Média |
| RF32 | Média |

Tabela 2 - Requisitos Funcionais - Prioridade

## Requisitos Não Funcionais – *RNF*

Os requisitos não funcionais (*RNFs*) de um sistema definem as características, qualidades e restrições que deve apresentar para garantir o funcionamento adequado. Estes requisitos especificam como o sistema deve comportar-se ao desempenhar as suas funcionalidades, abrangendo aspetos como desempenho, segurança, usabilidade e escalabilidade, assegurando que o software cumpre os seus objetivos, mas também atende aos padrões esperados de qualidade [17].

Na Tabela 3 estão identificados os requisitos não funcionais, descrevendo as suas características e restrições para garantir o desempenho, a qualidade e a eficiência do sistema.

|  |  |  |  |
| --- | --- | --- | --- |
| Requisitos Não Funcionais | Tipo | Descrição | Instigadores Beneficiados |
| RNF1 | **Desempenho** | O sistema deve garantir uma alta disponibilidade e tempos de resposta  rápidos para garantir que os utilizadores tenham acesso imediato às informações, especialmente em situações críticas. | Todos os Utilizadores |
| RNF2 | **Escalabilidade** | O sistema deve permitir a adição de novos utilizadores sem necessidade de reconfiguração significativas. | Administradores do Sistema |
| RNF3 | **Intuitividade** | O sistema deve ter uma interface simples, intuitiva e amigável para garantir que seja fácil de usar, mesmo para utilizadores com pouca experiência em tecnologia. | Todos os Utilizadores |
| RNF4 | **Integridade dos Dados** | O sistema deve garantir que nenhuma informação seja perdida durante atualizações, falhas ou reinicializações do sistema. | Todos os Utilizadores |
| RNF5 | **Compatibilidade com Windows** | O sistema deve ser projetado para funcionar exclusivamente no sistema operativo Windows, garantindo compatibilidade total com as suas funcionalidades e ambiente de execução. | Todos os Utilizadores |
| RNF6 | **Disponibilidade** | O sistema deve estar disponível 99% do tempo, garantindo o funcionamento contínuo e minimizando interrupções. | Todos os Utilizadores |
| RNF7 | **Manutenibilidade** | O sistema deve permitir atualizações e correções de bugs sem interrupções prolongadas ou perda de dados. | Todos os Utilizadores |
| RNF8 | **Requisitos Legal** | O sistema deve atender a todas as normas regulamentadas e legais aplicáveis ao ambiente hospitalar, incluindo requisitos específicos de segurança e auditoria. | Todos os Utilizadores |

Tabela 3 - Requisitos Não Funcionais

## Casos de Uso *MedStock*

No contexto do software desenvolvido, o diagrama de casos de uso é uma ferramenta importante para a compreensão das funcionalidades disponíveis e da interação entre os utilizadores e o sistema [5].

Após a autenticação no sistema, os utilizadores têm acesso às funcionalidades específicas relacionadas às suas funções. O sistema foi desenhado para garantir uma gestão automática dos consumíveis, requerimentos e relatórios, com diferentes níveis de acesso e responsabilidades:

O sistema é composto por quatro atores, administradores, gestores responsáveis, farmacêuticos e os requerentes (Médicos, Enfermeiros, Secretários Clínicos e Assistentes). Os casos de uso representados na Figura 3,serão detalhados de seguida.

Após a autenticação no sistema, os utilizadores têm acesso às funcionalidades específicas relacionadas às suas funções.

1. **Requerentes**

* O requerente tem a possibilidade de criar e submeter requerimentos.
* O processo de submissão inclui a seleção de consumíveis e a definição da urgência do pedido.
* Após a submissão, o requerente pode acompanhar o estado do requerimento e no final validar os consumíveis entregues.
* Caso existam erros nos consumíveis entregues, o requerente pode reportar essas falhas.

1. **Gestor Responsável**

* O gestor é responsável por validar os requerimentos submetidos, podendo aceitar ou recusar estes pedidos.

1. **Farmacêutico:**

* Este é responsável pela gestão do *stock*, incluindo a alteração de quantidades mínimas e as quantidades pedidas de cada consumível.
* Os farmacêuticos preparam os consumíveis solicitados e garantem a entrega aos requerentes.
* Estes são também responsáveis pela resposta aos requerimentos externos e a solicitação de consumíveis a fornecedores externos.
* Este também tem acesso às realocações de consumíveis que foram efetuados automaticamente pelo sistema.

1. **Administrador:**

* O administrador tem acesso a funcionalidades de gestão e parametrização do sistema.
* Este pode criar utilizadores, setores hospitalares, consumíveis e configurar os gestores responsáveis para diferentes áreas do hospital.

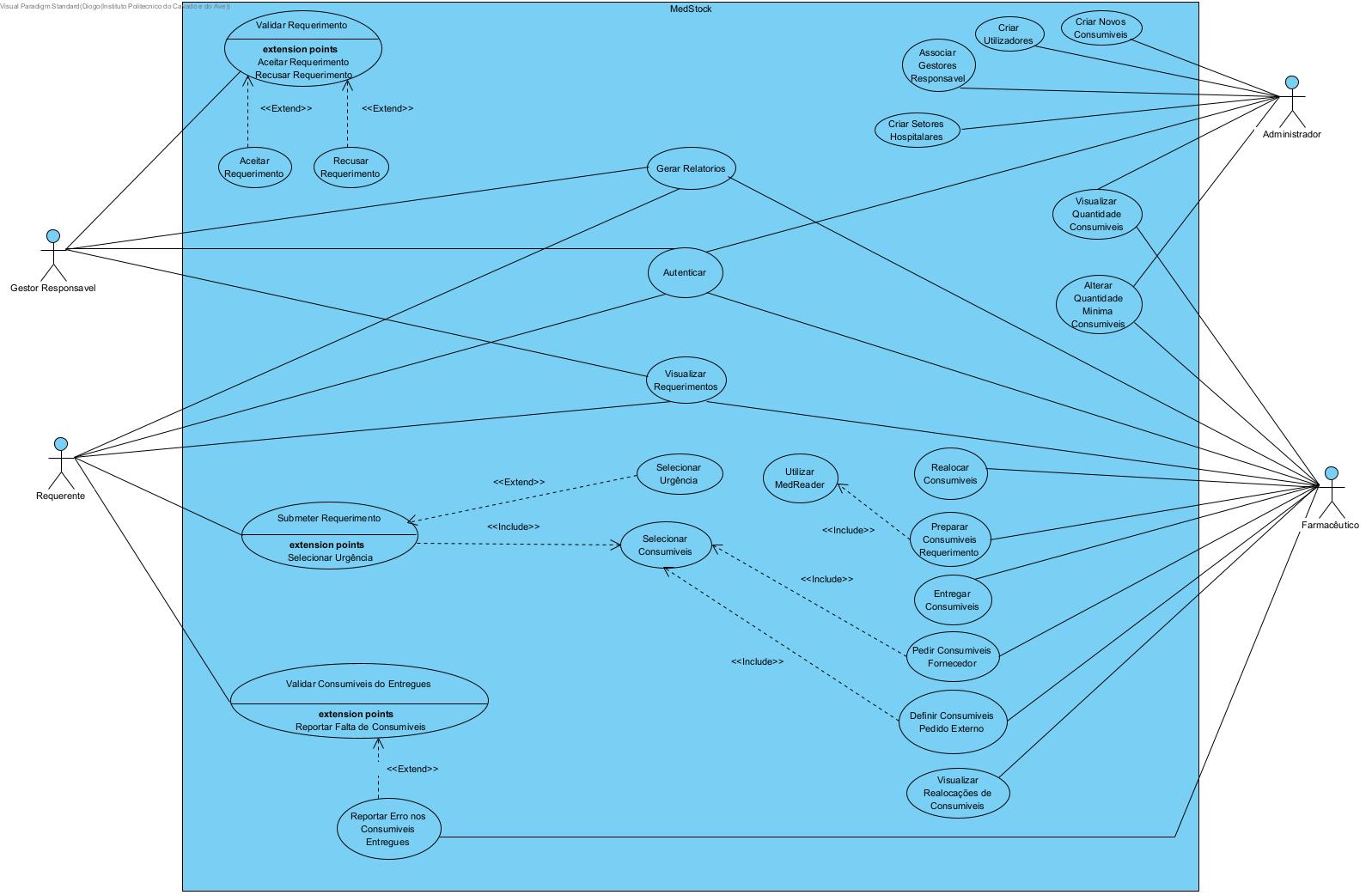


Figura 3 - Diagrama de Casos de Uso

## Diagrama Entidade Relação

O Diagrama de Entidade-Relação (*DER*) é uma representação gráfica que descreve a interação entre diferentes entidades, como pessoas, locais ou eventos, num sistema. Este diagrama permite compreender de forma clara o fluxo de dados entre entidades e a estrutura do sistema [16].

O *DER* apresentado na Figura 4, foi concebido e implementado como a base de dados do sistema. Em seguida, é apresentado uma descrição detalhada de cada entidade do diagrama, incluindo o seu propósito.

***Consumivel***

* Esta tabela armazena as informações sobre todos os consumíveis disponíveis no sistema.
* Representa os consumíveis em *stock*, como medicamentos, vacinas ou materiais hospitalares.

***Tipo\_Consumivel***

* Esta tabela armazena os diferentes tipos de consumíveis no sistema.
* Serve para categorizar os consumíveis, como medicamentos, vacinas ou outros materiais hospitalares.

***PedidoFornecedor***

* Esta tabela armazena os pedidos feitos aos fornecedores para reposição do *stock*.
* Contém informações como o fornecedor, data do pedido, data de entrega e estado atual do pedido.

***PedidoFornecedor\_Consumivel***

* Esta tabela intermédia armazena, os consumíveis específicos incluídos em cada pedido ao fornecedor.
* Contém a quantidade solicitada para cada consumível.

***Requerimento***

* Esta tabela armazena informações sobre os pedidos de consumíveis feitos pelos requerentes.
* Contém dados como o setor para o envio, o utilizador responsável, o tipo de requerimento e se é urgente.

***Consumivel\_Requerimento***

* Esta tabela intermédia armazena a relação entre os consumíveis e os requerimentos.
* Contém informações como a quantidade solicitada e a quantidade alocada.

***Redistribuicao***

* Esta tabela armazena as redistribuições de consumíveis entre requerimentos para atender pedidos urgentes.
* Contém informações como os requerimentos de origem e destino, o consumível redistribuído e a quantidade.

***Setor\_Hospital***

* Esta tabela armazena informações sobre os setores hospitalares para onde serão enviados os consumíveis pedidos.
* Contém detalhes como o nome, localização e o responsável pelo setor.

***Utilizador***

* Esta tabela armazena informações sobre todos os utilizadores do sistema.
* Representa a entidade principal para identificar e diferenciar os utilizadores.

***Role***

* Esta tabela armazena as diferentes funções que um utilizador pode ter no sistema, como administrador, farmacêutico ou gestor.
* É utilizado para controlo de acessos e permissões.

***Requerimento\_Externo***

* Esta tabela armazena informações sobre os pedidos de consumíveis vindos de entidades externas ao hospital.
* Contém informações sobre o paciente e a data de criação do requerimento.

***HistoricoRequerimento***

* Esta tabela armazena informações sobre o histórico de alterações no estado dos requerimentos.
* Contém detalhes como a data da modificação, o estado atual e o utilizador responsável.

***ValidacaoEntrega***

* Esta tabela armazena informações sobre a validação dos consumíveis entregues.
* Contém detalhes como o estado da validação, erros encontrados e a data de validação.

***Status\_Requerimento***

* Esta tabela armazena os diferentes estados possível de um requerimento no sistema.
* Contem detalhes como o identificador e uma descrição do estado.

Uma imagem com texto, escrita à mão, Retângulo, Paralelo

Descrição gerada automaticamente

Figura 4 - Diagrama Entidade Relação

## Diagrama de Contexto

O diagrama de contexto, também conhecido como diagrama de fluxo de dados de nível zero, é uma representação de alto nível que ilustra a interação de um sistema com os seus elementos externos, como utilizadores, organizações ou outros sistemas [14].

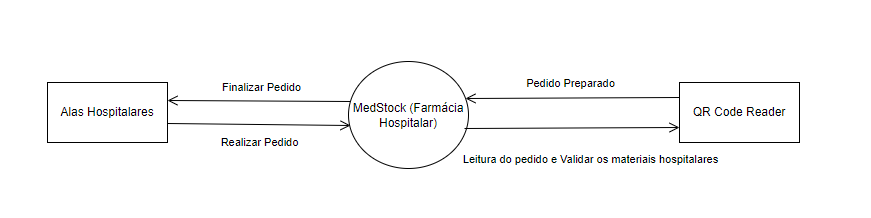


Figura 5 - Diagrama de Contexto

## Diagrama *Business Process Modeling Notation*

O *Business Process Modeling Notation* (*BPMN*) é uma técnica utilizada para modelar graficamente os processos de negócio de forma padronizada. Esta ferramenta possibilita uma visão clara e estruturada das atividades que compõem os procedimentos dentro de uma organização, simplificando a comunicação entre os diferentes intervenientes [13].

No âmbito do projeto da farmácia hospitalar, a utilização do *BPMN* permite representar os processos de forma visual, facilitando a análise e otimização das operações.

O diagrama *BPMN*, apresentado na Figura 6, ilustra o processo de gestão de requerimentos numa farmácia hospitalar, apresentado os detalhes desde a submissão inicial pelos requerentes até à validação final.

1. **Início do Processo**

O processo inicia-se quando o requerente acede ao sistema. Caso não esteja autenticado, é solicitado que efetue o login para prosseguir.

1. **Submissão do Requerimento**

Após a autenticação, o requerente pode criar e submeter um requerimento. Durante este processo, é possível definir a urgência e os consumíveis necessários.

* Caso seja urgente, o requerimento passa automaticamente para a analise do requerimento onde será verificada a disponibilidade do *stock*.
* Se não for urgente, o requerimento necessita de aprovação do gestor da ala hospitalar para onde serão enviados os consumíveis.

1. **Validar Requerimento**

Quando um requerimento não urgente é enviado para o gestor da ala hospitalar, este dispõe de duas opções de validação:

* Caso o gestor considere que o pedido é necessário, o requerimento é aprovado e segue para a etapa de análise.
* Caso o gestor entenda que o pedido não é necessário, o requerimento é rejeitado e o processo é finalizado.

Independentemente da decisão tomada pelo gestor, o requerente é automaticamente notificado por email sobre o estado do pedido.

1. **Analise do Requerimento**

De forma automática o sistema verifica a disponibilidade dos consumíveis:

* Se o *stock* disponível for suficiente, os consumíveis são alocados e o processo avança para a fase de preparação e o requerente é informado via email sobre o estado.
* Se não houver consumíveis suficientes, o requerimento é colocado em "*Stand By*" e o requerente é informado via email sobre o estado.

1. **Alocar Consumíveis**

Quando um requerimento se encontra em estado de "*Stand-By*", é necessário verificar a disponibilidade dos consumíveis para que este possa prosseguir para a seguinte etapa.

* Se, ao realizar a verificação, existir *stock* suficiente para atender às necessidades do requerimento, os consumíveis são alocados e o estado do requerimento é atualizado, permitindo que este seja enviado para a lista de espera.
* Caso ainda não exista *stock* suficiente para atender o requerimento, este permanece em *"Stand-By"*, aguardando a reposição de *stock* para prosseguir o processo.

1. **Validação do Requerimento**

O requerimento submetido é encaminhado para validação pelo gestor responsável. Este avalia o pedido e decide se aprova ou rejeita o requerimento.

* Caso seja rejeitado, o requerimento é encerrado e o requerente é informado via email sobre a decisão.
* Se aprovado, o requerimento avança para a fase de lista de espera e o requerente é informado via email sobre o estado.

1. **Preparação do pedido**

Após a disponibilidade dos consumíveis requisitados, o pedido é preparado e o requerente é informado via email sobre o estado.

* Durante esta etapa, o sistema *MedReader* é utilizado para validar os consumíveis, assegurando assim que os consumíveis preparados correspondem aos solicitados.

1. **Entrega dos Consumíveis e Finalização do Processo**

Concluída a preparação, o pedido é entregue ao requerente, que realiza uma validação para confirmar que os consumíveis fornecidos correspondem aos solicitados.

* Caso a entrega esteja conforme o pedido realizado, o requerimento é finalizado.
* Se a entrega não estiver de acordo, o requerimento é reavaliado para identificar e corrigir possíveis falhas.

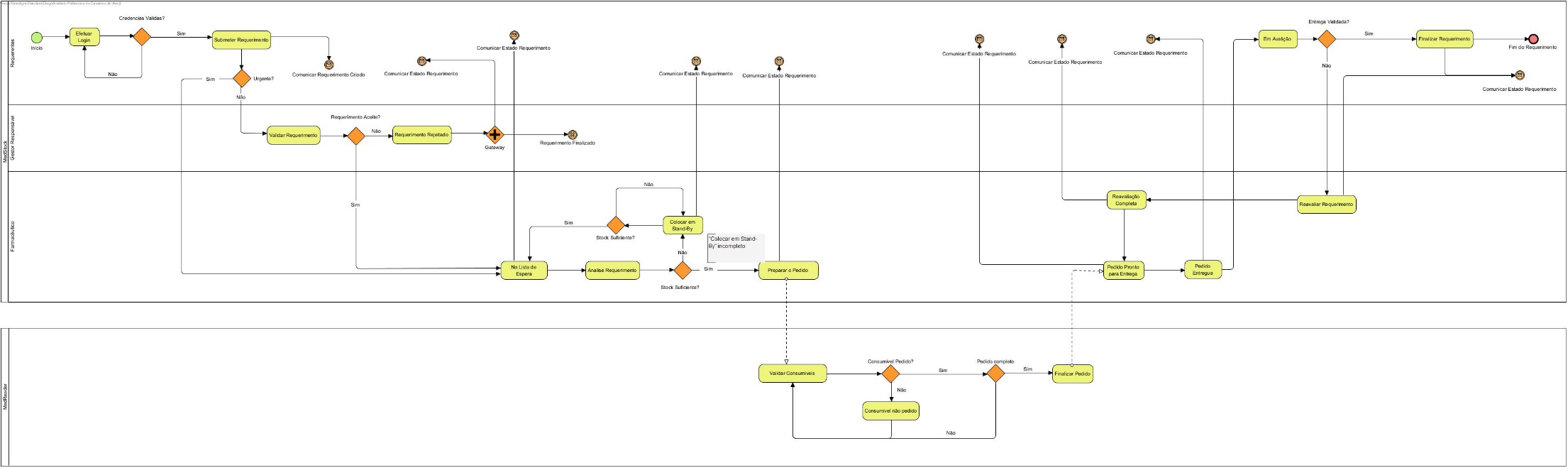


Figura - Diagrama BPMN

## Diagrama de Estados

O diagrama de estados é uma ferramenta visual que representa os estados e as transições entre eles, facilitando assim a compreensão do ciclo de vida e as diferentes combinações de informações que este pode conter [15].

Abaixo, na Figura 7, é apresentado o diagrama de estados que reflete os processos associados à gestão de requerimentos, destacando os estados pelos quais um requerimento pode passar e as transições entre eles.

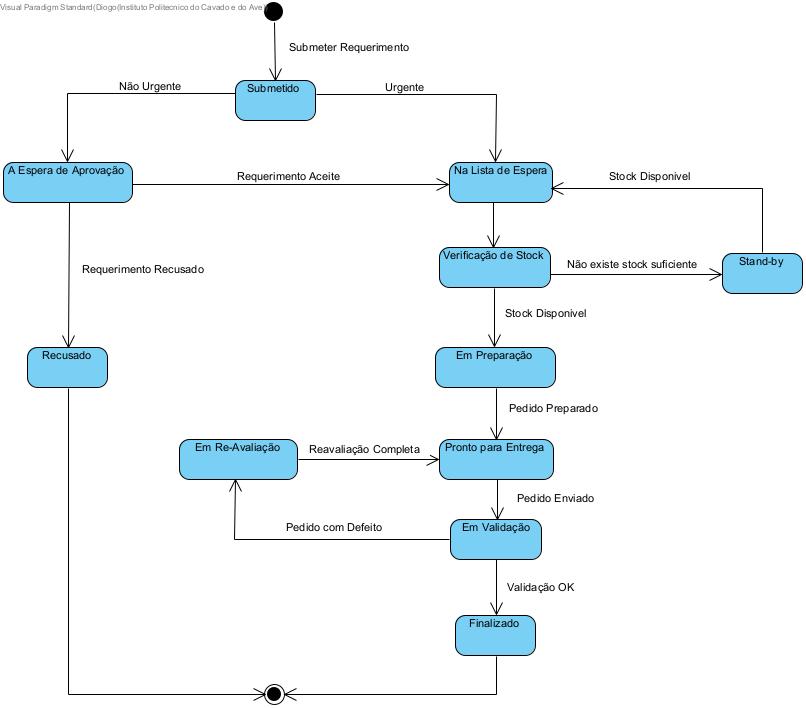


Figura 7 – Diagrama de Estados

O processo inicia com a submissão de um requerimento, que pode ser do tipo "Urgente" ou "Não Urgente". No caso de um requerimento "Não Urgente", este avança para o estado "À Espera de Aprovação", onde é avaliado pelo gestor da responsável da ala hospital onde serão enviados os consumíveis finais daquele requerimento.

Se for recusado, muda para o estado "Recusado” e o processo é finalizado.

Caso seja aprovado ou o requerimento seja do tipo urgente, realiza-se a "Verificação *do stock*". Se não houver *stock* suficiente, o requerimento entra no estado "*Stand-by*", ficando à espera até que os consumíveis necessários estejam disponíveis.

Quando o *stock* é suficiente, o processo avança para o estado "Na Lista de Espera", onde fica até este requerimento seja enviado para a fase de preparação.

Quando o requerimento é enviado para a fase de preparação, processo avança para o estado "Em Preparação", onde os consumíveis são organizados e preparados para envio.

Após a preparação, o estado muda para "Pronto para Entrega". Neste ponto, o pedido é enviado ao requerente e passa para "Em Validação", onde o requerente verifica a conformidade dos consumíveis recebidos.

Se a validação for bem-sucedida, o processo é finalizado. Contudo, se forem encontradas discrepâncias, o pedido é colocado no estado "Em Reavaliação". Neste estado, o processo é reanalisado e pode seguir novamente os passos de entrega e validação até que seja corretamente concluído e finalizado.

# Implementação

Neste capítulo, é apresentado o processo de implementação do sistema *MedStock*, detalhando as tecnologias utilizadas e a integração entre as diferentes camadas que o compõem. Esta fase concretiza as ideias que foram anteriormente delineadas, transformando-as num sistema funcional.

O *MedStock* integra diversas tecnologias para garantir a eficiência e funcionamento. O *Front-End* foi desenvolvido em *Python*. A comunicação com o *Back-End* é feita através de uma *API REST*, desenvolvida em *Python* e hospedada em uma plataforma *online*, permitindo que a solução esteja sempre disponível e qualquer dispositivo consiga ter acesso.

No *Back-End*, foi elaborada uma base de dados, hospedada também em uma plataforma *online*, para permitir que esta esteja sempre online e disponível para a *API* ter acesso aos dados da base de dados.

Também foi desenvolvida uma aplicação móvel, *MedReader*, para dispositivos *Android*. Esta aplicação conecta-se diretamente à *API* para executar funcionalidades como validações de consumíveis quando o requerimento está em fase de preparação.

Outro sistema implementado foi, um simulador denominado *MedOcorrencias*, uma plataforma *Web*, destinada a processar e enviar pedidos externos provenientes de serviços de emergência, como, por exemplo, ambulâncias.

Por último, foi desenvolvido um simulador adicional, apelidado de *MedSupply*, que funciona como um sistema externo integrado ao *MedStock*. O objetivo principal do *MedSupply* é permitir a criação de pedidos de reabastecimento de *stock* de consumíveis.

## Metodologia

Durante o desenvolvimento do projeto, o grupo adotou uma abordagem híbrida entre *SCRUM* e *Kanban*, adaptada às necessidades para o desenvolvimento do projeto. Esta metodologia combinou a estrutura e os princípios ágeis do *SCRUM* com a flexibilidade visual do *Kanban*, promovendo uma organização mais eficiente e uma comunicação mais fácil entre os membros da equipa.

As reuniões de planeamento e atualização das tarefas eram realizadas semanalmente, estas serviam para definir as tarefas mais prioritárias e organizar o fluxo de trabalho e manter o acompanhamento do projeto.

No que diz respeito a divisão de papeis e responsabilidades de cada elemento da equipa de desenvolvimento, foi definida de forma clara e organizada:

* Bruno Oliveira assumiu o papel de liderança, como gestor de projeto e/ou *Product Owner*. Assumindo também um papel ativo no desenvolvimento da aplicação *MedStock.*
* Diogo Pinheiro, foi responsável pelo desenvolvimento da aplicação móvel intitulada de *MedReader*.
* Ana Pinto, foi responsável pelo desenvolvimento do simulador *Web* de pedidos externos.
* Diogo Fernandes foi responsável pelo desenvolvimento do simulador de pedidos a fornecedores, também desenvolvida uma *API* para contribuir com a simulação dos processos externos relacionados aos pedidos para o reabastecimento *do stock*.
* Em relação a documentação do projeto, todos os elementos tiverem um papel ativo para a realização do mesmo.

## Arquitetura

A arquitetura do sistema foi desenvolvida com base no modelo *SOA* (Arquitetura Orientada a Serviços), como se pode visualizar na Figura 8, escolhido para garantir a interoperabilidade entre os diferentes sistemas e simuladores. Esta abordagem permitiu que cada componente funcione de forma independente, facilitando a comunicação e a integração entre as diversas partes da solução.

No caso do *MedStock*, o *SOA* permitiu a interligação entre a base de dados, a *API*, e as interfaces do utilizador, a aplicação móvel e o simulador *Web*. A *API* desempenha um papel de mediador nas trocas de dados entre os componentes.

Além destes, o simulador de pedidos a fornecedores, contribui para a simulação dos processos externos relacionados aos pedidos para o reabastecimento *do stock*, integrando-se de forma modular com o sistema.

Com esta arquitetura, o sistema suporta a inclusão de novos serviços ou módulos, assegurando flexibilidade no desenvolvimento e manutenção da solução.

Em suma, a escolha pela arquitetura *SOA* foi a estratégia para criar e fornecer um sistema escalável e integrado, capaz de suportar as necessidades da farmácia hospitalar e possibilitar a interoperabilidade entre todos os sistemas envolvidos.

Uma imagem com captura de ecrã, diagrama

Descrição gerada automaticamente

Figura 8 - Arquitetura do Sistema

## Tecnologias

Neste capítulo, serão abordadas as tecnologias utilizadas na implementação do projeto, desde a construção da interface do utilizador com *Python* em conjunto com a *framework PyQt*, até à base de dados em *PostgreSQL*, passando também pela *API* desenvolvida na *framework* *FastAPI* e pelos simuladores desenvolvidos em diferentes linguagens.

A escolha do *PostgreSQL* como base de dados foi motivada pela sua fiabilidade e capacidade de lidar com grandes volumes de dados. Esta base de dados está hospedada no servidor *Supabase*, uma plataforma que já era familiar ao grupo devido a trabalhos anteriores. A utilização do *Supabase* permitiu que a base de dados permanecesse online, garantindo assim a acessibilidade à *API* e facilitando a colaboração e execução de tarefas entre os membros do grupo.

O *Front-End* do *MedStock* foi desenvolvido em *Python* com recurso à *framework PyQt*, escolhida devido ao *background* existente na linguagem *Python* dentro do grupo e pela familiaridade com a versão em *C++* do *Qt*. Além disso, a comunidade e a documentação disponível para o *PyQt* proporcionaram um suporte rápido e ágil para resolver problemas durante o desenvolvimento.

A *API REST*, o mediador entre o *Front-End* e o *Back-End*, foi implementada com a *framework FastAPI*. A escolha pelo *FastAPI* baseou-se no conhecimento prévio adquirido em projetos passados, o que proporcionou um *know-how* sobre o funcionamento da biblioteca, facilitando no desenvolvimento.

Para hospedar a *API*, optou-se pela plataforma *Render*, uma solução que foi utilizada pela primeira vez no contexto de projeto. Inicialmente, o desconhecimento sobre o a tecnologia e o funcionamento do *Render* representaram muitos problemas ao colocar a solução online. No entanto, após a leitura da documentação e a realização de vários testes, foi possível configurar corretamente o servidor da plataforma e colocar a *API* em funcionamento. Esta configuração garantiu com que a *API* estivesse online para todos os serviços e aplicações do sistema.

Adicionalmente, foi desenvolvida uma aplicação móvel, codificada em *Kotlin* para dispositivos *Android*. Esta escolha foi a primeira experiência com a linguagem *Kotlin*, representando um desafio inicial por se tratar de uma nova linguagem de programação para o grupo. No entanto, após uma breve pesquisa sobre desenvolvimento para *Android*, foi verificado que o *Kotlin* é muito utilizado devido à sua integração nativa com o *Android*.

Além disso, foi criado um simulador *Web*, desenvolvido em *HTML*, *CSS* e *JavaScript*, com o objetivo de serem criados pedidos externos, provenientes de ambulâncias e serviços de emergência. Optou-se por um simulador *Web* devido à necessidade de ser uma aplicação ou uma plataforma online. Embora uma aplicação móvel também fosse uma possibilidade, mas a escolha pelo formato *Web* foi motivada pela experiência prévia existente.

Por fim, foi desenvolvido um simulador de pedidos a fornecedores, codificado em *React* para o *Front-End* e com uma *API* própria implementada em *Python* com recurso à *framework Flask*. Este simulador foi desenvolvido para criar pedidos relacionadas ao reabastecimento de *stock*. A utilização do *Flask* representou uma primeira experiência com esta *framework*, sendo escolhida para testar e avaliar a tecnologia na criação de *APIs*. Já o *React*, embora tenha sido utilizado em alguns projetos anteriores, ainda é uma tecnologia relativamente nova para o grupo, o que também contribuiu como uma forma de aprofundar conhecimentos.

## Integrações

As integrações desempenham um papel fundamental no funcionamento do *MedStock*, garantindo a comunicação entre diferentes componentes do sistema e serviços externos. Estas conexões permitem automatizar processos, melhorar a experiência dos utilizadores e otimizar a gestão de consumíveis e requisições. O papel da *API*, em alguns destes sistemas, atua como um mediador, facilitando a troca de informações entre os diferentes serviços integrados.

### *FireBase*

A integração com o *Firebase*, apresentada na Figura 9, é utilizada exclusivamente quando um utilizador tenta autenticar-se na aplicação. Esta integração possibilita a validação das credenciais inseridas no momento do *login* e também possibilidade de registo de novos utilizadores.

A escolha do *Firebase* deve-se à sua reputação como uma plataforma confiável e segura, desenvolvida pela Google, além de oferecer um serviço robusto de autenticação que simplifica a implementação de mecanismos de controlo de acesso. Durante o processo de autenticação, as credenciais do utilizador são enviadas para o *Firebase*, que valida os dados e responde ao sistema. Caso as credenciais sejam válidas, o *Firebase* retorna uma confirmação de sucesso, permitindo o acesso do utilizador. Em caso de credenciais inválidas ou inexistentes, o *Firebase* retorna uma mensagem de erro, negando a autenticação.

Uma imagem com captura de ecrã, Modelagem 3D, Composição digital, Software de videojogos

Descrição gerada automaticamente

Figura 9 - Esquema Integração com *FireBase*

### *Gmail*

A integração com o *Gmail*, representada na Figura 10, permite a comunicação entre o *MedStock* e os utilizadores, garantindo que estes estejam sempre ocorrentes e informados sobre o estado dos seus pedidos. Com a integração deste serviço obtém-se um serviço automatização da comunicação entre a farmácia e os requerentes.

A escolha do *Gmail* como plataforma de email para esta integração foi motivada pela sua alta fiabilidade e ampla aceitação como plataforma de comunicação desenvolvida pela Google.

Relativamente a troca de dados nesta integração, o sistema central envia o email do destinatário, assunto e o conteúdo do email para o *Gmail*, este depois processa o envio, retornando o estado do envio da mensagem, no caso de sucesso ou não retorna a informação para o sistema central.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 10 - Esquema Integração com *Gmail*

### *MedReader*

O *MedReader*, detalhada na Figura 11, é uma aplicação móvel que se conecta à *API* do sistema para realizar validações de consumíveis através de *QR Codes* ou Códigos de Barras. Esta funcionalidade permite que, durante a preparação de consumíveis, os códigos sejam lidos e comparados com os dados das requisições, garantindo que os itens preparados correspondam exatamente aos solicitados. Esta integração reduz erros manuais e aumenta a eficiência no processo de distribuição de consumíveis.

Esta integração foi desenvolvida com o propósito de demonstrar o funcionamento do sistema e está estruturada de forma a permitir que outras empresas do setor possam desenvolver soluções ou produtos que repliquem ou expandam esta solução.

A troca de dados nesta integração baseia-se em três validações principais. Primeiro, é realizada a autenticação do utilizador, garantindo que apenas farmacêuticos registados no *MedStock* tenham acesso à aplicação. Em seguida, a aplicação consulta os requerimentos que se encontram em fase de preparação. Por fim, é efetuado o envio dos dados relacionados ao processo de finalização da preparação, assegurando o acompanhamento e a rastreabilidade das operações realizadas no sistema central.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 11 - Esquema Integração com *MedReader*

### *MedOcorrências*

O simulador *MedOcorrências*, representado na Figura 12, é uma aplicação web que permite a criação de pedidos externos provenientes de serviços de emergência, como ambulâncias e serviços de urgência.

Esta integração permite a comunicação entre os serviços externos e o sistema central, garantindo que os pedidos sejam processados e atualizados de forma rapida. O *MedOcorrências* foi concebido para auxiliar em emergências, permitindo que, quando um utente é encaminhado para o hospital, os consumíveis necessários para o seu atendimento já estejam preparados para agilizar o processo clínico.

Tal como o *MedReader*, este simulador foi desenvolvido como uma prova de conceito e demonstração de funcionamento, estando aberto para que outras empresas do setor possam criar soluções ou produtos complementares baseados nesta integração.

A troca de dados nesta integração é realizada através do envio de informações como o nome do utente e o seu estado clínico, que são transmitidos do simulador para o sistema central, assegurando a integração dos serviços de emergência com o sistema hospitalar.

Uma imagem com captura de ecrã, computador, Modelagem 3D

Descrição gerada automaticamente

Figura 12 - Esquema Integração com *MedOcorrências*

### *MedSupply*

O *MedSupply*, representado na Figura 13, é um simulador desenvolvido para permitir que o *MedStock* comunique com fornecedores externos para a reposição de consumíveis em falta. Este simulador possibilita a criação de pedidos de consumíveis para o reabastecimento de *stock* e automação de todo o processo de gestão de *stock*. Esta integração foi desenvolvida com o intuito de simular a integração com fornecedores reais, sendo uma preparação para futuras expansões do sistema em ambientes hospitalares reais.

Esta integração foi concebida para que a *API* do *MedSupply* suporte múltiplos fornecedores, refletindo em um contexto realista do mercado, com um maior leque de opções e disponibilidade de consumíveis. O *MedStock* consulta a *API* para obter informações sobre os consumíveis disponíveis por cada fornecedor, permitindo que o farmacêutico selecione os consumíveis necessários e realize o pedido.

Além disso, quando o *stock* de um consumível atinge ou fica abaixo do limite mínimo, o sistema realiza automaticamente uma pesquisa entre os fornecedores disponíveis, priorizando aquele que oferece a entrega no menor tempo útil. Após a análise, o pedido é efetuado com a quantidade previamente configurada no *MedStock*, assegurando a reposição e evitando ruturas de *stock*.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 13 - Esquema Integração com *MedSupply*

## API

Neste tópico, será apresentada a *API* desenvolvida para o sistema *MedStock*, detalhando a sua estrutura, a padronização adotada para as respostas, e a sua documentação. Esta *API* desempenha um papel central no sistema, servindo como intermediário entre os diferentes componentes.

### Estrutura

A estrutura da API foi concebida e implementada de forma modular, alinhada com as integrações realizadas no sistema. Foram desenvolvidos módulos específicos para cada integração, incluindo a autenticação de utilizadores com o *Firebase*, o envio de emails com o *Gmail* e para as aplicações e simuladores associados ao sistema. Além disso, foram implementados módulos dedicados à leitura e escrita de dados na base de dados.

Cada módulo foi projetado com as suas próprias rotas (*routes*), permitindo a execução dos processos de forma organizada. Esta abordagem modular facilita em futuras manutenções e futuras expansões da *API*, assegurando que cada funcionalidade seja tratada de forma independente dentro do sistema.

### Servidor *Render*

A *API* foi hospedada no servidor *Render*, disponível no link [https://med*stock*-api-ce98.onrender.com](https://medstock-api-ce98.onrender.com), garante que a API fique permanentemente online e acessível a todos os componentes e aplicações do sistema. A escolha do *Render* como plataforma de *hosting* permitiu assegurar a disponibilidade do acesso à *API* e a sua integração com os diversos módulos desenvolvidos no projeto. Esta configuração assegura uma comunicação contínua entre o *Front-End*, a base de dados e os sistemas desenvolvidos.

### Documentação

A documentação da *API* é gerada automaticamente pela *framework FastAPI*, que está acessível através do link [https://med*stock*-api-ce98.onrender.com/docs](https://medstock-api-ce98.onrender.com/docs). Esta funcionalidade permite que os desenvolvedores e/ou utilizadores consultem de forma clara e organizada todas as rotas disponíveis. Conseguem verificar desde os métodos suportados, os parâmetros necessários e a estruturas da resposta. A documentação é atualizada em tempo real com base nas alterações realizadas na *API*, facilitando a integração e o desenvolvimento de novos módulos.

### Padronização desenvolvida

Durante o desenvolvimento do sistema, foi estabelecida uma estrutura padronizada para as respostas da *API*. Esta padronização garante consistência e facilita na integração com outros componentes do sistema. Cada resposta segue um formato uniforme, que inclui campos para identificar o estado da operação, mensagens associadas e, quando aplicável, os dados retornados.

As respostas da *API* são construídas com três campos:

* ***response:*** Indica o resultado da operação. É retornado o valor *true* em caso de sucesso ou *false* em caso de erro.
* ***data:*** Contém os dados solicitados em operações bem-sucedidas. Este campo está ausente em respostas de erro.
* ***error:*** Fornece informações detalhadas sobre problemas ocorridos. Este campo está presente apenas em respostas que indicam erro.

Numa operação bem-sucedida, como obter os detalhes de um utilizador pelo seu email, a *API* devolve uma estrutura com os campos definidos, exemplificada na Figura 14, onde apresenta um exemplo de uma resposta ao método *GET*:



Figura 14 - Resposta da *API* do método GET – Sucesso

Nesta resposta, o campo *response* indica que a operação foi concluída com sucesso. O campo *data* fornece as informações obtidas, como o identificador do utilizador, o nome, o email, o sexo, a data de nascimento e a função associada no sistema.

Em caso de erro, como a tentativa de obter informações de um utilizador inexistente, a *API* retorna uma estrutura que identifica o problema, como se pode visualizar na Figura 15. A resposta inclui o campo *response*, indicando um erro durante a operação, e o campo *error* fornece uma descrição detalhada do erro ocorrido. Esta abordagem padronizada facilita a validação, identificação e resolução de problemas no sistema.



Figura 15 - Resposta da *API* do método GET – Erro

A padronização das respostas da *API* contribui diretamente para a eficiência e robustez do sistema. Primeiramente, assegura que as respostas seguem um formato consistente e fácil de interpretar, facilitando a compreensão dos estados das operações. Além disso, simplifica o processo de *debug*, devido às mensagens detalhadas que permitem identificar os problemas rapidamente.

## Aplicação

Neste tópico, serão apresentados os aspetos relacionados ao desenvolvimento das diferentes aplicações que compõem o sistema. Cada aplicação foi desenvolvida para atender as necessidades específicas.

Em baixo estão detalhadas as funcionalidades e características das seguintes aplicações:

* ***MedStock:*** Sistema para gestão de *stock*, requisições internas e alocação e redistribuição de consumíveis entre os requerimentos.
* ***MedReader:*** Aplicação móvel, utilizada para validação de consumíveis durante o processo de preparação, garantindo a conformidade dos pedidos.
* ***MedOcorrencias:*** Simulador *Web*, destinado a criar pedidos externos provenientes de serviços de emergência, como ambulâncias e unidades de emergência.
* ***MedSupply:*** Simulador desenvolvido para criar e gerir os pedidos a fornecedores, para simular os processos externos de reposição de *stock*.

Cada aplicação será descrita em detalhes, destacando o seu papel no sistema.

### *MedStock*

A aplicação *MedStock* foi desenvolvido em *Python* em conjunto com a *framework* *PyQt* no *Front-End*, onde foi utilizado como base um repositório público [22], no qual a interface gráfica já estava pré-definida. Assim, o desenvolvimento do sistema concentrou-se nos ajustes necessários para adaptar a *UI* às funcionalidades específicas do *MedStock*.

O *MedStock* foi desenvolvido com o intuito de automatizar os processos como a alocação de consumíveis, priorizar as requisições urgentes e ajustar dinamicamente os recursos disponíveis. Além disso, a aplicação permite monitorizar níveis *do stock* em tempo real, gerar relatórios detalhados e integrar-se com sistemas externos.

A implementação do *MedStock* visa proporcionar um *software* funcional, alinhada com as necessidades das farmácias hospitalares, garantindo uma gestão de recursos mais eficiente e a melhoria dos cuidados de saúde prestados.

Cada tipo de utilizador no *MedStock* desempenha um papel específico dentro do sistema, o que determina os menus e funcionalidades a que tem acesso. Seguidamente, serão apresentadas as janelas associadas a cada tipo de utilizador, destacando os menus disponíveis e as funcionalidades que podem ser executadas por cada perfil.

#### Administrador

O administrador no sistema desempenha um papel central na gestão de utilizadores e configurações essenciais. Este tipo de utilizador tem acesso a diversas funcionalidades, organizadas em menus específicos, apresentadas nas figuras abaixo.

O administrador tem a funcionalidade de criar utilizadores no sistema, como ilustrado na Figura X. Durante este processo, é também possível atribuir o papel específico que cada novo utilizador criado irá desempenhar, garantindo que as permissões e funcionalidades sejam ajustadas às suas responsabilidades no sistema.

Uma imagem com texto, captura de ecrã, computador, software

Descrição gerada automaticamente

Figura -

Além disso, é possível associar utilizadores a setores específicos, atribuindo-lhes o papel de gestor responsável.

Uma imagem com texto, captura de ecrã, software, computador

Descrição gerada automaticamente

Figura -

Outra funcionalidade disponível é a criação de novos consumíveis, como mostrado na Figura X, permitindo a adição de novos consumíveis para o funcionamento hospitalar. O administrador também pode configurar novos setores hospitalares, conforme ilustrado na Figura X, ampliando as opções disponíveis no sistema.

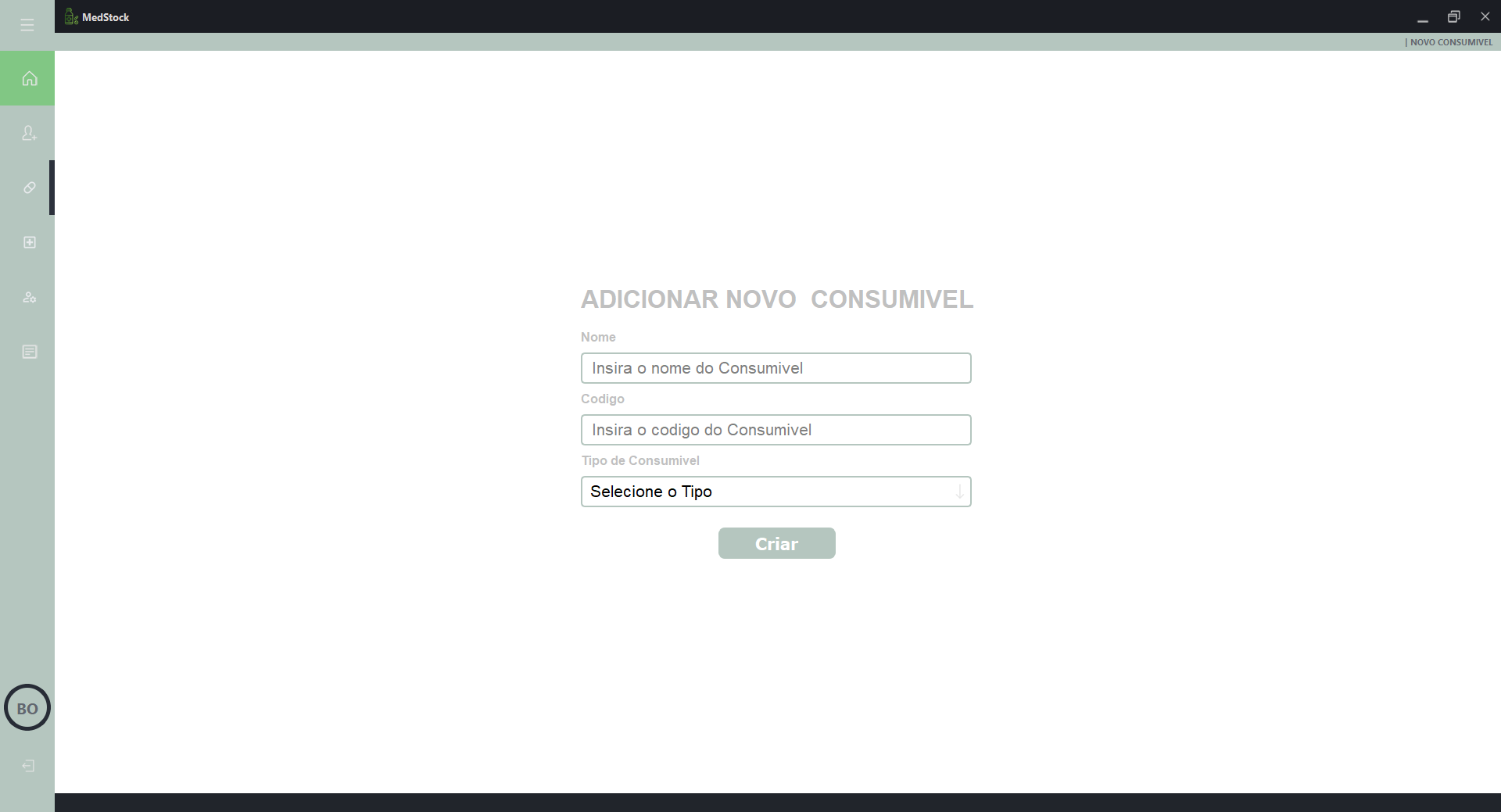


Figura -

Uma imagem com texto, computador, Retângulo, captura de ecrã

Descrição gerada automaticamente

Figura -

Além disso, o administrador tem acesso à visualização do *stock* atual de consumíveis. Nesta funcionalidade, é possível parametrizar o limite mínimo de cada consumível e definir a quantidade a ser automaticamente solicitada quando o *stock* atinge este limite, como exemplificado na Figura X. Neste menu do *stock* atual, o administrador também tem a opção de gerar um relatório em *PDF*, onde este é um extrato de todas as informações sobre os consumíveis, conforme ilustrado na Figura X.

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

Figura -

Uma imagem com texto, captura de ecrã, número, menu

Descrição gerada automaticamente

Figura -

#### Requerentes

#### Gestor de Ala Hospitalar

#### Farmacêutico

#### Algoritmo de gestão *de stock*s

Antes de proceder à implementação e desenvolvimento do algoritmo de gestão de *stock*, foi realizada uma investigação para identificar soluções já existentes e validar as práticas associadas a este tipo de problema. Foram analisados relatórios e artigos científicos, como as referências [1], [3], [4] e [7], que abordam diferentes abordagens na gestão de *stock*. Estes estudos exploram aplicações distintas que utilizam algoritmos de redistribuição e priorização de recursos, além de conceitos matemáticos relacionados, como cálculo de *stock* de segurança, gestão de ruturas de *stock*, entre outras.

Com base nesta pesquisa, foi possível compreender as limitações e vantagens de diferentes abordagens, ajustando o desenvolvimento do algoritmo às necessidades específicas do *MedStock*.

O sistema *MedStock* utiliza um algoritmo de gestão *do stock* projetado para assegurar a alocação eficiente e prioritária dos consumíveis entre os diferentes requerimentos hospitalares. Este algoritmo foi desenvolvido para atender às necessidades dos diversos requerentes, em diferentes setores hospitalares e também em diferentes tipos de urgência. Garantindo assim que os consumíveis sejam distribuídos de forma a minimizar atrasos e atender rapidamente os pedidos urgentes.

A primeira etapa do algoritmo consiste na verificação do *stock* disponível. Para cada pedido submetido, o sistema avalia se o *stock* atual é suficiente para atender à quantidade necessária. Esta verificação considera um “*stock* virtual”, calculado com a diferença entre o *stock* total e o *stock* já alocado para outros requerimentos. Se a quantidade disponível for suficiente, os consumíveis são alocados diretamente ao requerimento. Caso contrário, o sistema aloca apenas a quantidade disponível e automaticamente é criado um pedido externo, para reabastecer o *stock* do consumível em falta.

Quando o *stock* disponível é insuficiente para atender um requerimento urgente, o algoritmo implementado prioriza o atendimento deste tipo de pedidos. Para isso, é realizada uma análise dos consumíveis alocados em requerimentos não urgentes. Esta análise, verifica se somando as quantidades disponíveis, é possível satisfazer integralmente o requerimento urgente. Caso o total de consumíveis alocados nos pedidos não urgentes não seja suficiente, nenhuma redistribuição é efetuada, evitando a criação de múltiplos requerimentos pendentes.

A redistribuição, quando necessária, segue uma ordem predefinida, começando pelos pedidos não urgentes mais recentes. Apenas a quantidade necessária para atender o requerimento urgente é realocada, evitando assim alocações em excesso. Este processo garante que os pedidos urgentes sejam tratados com a máxima prioridade, minimizando o impacto nos pedidos não urgentes e mantendo a eficiência no atendimento geral.

O algoritmo também considera a possibilidade de falhas na alocação. Em casos onde não seja possível atender integralmente um pedido, o sistema atualiza o estado do requerimento para "*Stand-By*". Este mecanismo garante que nenhum pedido seja ignorado, permitindo que o processo seja retomado assim que os recursos necessários estejam disponíveis.

Para representar o funcionamento do algoritmo, exemplificamos com o seguinte exemplo:

Um hospital possui cinco requerimentos não urgentes, onde em cada requerimento foram solicitadas 20 unidades de um consumível X. O *stock* atual está esgotado, e um pedido urgente chega e é solicitado também 30 unidades do mesmo consumível X.

O sistema identifica que é necessário redistribuir os consumíveis já alocados. Inicia-se pelo requerimento não urgente mais recente, são realocadas 20 unidades, e prossegue para o próximo pedido, realocando mais 10 unidades. Com isso, o pedido urgente é atendido integralmente e os pedidos não urgentes são colocados em *StandBy* até que o *stock* seja reabastecido.

Este algoritmo não apenas prioriza a eficiência, mas também a rastreabilidade. Todas as alocações, redistribuições e alterações de estado dos pedidos são registadas na base de dados, garantindo o controlo sobre as operações realizadas.

Com este mecanismo de gestão *do stock*, o *MedStock* consegue assegurar a otimização dos recursos disponíveis, reduzindo falhas e garantindo um atendimento mais ágil e eficiente.

Falar das vantagens e desvantagens deste algoritmo

### *MedReader*

*MedReader* é uma aplicação móvel desenvolvida para dispositivos Android, projetada para atender as necessidades das farmácias hospitalares na gestão dos pedidos de consumíveis. A aplicação tem como principal objetivo validar consumíveis quando o requerimento está em processo de preparação, como também dar prioridade à entrega de consumíveis urgentes.

**Como Funciona?**

1. **Login**

O acesso a aplicação é realizado por meio de um sistema de *login,* como ilustrado na Figura 22. O utilizador acede ao *MedReader* utilizando o *email* e uma *password*. Apenas utilizadores autorizados, como os farmacêuticos, têm permissão para aceder a aplicação.

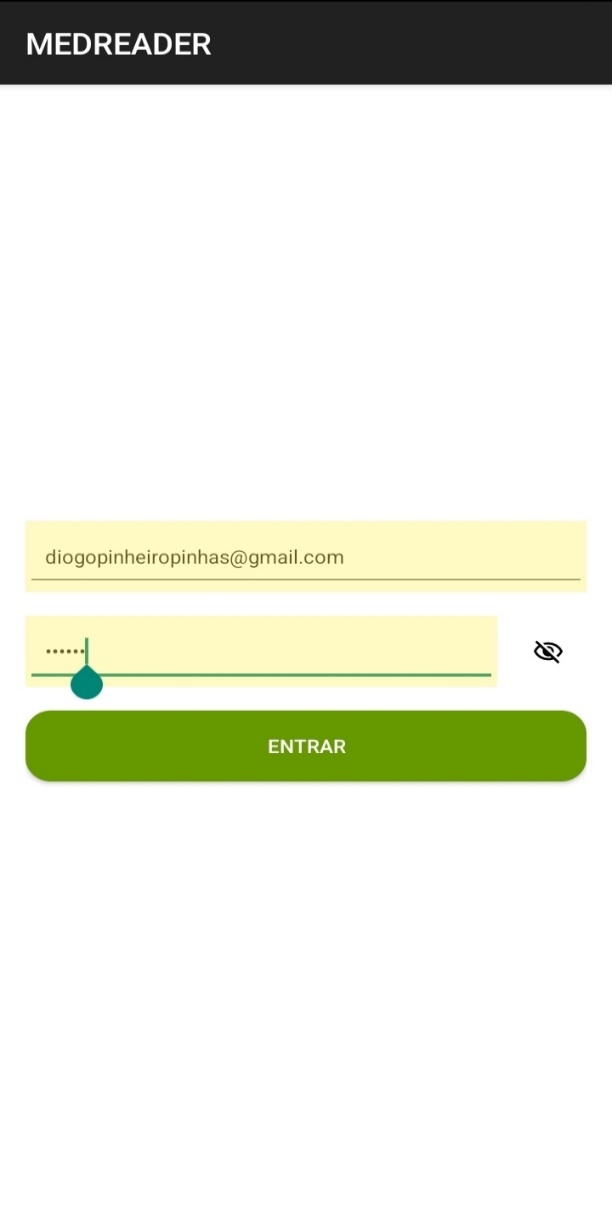


Figura - *MedReader* - Login

1. **Visualização Requerimentos**

Após o *login*, a aplicação apresenta uma lista de requerimentos como ilustrado na Figura 23. Os requerimentos são organizados por ordem de prioridade e chegada, com destaque para os requerimentos marcados como “urgente”, facilitando a identificação e o atendimento dos pedidos mais críticos.

Caso existam requerimentos urgentes na lista, a aplicação impede a seleção dos requerimentos não urgentes, como ilustrado na Figura 24, até que todos os pedidos críticos sejam atendidos. Esta funcionalidade assegura que os pedidos urgentes sejam atendidos primeiro.

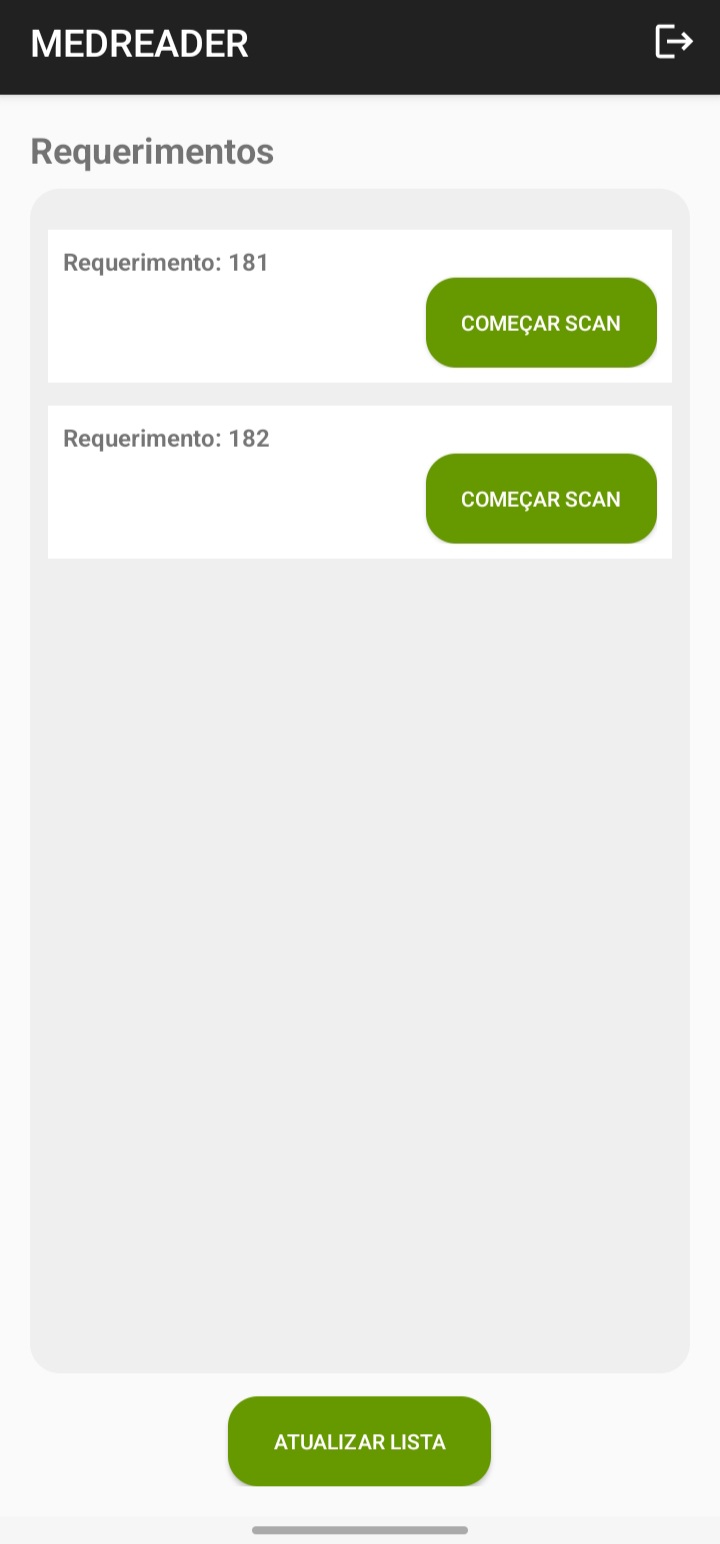
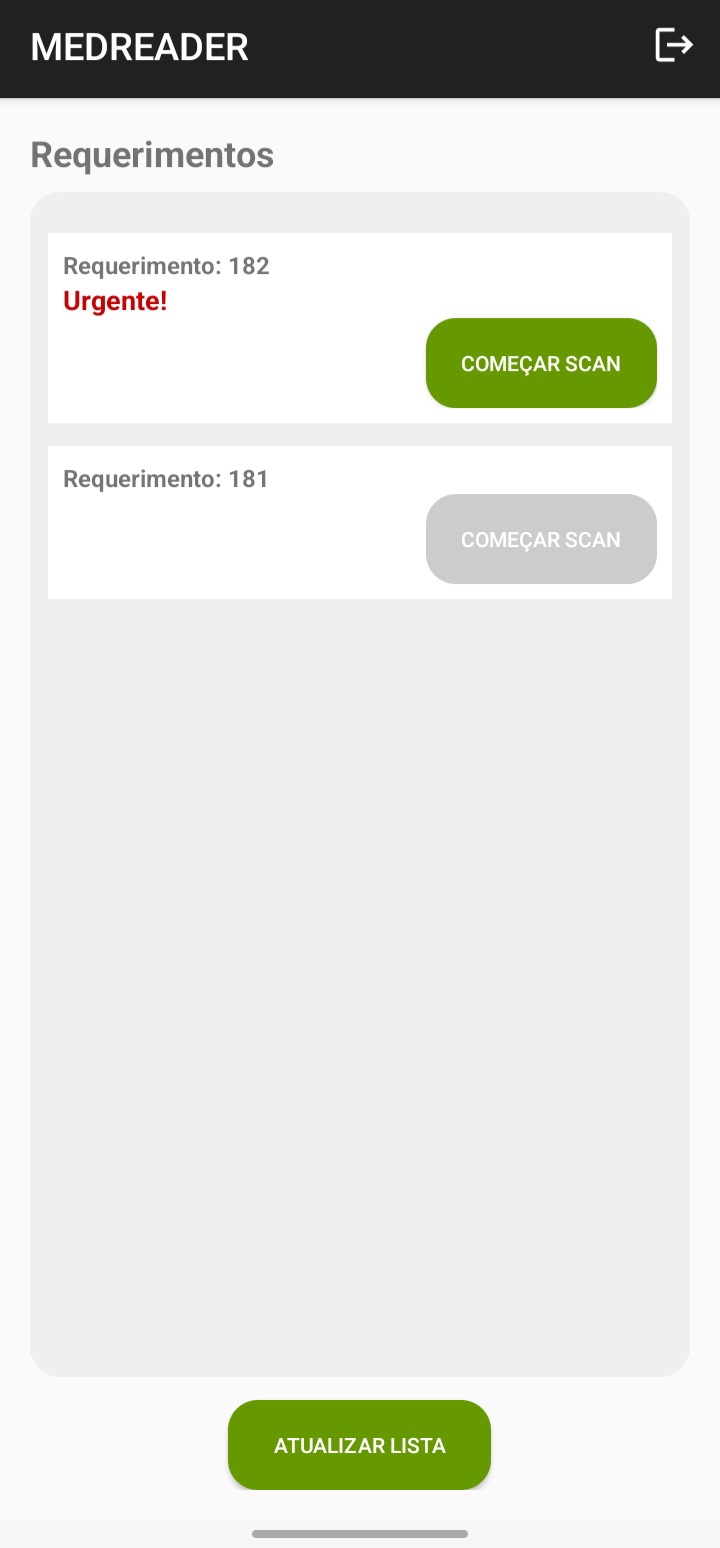


Figura - Lista Requerimentos - Não Urgentes

Figura - Lista de Requerimentos - Urgentes

1. **Seleção Requerimento**

O farmacêutico pode selecionar um requerimento específico para visualizar os consumíveis associados.

Uma imagem com texto, multimédia, captura de ecrã, software

Descrição gerada automaticamente

Figura - Consumíveis Requerimento Selecionado

1. **Leitura de Códigos Qr e/ou códigos de Barras**

Após a seleção do requerimento, são apresentados os itens pedidos dentro do requerimento, uma zona para fazer a leitura e os itens lidos pelo *scanner,* como ilustrado na Figura .

Cada item solicitado é identificado por meio de um código Qr ou código de barras. O farmacêutico realiza a leitura dos códigos com o dispositivo móvel, e a aplicação automaticamente atualiza as quantidades lidas de cada item. Caso necessário, para não estar sempre a ler os códigos quando estes são pedidos em grandes quantidades, também é possível ajustar manualmente as quantidades.



Figura - Leitura de Consumíveis

1. **Atualização *MedStock***

Após a validação dos consumíveis de cada requerimento, o utilizador é redirecionado para a página onde contém a lista dos requerimentos e as informações são enviadas para o servidor, atualizado o estado do requerimento, ilustrado na Figura 27.

Uma imagem com texto, software, Ícone de computador, computador

Descrição gerada automaticamente

Figura - Atualização Estado Requerimento - *MedStock*

1. **Comunicação *API***

A aplicação comunica com o servidor para realizar consultas sobre requerimentos e consumíveis, além de enviar dados relacionados à validação dos consumíveis.

### *MedOcorrencias*

*MedOcorrencias* é uma aplicação web, para atender as necessidades de ocorrências exteriores. A mesma tem como objetivo enviar as ocorrências de modo a serem atendidas com prioridade.

**Como Funciona?**

1. Introdução

Ao inicializar a aplicação, debate-se com uma simples introdução. Além disso, para iniciar a ocorrência, será necessário carregar no botão presente abaixo, intitulado “Entrar”.

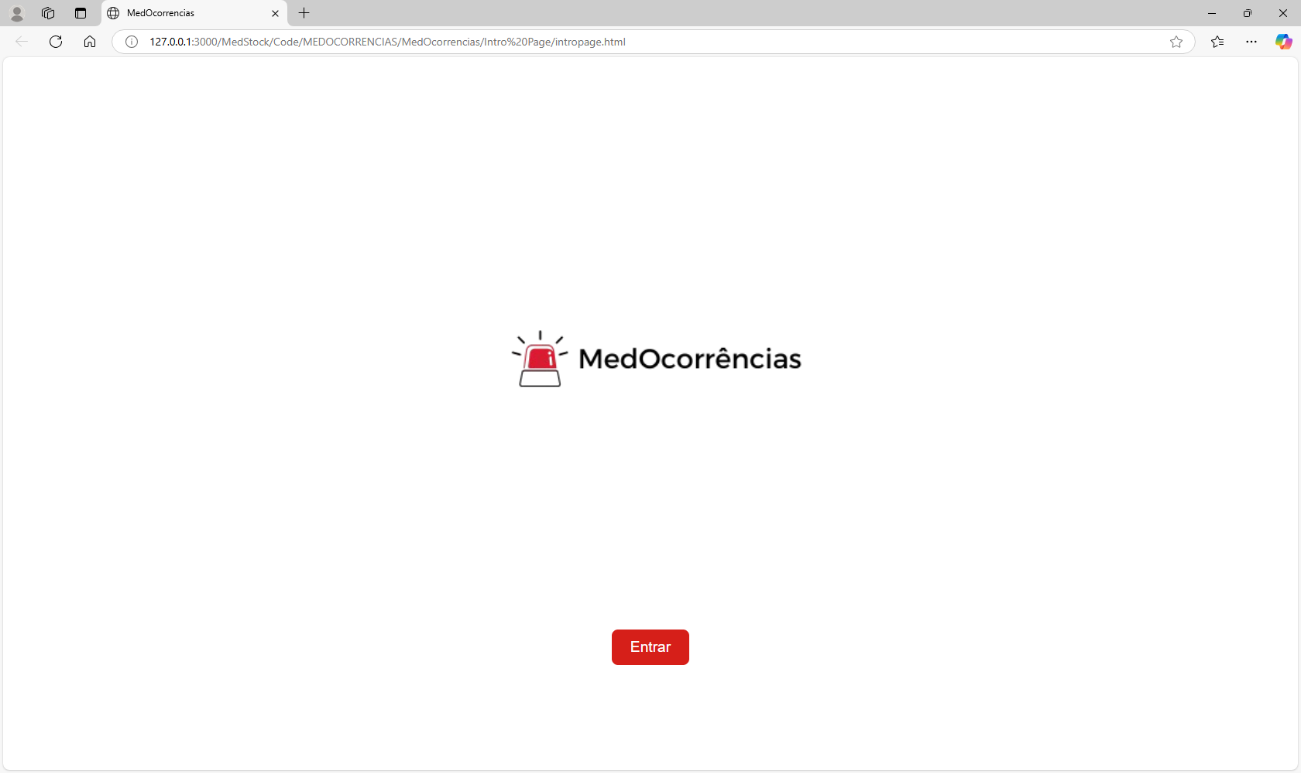


Figura - Página de Introdução

1. Login

Para aceder à aplicação, o utilizador irá efetuar o *login*, utilizando o respetivo *email* e *password*. Apenas utilizadores dos serviços externos poderão realizar ocorrências, como, por exemplo, socorristas de ambulância, paramédicos, entre outros.

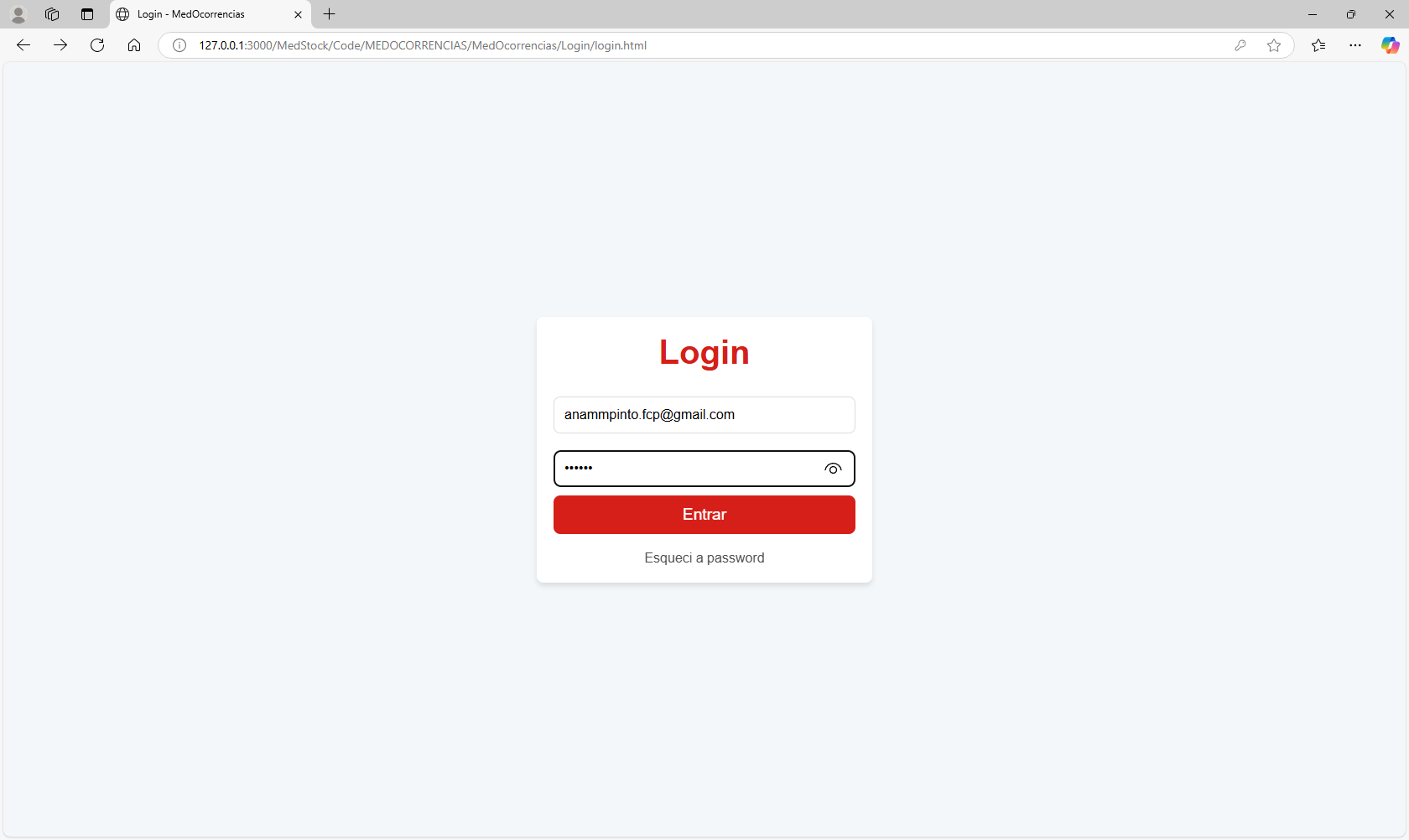


Figura - Login

1. Registar Ocorrência

Após efetuado o *login*, o utilizador poderá proceder ao preenchimento dos dados essenciais para registar a ocorrência. Consequentemente, deverá preencher o nome do respetivo paciente em análise e descrever o seu estado, como exemplificado na Figura 30.

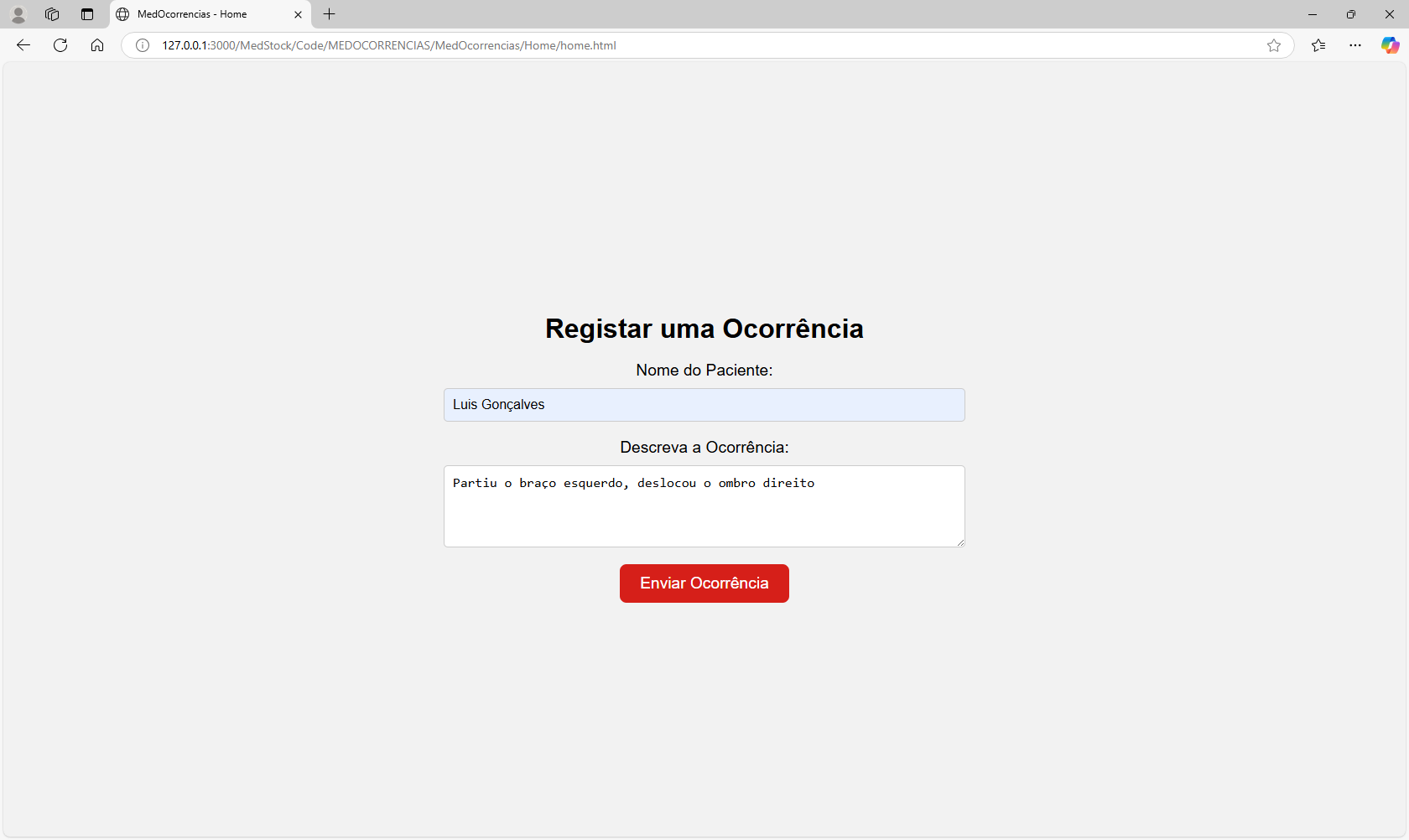


Figura - Registo de uma ocorrência

1. Ocorrência Registada com Sucesso

De seguida, o preenchimento correto dos dados pedidos para registar uma ocorrência, será necessário carregar no botão intitulado de “Enviar Ocorrência”. Como representa a Figura 31, aparecerá um *popup* com o objetivo de confirmar o envio da ocorrência.

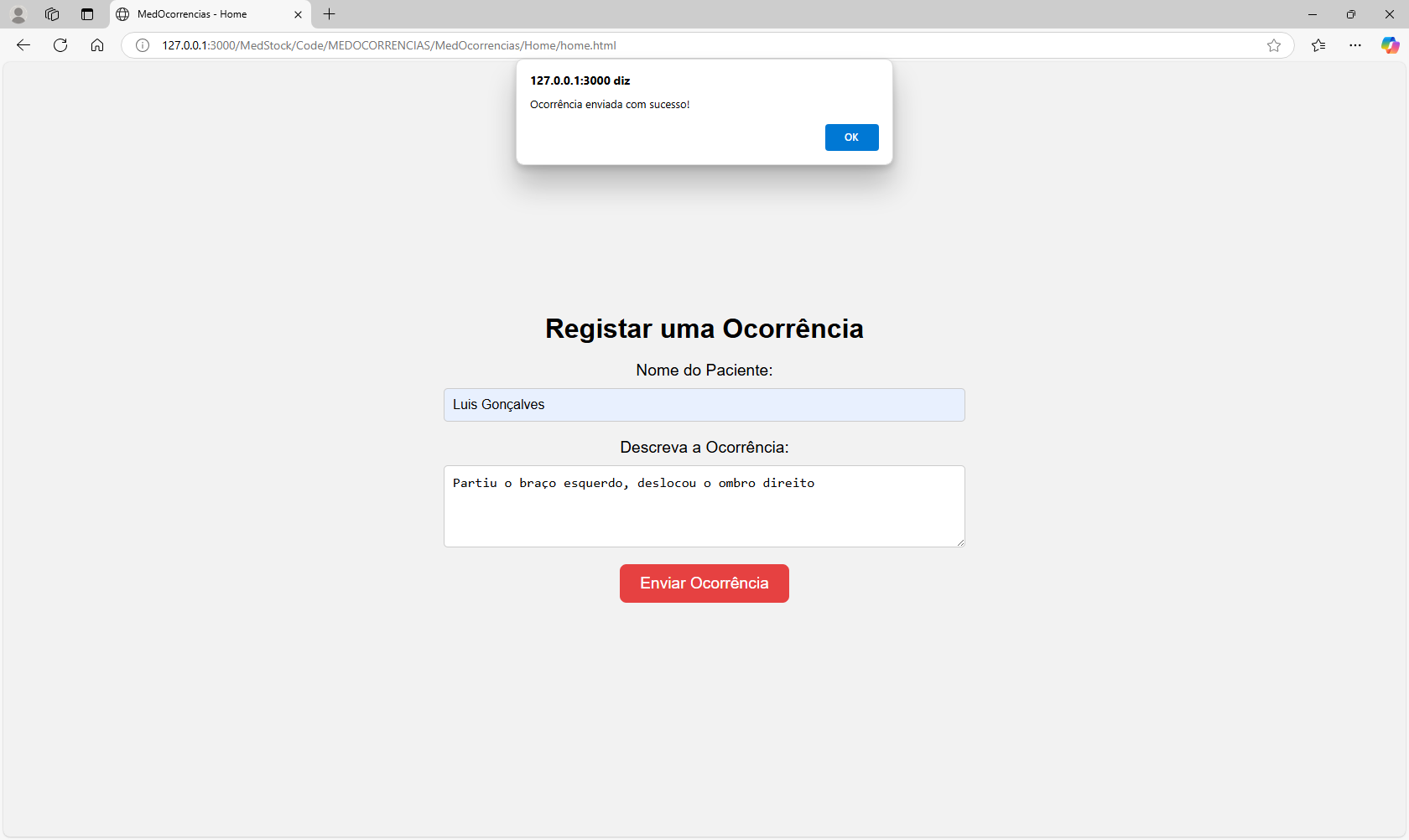


Figura - Envio da ocorrência

Por fim, como representa a Figura 32, seguidamente de uma ocorrência registada com sucesso, o pedido é apresentado na página de requerimentos. Como se pode observar, um pedido externo é sempre urgente.

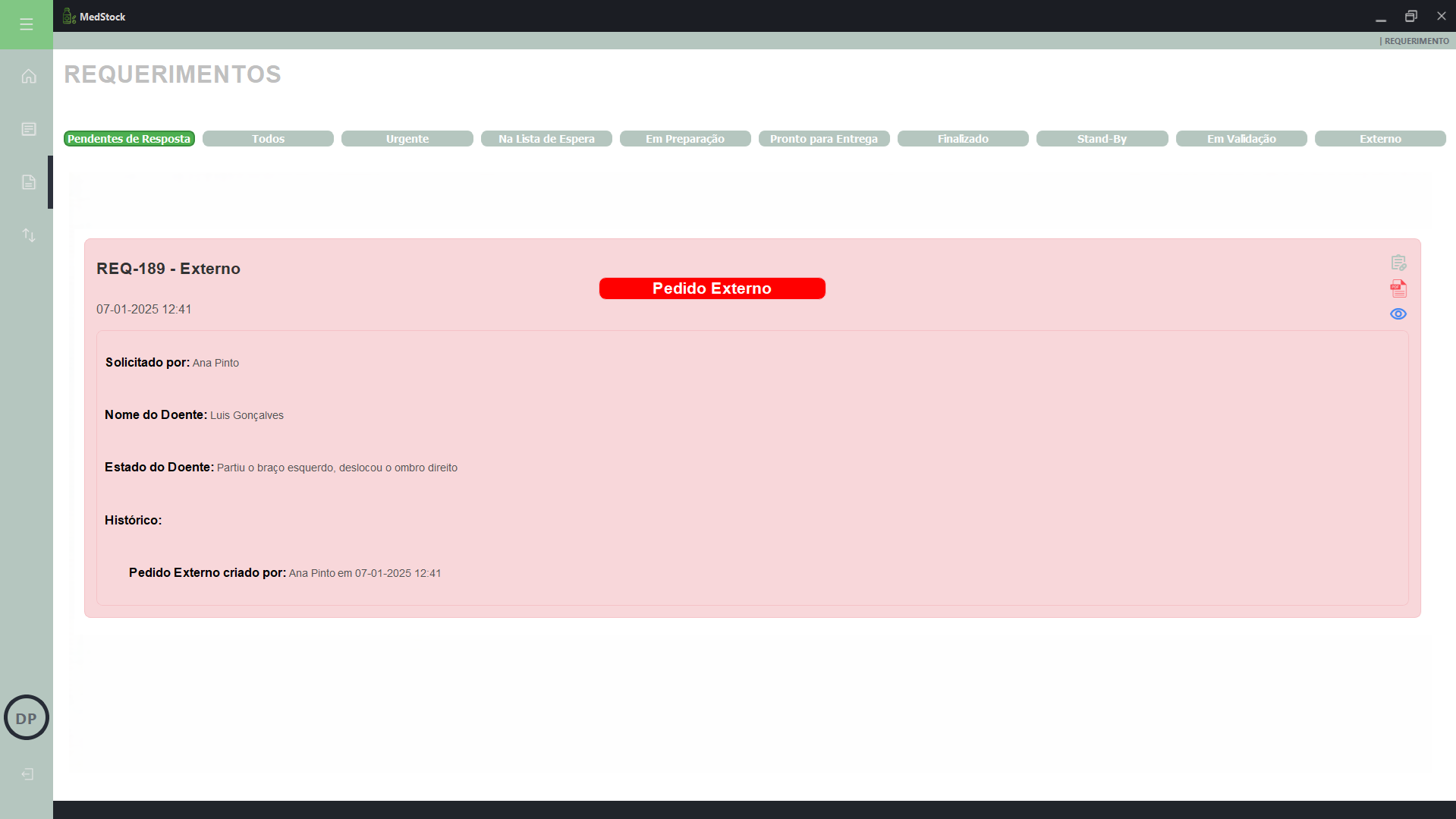


Figura - Visualização da ocorrência - *MedStock*

### *MedSupply*

# Testes

# Repositório Projeto

# Trabalho Futuro

# Conclusão

# Bibliografia

1. Álvaro da Cruz de Assis, J., & Manuel Loução de, J. (n.d.). *Universidades Lusíada*. Retrieved January 3, 2025, from http://repositorio.ulusiada.pt
2. *Cloud Application Platform | Render*. (n.d.). Retrieved January 3, 2025, from https://render.com/
3. *Como os algoritmos de otimização de estoque podem agilizar o controle de estoque?* (n.d.). Retrieved January 3, 2025, from https://www.linkedin.com/advice/3/how-can-inventory-optimization-algorithms-lgehe?lang=pt&originalSubdomain=pt
4. *Conheça o segredo para uma gestão de stocks eficiente*. (n.d.). Retrieved January 3, 2025, from https://pt.primaverabss.com/pt/blog/gestao-eficiente-*stock*s/
5. *Diagrama de caso de uso UML: O que é, como fazer e exemplos | Lucidchart*. (n.d.). Retrieved January 3, 2025, from https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml
6. *FastAPI*. (n.d.). Retrieved January 3, 2025, from https://fastapi.tiangolo.com/
7. *Gestão de armazéns com inteligência artificial: Melhorar a eficiência*. (n.d.). Retrieved January 3, 2025, from https://www.manutan.pt/blog/inteligencia-artificial-gestao-armazens-eficiencia/
8. *Home | QT-PyQt-PySide-Custom-Widgets Documentation*. (n.d.). Retrieved January 3, 2025, from https://khamisikibet.github.io/Docs-QT-PyQt-PySide-Custom-Widgets/
9. *HTML and CSS - GeeksforGeeks*. (n.d.). Retrieved January 3, 2025, from https://www.geeksforgeeks.org/html-css/
10. *KhamisiKibet/QT-PyQt-PySide-Custom-Widgets: Awesome custom widgets made for QT Desktop Applications. Simplify your UI development process. These widgets can be used in QT Designer then imported to PySide code.* (n.d.). Retrieved January 3, 2025, from https://github.com/KhamisiKibet/QT-PyQt-PySide-Custom-Widgets
11. *Kotlin Docs | Kotlin Documentation*. (n.d.). Retrieved January 3, 2025, from https://kotlinlang.org/docs/home.html
12. *Matriz De Poder E Interesse — Domine A Gestão De Stakeholders*. (n.d.). Retrieved January 3, 2025, from https://blog.adapt.works/matriz-de-poder-e-interesse-gestao-de-stakeholders/
13. *O que é BPMN? Como fazer o Diagrama? Modelos e Exemplos*. (n.d.). Retrieved January 3, 2025, from https://miro.com/pt/diagrama/o-que-e-bpmn/
14. *O que é um diagrama de contexto? (e como você pode criar um)*. (n.d.). Retrieved January 3, 2025, from https://pt.venngage.com/blog/diagrama-de-contexto/
15. *O que é um diagrama de máquina de estados? | Lucidchart*. (n.d.). Retrieved January 3, 2025, from https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-maquina-de-estados-uml
16. *O que é um diagrama entidade relacionamento? | Lucidchart*. (n.d.). Retrieved January 3, 2025, from https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento
17. *O que são Requisitos Funcionais: Exemplos, Definição, Guia Completo - Visure Solutions*. (n.d.). Retrieved January 3, 2025, from https://visuresolutions.com/pt/blog/requisitos-funcionais/
18. *PyQt - Python Wiki*. (n.d.). Retrieved January 3, 2025, from https://wiki.python.org/moin/PyQt
19. *Python | Introduction to PyQt5 - GeeksforGeeks*. (n.d.). Retrieved January 3, 2025, from https://www.geeksforgeeks.org/python-introduction-to-pyqt5/
20. *Qt for Python*. (n.d.). Retrieved January 3, 2025, from https://doc.qt.io/qtforpython-6/
21. *React Reference Overview – React*. (n.d.). Retrieved January 4, 2025, from https://react.dev/reference/react
22. *Wanderson-Magalhaes/Simple\_PySide\_Base*. (n.d.). Retrieved January 3, 2025, from https://github.com/Wanderson-Magalhaes/Simple\_PySide\_Base
23. *Welcome to Flask — Flask Documentation (3.1.x)*. (n.d.). Retrieved January 4, 2025, from https://flask.palletsprojects.com/en/stable/#user-s-guide