Escola Superior
de Tecnologia
e Gestão
Politécnico de Coimbra

Relatório de Projeto

Programação

Avaliação Periódica

Autores:

Guilherme Gonçalves a2022156457@alunos.estgoh.ipc.pt
Hugo Pais a2022129956@alunos.estgoh.ipc.pt

Data: Janeiro 2024

Resumo

Este projeto implementa uma aplicação para gestão da atividade farmacêutica em Java, recorrendo ao paradigma de Programação Orientada a Objetos (POO). O objetivo principal é desenvolver uma solução abrangente para a gestão de medicamentos, serviços e utilizadores (e.g., clientes, farmacêuticos e gestores) em ambiente farmacêutico, utilizando estruturas de dados eficientes e ficheiros para armazenamento dos dados. A metodologia envolveu a aplicação de conceitos de POO e uma articulação entre fases planeadas para o desenvolvimento e consistência do projeto. Os resultados destacam a eficácia na gestão de utilizadores, serviços e medicamentos, utilizando estruturas como vetores e objetos para articular os dados. A nível de conclusão ressalta-se a robustez e pertinência do projeto desenvolvido. O trabalho futuro irá incidir sobre atualizações e melhorias contínuas ao trabalho aqui explanado.

Palavras-chave

Aplicação, Farmácia, Gestão, Medicamentos, Programação, Serviços, Utilizadores

Índice

Lista de Tabelas

Lista de Acrónimos

ER	Modelo Entidade-Relacionamento
ESTGOH	Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
POO	Programação Orientada a Objetos

1. Introdução

O presente projeto pressupõe o desenvolvimento de uma aplicação que permita a gestão de medicamentos e serviços disponíveis numa farmácia. A correta operacionalização irá possibilitar a transmissão de dados entre clientes, farmacêuticos e gestores, permitindo o acompanhamento e concretização de encomendas, a consulta de medicamentos, a gestão de utilizadores, entre muitas outras funcionalidades.

A aplicação a desenvolver recorre à linguagem Java, seguindo o paradigma de Programação Orientada a Objetos. A implementação recorre aos conceitos abordados na Unidade Curricular de Programação, na licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH), Unidade Orgânica, do Instituto Politécnico de Coimbra. Concretamente, foram integrados neste projeto competências relativas a objetos e classes, estrutura de dados estáticas e dinâmicas, tratamento de exceções, heranças e polimorfismo e manipulação de ficheiros para armazenamento persistente dos dados (Veloso, 2023).

A interface para interação com o utilizador é a parte central da aplicação, onde todo o menu e suas funcionalidades estão dispostas. O projeto foi planeado, estruturado, concebido, testado e distribuído considerando os requisitos iniciais, os tempos estabelecidos e a pertinência, oportunidade e necessidade que existe no mercado.

Os objetivos considerados para desenvolvimento do projeto, não só contemplam o levantamento e briefing da empresa (cliente), como pretendem garantir competências de programação em JAVA por parte dos desenvolvedores.

Nos próximos capítulos serão dadas mais informações relativamente ao projeto desenvolvido e suas características.

2. Estado da Arte

O desenvolvimento de aplicações relacionadas com a gestão de farmácias e serviços de saúde foi ao longo dos últimos anos, uma área de interesse. A otimização de processos e a melhoria da eficiência organizacional foram a alavanca necessária ao desenvolvimento de soluções que conferissem mais valias ao mercado (Jordan, Neves & Rodrigues, 2015).

Neste sentido, os Sistemas de Controlo e Gestão foram-se desenvolvendo, permitindo uma otimizada organização e articulação dos dados e uma reflexão e revisão metodológica fomentadoras e promotoras das melhores estratégias e práticas organizacionais. Monitorizar e desenvolver soluções que não só cumpram uma ótica fiscalizadora, mas essencialmente de controlo, planeamento e gestão participativa entre as diferentes hierarquias e os objetivos individuais e coletivos dos Stakeholders foi o foco de ação (e.g., Dupuiy e Roland, 1999; Jordan, Neves & Rodrigues, 2015).

Em 2020 com a crise pandémica vivenciada a nível global e as medidas de distanciamento social a serem introduzidas em inúmeros países houve um aumento do recurso à internet por empresas, instituições e cidadãos de modo a acederem a serviços e a colmatarem necessidades. Este contexto transformou o mercado tornando-o mais digital e fomentou o desenvolvimento tecnológico pelas organizações das mais diversas áreas (Comissão Europeia, 2024).

Torna-se evidente com o breve enquadramento realizado que há empresas que tentando responder às necessidades do mercado tentem proativamente dispor de soluções que se configurem como vantagens de posicionamento e desenvolvimento do negócio. A pertinência e execução deste projeto assenta na premência da empresa cliente em proceder à digitalização e inovação tecnológica do seu negócio e aproveitando a comparticipação do projeto por fundos comunitários, concretamente, Portugal 2030.

3. Objetivos e Metodologias

Os objetivos consignados ao projeto balizam a metodologia e ações desenvolvidas e, indiretamente, estruturam e planificam a conceção, codificação e implementação do trabalho. A execução foi alicerçada no briefing dado pelo cliente e nos requisitos por ele considerados e seguidamente apresentados:

Gestão de acesso e utilizadores
[R1] Permitir os utilizadores registarem-se e autenticarem-se na aplicação.
[R2] Permitir o acesso à aplicação por 3 tipos de utilizadores: gestores, farmacêuticos e clientes.
[R3] Os utilizadores são caracterizados pelos atributos login, password, nome, estado (ativo/inativo), email e tipo (gestores, farmacêuticos ou clientes).
[R4] Cada utilizador apenas pode alterar a sua própria informação, não podendo alterar ou visualizar dados de outros utilizadores, ou criar utilizadores.
[R5] O login e email devem ser únicos.
[R6] Os clientes e farmacêuticos caracterizam-se adicionalmente por número de identificação fiscal (NIF), morada e contacto telefónico. O NIF e contacto telefónico são únicos.
[R7] Caso não existam utilizadores criados, a aplicação deve solicitar a criação de um utilizador (gestor).
[R8] Após a autenticação, a aplicação deve apresentar a mensagem: "Bem-vindo [nome utilizador]".
[R9] Quando a aplicação estiver a encerrar, deve apresentar a mensagem: "Adeus [nome utilizador]".

Funções
[R10] Os gestores aprovam os pedidos de registo dos utilizadores. Todos os pedidos devem ser aprovados antes de poderem ser usados para autenticação.
[R11] Os gestores aprovam e encerram os pedidos de serviço (encomenda), bem como associam um farmacêutico responsável a cada serviço.
[R12] Os clientes solicitam pedidos de serviço, que devem ser aprovadas pelos gestores.
[R13] Os farmacêuticos introduzem medicamentos, categorias e excipientes e gerem os serviços.
[R14] Os gestores podem consultar todos os pedidos de serviços realizados.
[R15] Os farmacêuticos apenas podem consultar os seus serviços.
[R16] Os clientes apenas podem consultar os seus serviços.

Ações
[R17] O processo de serviço inicia com o cliente a solicitar uma encomenda, indicando os medicamentos pretendidos ou as componentes ativas (neste caso o farmacêutico é que escolhe o medicamento adequado).
[R18] O gestor aprova o serviço e associa um farmacêutico responsável.
[R19] Após ser designado para um serviço, o farmacêutico responsável regista a informação do processo, nomeadamente pode criar sub-tarefas e associar outros farmacêuticos adicionais às sub-tarefas, e gere o processo até à sua conclusão.
[R20] Um serviço (encomenda), além de ter associado um farmacêutico responsável, inclui a listagem dos produtos incluídos (medicamentos), o valor total da venda, uma data em que o serviço foi requisitado, tempo despendido no serviço (em horas) e uma descrição.
[R21] Cada serviço tem adicionalmente um tipo (se urgente ou normal) e um estado que indica:
a) se o serviço está pendente (cliente realizou o pedido, mas ainda não foi aceite pelo gestor);
b) se a serviço foi aceite pela farmácia (gestor aceita o serviço);
c) se está a decorrer (farmacêutico responsável inicia o processo);
d) se já foi concluída (farmacêutico responsável termina o processo);

e) ou se o processo já foi encerrado (gestor encerra o processo).

[R22] Para garantir uma rápida identificação, cada serviço possui um código único sequencial.
[R23] A aplicação deve permitir inserir e gerir informação dos medicamentos. Os medicamentos são caracterizados por uma designação (ou nome), marca (laboratório de fabrico), lote, componente ativo, dosagem, quantidade em stock, preço de venda, ano de fabrico, se necessita de autorização médica e se o medicamento é genérico.
[R24] Cada componente ativo é caracterizada pela sua designação, código e quantidade.
[R25] Cada medicamento deve ter associado uma lista de 5 excipientes (ou menos).
[R26] Cada excipiente é caracterizado pela sua designação, classificação INFARMED e quantidade.
[R27] Cada medicamento deve apresentar uma lista de 3 categorias (ou menos), e.g. analgésico, antibiótico, anti-inflamatório, antipirético.
[R28] Cada categoria é caracterizada pela sua designação, classificação INFARMED, código e fornecedor.
O atributo código deve ser único.
[R29] É possível alterar a quantidade em stock de um medicamento.
[R30] Sempre que um medicamento é vendido, a quantidade em stock deve ser atualizada.
[R31] Após um serviço ser concluído pelo farmacêutico principal, o gestor pode encerrar o pedido.

Listagens e pesquisas

[R32] Deve ser possível ordenar utilizadores por ordem alfabética do nome.
[R33] Deve ser possível ordenar medicamentos por designação.
[R34] Deve ser possível listar todos os utilizadores.
[R35] Deve ser possível listar utilizadores por tipo (gestores, farmacêuticos ou clientes).
[R36] Deve ser possível listar todos os medicamentos.
[R37] Deve ser possível listar todos os serviços associados a um cliente.
[R38] Deve ser possível listar todos os serviços por estado (a decorrer ou encerrados).
[R39] Deve ser possível pesquisar utilizadores por login ou nome.
[R40] Deve ser possível pesquisar medicamentos por designação.
[R41] Deve ser possível pesquisar medicamentos por categoria.
[R42] Deve ser possível pesquisar medicamentos que possuem uma determinada componente ativa.
[R43] Deve ser possível pesquisar medicamentos que sejam (ou não) genéricos.
[R44] Deve ser possível pesquisar medicamentos com uma quantidade de stock abaixo de um determinado limite (introduzido pelo utilizador no momento de pesquisa).
[R45] Deve ser possível pesquisar serviços por código.
[R46] Deve ser possível pesquisar serviços com tempo despendido superior a um determinado limite (introduzido pelo utilizador no momento de pesquisa).
[R47] Deve ser possível realizar pesquisas avançadas, ou seja, apresentar todos os registos que apresentem um termo de pesquisa, mesmo que parcialmente (e.g. termo de pesquisa “Ana” deve apresentar como resultado “Ana Sousa”, “Ana Silva” e “Anabela”).
[R48] Os clientes podem listar e pesquisar os serviços que realizaram.
[R49] Os farmacêuticos podem listar e pesquisar os serviços que processaram.
[R50] Os gestores podem listar e pesquisar todos os serviços. Manipulação e armazenamento de dados persistente
[R51] O acesso à aplicação deve ser restringido com credenciais (login/password), informação que deverá ser armazenada num ficheiro de texto “credenciais_acesso.txt”.
[R52] Durante o encerramento da aplicação, os dados devem ser automaticamente guardados num ficheiro de objetos “dados_apl.dat”.
[R53] A aplicação, no arranque, deve automaticamente ler os dados do ficheiro de objetos “dados_apl.dat”, caso este exista, e informar o utilizador que os dados foram lidos com sucesso. Caso o ficheiro esteja indisponível deve surgir uma mensagem a informar a situação.

Interação com o utilizador

[R54] Disponibilizar uma interface em modo texto onde o utilizador possa interagir e controlar a aplicação.

Monitorização de acessos
[R55] Todas as ações dos utilizadores deverão ser guardadas num ficheiro de texto denominado "log.txt".
Este ficheiro deverá ser escrito de forma sequencial, apresentando as ações mais recentes no início do ficheiro. As entradas deverão ter o seguinte formato: <utilizador> <ação>.
[R56] Deve ser possível consultar o log de ações através da aplicação.
[R57] Deverá ser registado o número total (até ao momento) de execuções do sistema e o login do último utilizador que acedeu à aplicação. Esta informação deverá ser guardada num ficheiro de objetos e atualizada sempre que a aplicação é executada, apresentando os dados ao utilizador. Quando a aplicação é terminada esta informação deve ser novamente armazenada no ficheiro, sendo este denominado "info_sistema.dat".

Programação Orientada a Objetos
[R58] A aplicação deve estar corretamente estruturada, tendo em conta o paradigma Orientado a Objetos e recorrendo à linguagem Java.
[R59] Implemente as estruturas de armazenamento necessárias, procurando otimizar os recursos utilizados.

Validação de dados e notificações
[R60] Valide todas as leituras de dados do utilizador (e.g. verifique se os nomes são únicos).
[R61] Sempre que necessário, apresentar ao utilizador mensagens informativas adequadas. Quando um utilizador realizar uma ação sobre a aplicação, esta deve informar se ação foi realizada com sucesso ou insucesso.

O objetivo central impele à criação de uma aplicação que cumpra os pressupostos da empresa cliente e seja uma mais valia para implementação aos negócios de farmácia.
A metodologia implementada teve várias etapas de execução. A primeira visou a construção das classes, objetos, atributos necessários. A segunda, a definição dos métodos necessários e a articulação entre eles. A terceira, o inicio da conceção do menu (e respetiva class main) e ajustes necessários ao seu correto funcionamento. A última fase foi testagens e correções para posterior distribuição. Foram alocados dois desenvolvedores ao projeto e o trabalho foi igualmente repartido entre eles.

3.1. Ferramentas e Tecnologias

A constituição deste projeto recorreu à utilização do Visual Studio Code (VS Code) uma aplicação que possibilita um Ambiente de Desenvolvimento Integrado (do inglês IDE), gratuito e de código aberto desenvolvido pela Microsoft. Projetado para ser leve, rápido e altamente customizável, é adequado para uma ampla variedade de linguagens de programação.

O VS Code é conhecido pela sua interface e pelo suporte de uma variedade de extensões que podem ser instaladas para adicionar funcionalidades específicas. Algumas das características mais relevantes são: edição de texto avançada, depuração e criação de código. Os compiladores e as ferramentas de conclusão de código são outros dos recursos disponíveis para tornar mais eficiente o desenvolvimento de software (Visual Studio, 2023).

A escolha desta ferramenta cingiu-se a dois princípios base: familiarização e oportunidade, uma vez que é recorrente o seu uso noutros projetos, gratuito e de fácil manuseamento.

Para um progresso mais célere, recorreu-se à plataforma online Replit, que também oferece um IDE para programação e permite a colaboração e simplificação do processo de codificação.

3.2. Planeamento

O planeamento alicerçado nas etapas definidas na metodologia supriu parte dos requisitos propostos e expectáveis nesta etapa de desenvolvimento do projeto. Das etapas inicialmente definidas, considerou-se as duas primeiras (i.e., criação das classes de suporte ao projeto e implementação dos métodos que permitem a manipulação de dados) como a base necessária para a realização do trabalho até à entrega do relatório intermédio. Posteriormente respondeu-se às restantes fases e cumpriu-se a calendarização com uma ligeira discrepância.

Tabela 1 - Planeamento Previsto

	29/11	06/12	13/12	16/12	30/12	15/01	Previsão Tempo
1ª Fase							15 horas
2ª Fase							30 horas
3ª Fase							40 horas
4ª Fase							35 horas
Relatório							8 horas

A realização do projeto, embora executada com sucesso, teve um contratempo na fase final por limitações na escrita de objetos em ficheiro. Este percalço levou a prejuízo no requisito 56 que não foi de todo concretizado e nos requisitos 55 e 57 que foram parcialmente implementados. A mudança de estratégia a meio do desenvolvimento do projeto permitiu uma melhor operacionalização do tempo despendido, concretamente, no início o projeto começou por ser executado sem a divisão de tarefas pelos elementos da equipa e sem recurso a uma plataforma que facilitasse o trabalho simultâneo. Porém rapidamente se considerou como benéfico recorrer à plataforma Replit. Esta ferramenta online proporcionou a divisão das tarefas pelos membros do grupo, considerando as suas funcionalidades (e.g., trabalho simultâneo, edição em tempo real).

Em suma o planeamento no global cumpriu-se e foi o adequado dada a extensão do projeto.

4. Trabalho Desenvolvido

Neste capítulo será apresentada uma visão mais ao detalhe daquilo que realmente foi desenvolvido no projeto, descrevendo desde aquilo que foi realizado em termos de classes e métodos, e qual o pensamento para a sua implementação, até ao objetivo de se obterem os respetivos resultados previstos.

4.1. Requisitos Implementados

Os requisitos apresentados no capítulo objetivos e metodologias foram todos implementados, com exceção do [56] que por limitações de tempo e não ser essencial para o cliente foi aditado a uma segunda fase de desenvolvimento (e de atualização) da plataforma. O requisito [55] e [57] encontram-se parcialmente implementados uma vez que o primeiro não cumpre a ordem de escrita estabelecida e o segundo não apresenta a informação ao utilizador. Não obstante, importa referir esta manipulação tão especifica teve suporte em pesquisa online (e.g., Simplilearn, 2024), uma vez que não estava a ser possível concretizar só com as informações de aula.

4.2. Classes

O projeto foi estruturado com base em vários ficheiros, no qual, cada ficheiro, de forma a estarem mais organizados e de melhor perceção, continha uma classe referente a uma certa funcionalidade. De seguida estarão representadas todas as classes utilizadas no mesmo, com os principais métodos implementados no trabalho, explicados com base na sua função.

Classe "Main.java"

Esta é a principal classe do projeto, pois contém o método "main" que é o principal responsável para a execução do programa. Dentro do método "main", apenas estarão as funções para apresentar o menu ao utilizador e a função para executar a opção correspondente à que o utilizador escolheu. Cada opção depende da escolha do utilizador e será chamada a devida função. Por exemplo, a opção 1 serve para registar um novo utilizador, lembrando que caso a aplicação não tenha qualquer utilizador criado anteriormente, este primeiro será sempre do tipo "Gestor". Dentro das opções estarão representados os vários métodos, consoante a lógica da opção, das outras classes também descritas de seguida. Importa destacar que houve necessidade de consulta para criação da função que permitisse limpar o terminal (Replit, 2024).

Classe "Categoria.java"

Nesta classe estarão representadas as variáveis a ser utilizadas no objeto, sendo que estas estarão inseridas no medicamento, o seu devido construtor e um método de leitura de um atributo.

Classe "ComponenteAtiva.java"

Nesta classe são representadas as componentes ativas de um medicamento, o devido construtor e um método "get" para obter a informação de um atributo.

Classe "Excipiente.java"

Esta classe é parecida à anterior, "ComponenteAtiva.java", com os atributos e o seu construtor. Esta será usada nos medicamentos, caso seja necessário colocar o excipiente.

Classe "Clientes.java"

Esta classe é o único ponto do projeto que se recorreu à herança, sendo esta a classe filha, associada à superclasse "Utilizadores". Foi necessário recorrer à herança visto que nesta classe se enquadram os tipos, "Clientes" e "Farmacêuticos", que, ao contrário dos "Gestores", estes apresentam três atributos adicionais (i.e., nif, morada e telefone). Tal como nas classes anteriores, esta também apresentada um construtor, de forma a inicial o objeto, as operações que permitem a leitura e a atualização de certos atributos, e também um método "toString" de forma a utilizar-se no retorno de certas funções, em que é necessário obter uma frase com certos dados do objeto.

Classe "Utilizadores.java"

Esta classe, tal como dito na classe anterior, é a superclasse, visto que as outras classes herdam os atributos desta. Representa um utilizador, e contém atributos relativos a este, incluindo informações como login, password, nome, email, entre outros. Tal como as outras classes, esta também contém, formas de obter os atributos, alterar e uma função "toString". Também tem, adicionalmente, um método "compareTo" que será usado para uma possível ordenação dos utilizadores com base no nome.

Classe "Medicamentos.java"

Esta classe modela um medicamento no sistema, com as devidas informações do medicamento assim, juntado algumas informações sobre certas classes vistas anteriormente. Segue a lógica de algumas classes já vistas, métodos para obter

informações de um certo atributo, métodos para alterar o conteúdo de certos atributos e um método "toString".

Classe "Servicos.java"

Representa a informação dos serviços e também segue aquilo que foi feito nas outras classes.

Classe "GereUtilizadores"

Nesta classe é onde estão todos os métodos que permitem gerir as várias operações relacionadas aos utilizadores. Alguns dos métodos criados e implementados foram por exemplo:

- "adicionarUtilizador" – método que permite adicionar o novo utilizador à lista que contém todos os utilizadores;
- "verificarUtilizador" – método implementado no momento da autenticação, que verifica se o utilizador com um determinado login e password inseridos pelo utilizador, existe na lista. Se o utilizador existir, é devolvido esse mesmo utilizador que fica com acesso às diferentes opções tendo em conta o seu tipo. Caso não exista, é retornado ao menu inicial que qualquer pessoa sem a correta autenticação tem acesso.
- "listarUtilizadoresPendentes" – método que retorna a listagem com todos os utilizadores pendentes para o gestor aprovar.

Classe "GereMedicamentos"

Na classe "GereMedicamentos" estão implementados os métodos capazes de gerir as várias operações relacionadas aos medicamentos. Alguns dos métodos criados e implementados foram por exemplo:

- "adicionarMedicamento" – método que permite adicionar um novo medicamento à lista que contém todos os medicamentos;
- "ordenar" – método que permite, juntamente com o método "compareTo" da classe "Medicamentos", ordenar os medicamentos por ordem alfabética do nome;
- "listarMedicamentos" – método que retorna a listagem com todos os medicamentos disponíveis.

Classe "GereServicos"

Classe responsável por gerir, através dos métodos criados nesta, os vários serviços da aplicação. Alguns dos métodos criados e implementados foram por exemplo:

- "adicionarServico" – método que permite adicionar um novo serviço à lista que contém todos os serviços;
- "listarServicos" – método que retorna a listagem com todos os servicos disponíveis consoante o tipo de utilizador. Por exemplo, se for um gestor, este tem acesso à listagem de todos os serviços, sejam eles pendente ou até concluídos. No caso dos clientes ou farmacêuticos, estes só conseguirão visualizar os pedidos que realizaram e caso estes estejam pelo menos a decorrer (estado 3) até poderem estar concluídos (estado 5);
- "listarServicosPendentes" – método que retorna a listagem de todos os serviços que ainda se encontrem em estado pendente, estado 1, para posteriormente o gestor poder aprovar.

Classe "Logger.java"

Esta classe é o que vai permitir escrever no ficheiro "log.txt" o utilizador que realizou uma determinada ação, juntamente com a data em que foi realizado o evento, utilizando o método "logAcao".

Classe "SistemaInfo.java"

Através desta classe é possível registar o número de vezes de execuções do sistema e o login do último utilizador, até ao momento.

Classe "ManipulaFicheirosTexto.java"

É através desta classe que é possível manipularmos ficheiros de texto, como a sua abertura, escrita e o seu encerramento. Alguns dos métodos criados e implementados foram por exemplo:

- "abrirFicheiroLeitura" – método que permite abrir um determinado ficheiro de texto para a sua leitura;
- "adicionarCredenciais" – método para se conseguir realizar o requisito [51], ou seja, neste caso guardamos os dados de autenticação no ficheiro "credenciais.txt";

Classe "ManipulaFicheirosObjectos.java"

Talvez um dos ficheiros mais importantes no trabalho, pois é a partir deste ficheiro que se consegue guardar todos os objetos que se tem atualmente no sistema, entre execuções. Alguns dos métodos criados e implementados foram por exemplo:

- "abrirFicheiroLeitura" – método que permite abrir um determinado ficheiro de objetos para a sua leitura;

- "lerFicheirosUtilizadores" – método para ser possível ter acesso aos utilizadores inseridos anteriormente;

Para uma análise mais simples e relativa às classes principais destaca -se o seguinte diagrama que relaciona as classes entre si:

Class GereUtilizadores

```
private Vector<Utilizadores> lista;

Ex. Métodos
listarUtilizadoresPorTipo()
ordenarUtilizadores()
aprovarUtilizador()
verificaLogin()
verificaNif()
…
```

Class Clientes (Classe Filha)

```
private int nif;
private String morada;
private int telefone;

Ex. Métodos
getMorada()
setNif()
…
```

(Relativa a clientes e Farmacêuticos)

Class GereServicos

```
private Vector<Servicos> listaServicos;

Ex. Métodos
adicionarServico()
listarServicos()
atribuirFarmaceutico()
listarServicosEstadoConcluido()
pesquisarServicosPorCodigoUtilizador()
alterarTempoServico()
…
```

Class GereMedicamentos

```
private Vector<Medicamentos>
listaMedicamentos;

Ex. Métodos
adicionarMedicamento()
ordenar()
listarMedicamentos()
pesquisarPorDesignacao()
pesquisarPorComponenteAtiva()
pesquisaMedicamentoGenerico()
pesquisaPorStock()
alterarStock()
…
```

Class Utilizadores (SuperClasse)

```
protected String login;
protected String password;
protected String nome;
protected boolean estado;
protected String email;
protected int tipo;
protected int id;

Ex. Métodos
getNome()
setEstado()
…
```

Class Servicos

```
private int idUtilizador;
private int idFarmaceutico;
private Vector<Medicamentos>
listaMedicamentos;
private float precoTotal;
private String data;
private int tempoServico;
private String descricao;
private boolean urgencia;
private int estadoProcesso;
private int codigoServico;

Ex. Métodos
getListaMedicamentos()
setTempoServico()
…
```

Class Medicamentos

```
private String nomeMedicamento;
private String marca;
private String lote;
private ComponenteAtiva
componenteAtiva;
private int dosagem;
private int stock;
private float precoMedicamento;
private int anoFabrico;
private boolean autorizacaoMedica;
private boolean medicamentoGenerico;

Ex. Métodos
setStock()
getComponenteAtiva()
addExcipiente()
addCategoria()
…
```

4.3. Algoritmos

Para a realização do projeto, foram necessários implementar alguns algoritmos essenciais para o correto funcionamento do mesmo, tendo em atenção as várias operações de gestão. Alguns dos principais algoritmos implementados no projeto foram os seguintes:

- O algoritmo de autenticação que faz verificação das credenciais fornecidas pelo utilizador, garantindo que estas estão corretas;
- A aprovação de registos, no qual o gestor tem a oportunidade de aprovar os registos de novos utilizadores, validando informações;
- A gestão de serviços, no qual é necessário a criação, a aprovação e a execução de serviços ou inicialmente dos pedidos, envolvendo os gestores, farmacêuticos e os clientes, de forma a garantir uma maior eficiência nas suas operações;
- A gestão de medicamentos, em que é necessário adicionar, atualizar e se necessário lista-los;
- O algoritmo na parte de guardar os dados, estes que são essenciais, pois garantem que as informações dos utilizadores, de serviços e de medicamentos sejam armazenados corretamente em ficheiros, e recuperados da forma certa;

É através deste tipo de algoritmos que se mantém um correto funcionamento do projeto, garantindo que este funcione de forma acertada de uma forma geral, pela aplicação toda.

4.4. Estruturas de Dados

O projeto utiliza Vectores para armazenar listas de utilizadores, medicamentos e serviços. A opção por Vector deve-se ao facto de ser uma estrutura dinâmica sincronizada que, contrariamente aos ArrayList garante segurança em operações concorrentes. Além disso é uma estrutura de fácil manipulação, concretamente, no acesso sequencial aos dados.

Vector<Utilizadores>: Utilizado para armazenar a lista de utilizadores.

Vector<Medicamentos>: Aplicado para manter a lista de medicamentos.

Vector<Servicos>: Utilizado para armazenar a lista de serviços.

4.5. Armazenamento de Dados

A persistência de dados é no projeto garantida pela utilização de métodos de armazenamento baseados em ficheiros (e.g., ficheiros de objetos e de texto). Essa abordagem implicou a serialização de objetos Java para arquivos binários, permitindo que as informações fossem salvas de forma fácil e eficiente e posteriormente recuperadas. A utilização de ficheiros foi uma solução simples e eficaz para armazenar dados entre execuções.

Os principais ficheiros de armazenamento de dados utilizados no projeto foram:

dados_apl.dat: Este ficheiro guarda os principais dados da aplicação, nomeadamente, informações sobre os utilizadores, serviços e medicamentos.

credenciais_acesso.txt: Este ficheiro contém informações de login e password para restrição dos acessos à aplicação.

log.txt: Todas as ações dos utilizadores são registadas neste arquivo, seguindo um formato sequencial, com registos integrando o utilizador e a ação realizada.

info_sistema.dat: O número total de execuções do sistema e o último utilizador que acede à aplicação são colocados neste arquivo. Atualizado durante a execução e no encerramento do programa.

4.6. Procedimentos de Teste

Os procedimentos implementados e os testes realizados visaram garantir o correto funcionamento da aplicação e a qualidade do código desenvolvido. No âmbito deste projeto, foram feitos testes unitários (i.e., relativos a cada opção do menu) para análise das funcionalidades, validação do funcionamento e articulação dos métodos concebidos. Além disso, foram feitos testes com foco especial na interação com o utilizador perspetivando um desempenho e comportamento acessíveis aos diferentes tipos de utilizador e a prevenção de erros. Os ficheiros e a informação transitada entre execuções também foi alvo de análise, nomeadamente, para suprir limitações de abertura do ficheiro dados_apl.dat.

Estes procedimentos foram considerados de forma sistemática ao longo do desenvolvimento, contribuindo para a robustez e confiabilidade do código, além de garantir que a aplicação atende às expectativas da farmácia (cliente).

5. Conclusões

A realização deste projeto, tendo como objetivo geral a implementação de uma aplicação em linguagem Java capaz de auxiliar o funcionamento de uma farmácia. Com a utilização de estruturas de dados e a manipulação de ficheiros, e outros conteúdos abordados ao longo do semestre, foi possível a criação de um sistema versátil e funcional.

A aplicação aborda eficientemente a gestão dos utilizadores, com diferenças entre os três tipos. Uma das coisas que contribuiu para a eficiência do sistema foi a forma coerente de como foram feitas as diversas manipulações.

Assim sendo, podemos concluir que o projeto atingiu os objetivos inicialmente estabelecidos, com a exceção de um ou outro requisito não implementado, ao criar uma aplicação capaz de lidar com complexidades da gestão de uma farmácia. Também podemos afirmar que, postos à prova num projeto desta dimensão com esta dificuldade, foi importante pois, permitiu-nos melhorar os nossos conhecimentos nesta linguagem de programação orientada a objetos, esta que foi uma novidade, visto que nunca tínhamos utilizado este tipo de linguagem antes do início do presente semestre, tendo sido o primeiro contacto feito nesta Unidade Curricular de Programação.

De seguida estará representada uma tabela (Tabela 2) com o tempo gasto com este projeto e com a disciplina na globalidade, por elemento do grupo, de forma a ter-se uma noção do quão envolvido foi a realização do mesmo.

Tabela 2 - Tempo Utilizado com a Unidade Curricular de Programação por Elemento do Grupo.

Aulas	Projeto	Relatório
5h / semana	± 55h	7h

5.1. Forças

As forças deste projeto que as distingue de outras abordagens é o facto da aplicação oferecer uma gestão robusta e eficiente no que toca aos utilizadores, permitindo o registo, a devida autenticação e a sua correta identificação. Outro aspeto importante é a parte da segurança, por exemplo um utilizador não conseguir alterar a informação de outro utilizador. A utilização de corretas estruturas de dados contribuindo para a eficácia das operações e as diversas opções para gerir os utilizadores, serviços ou medicamentos. Outro aspeto importante, foi o facto de colocarmos uma verificação a seguir a cada escolha de opção, para caso o utilizador se tenha enganado a escolher, ainda tenha a oportunidade de voltar atrás sem ter de obrigatoriamente entrar nessa opção.

5.2. Limitações

Apesar de existirem forças em relação ao trabalho, também existem algumas limitações. As principais limitações encontradas ao longo do trabalho foram por exemplo, aquando do pedido de dados ao utilizador, não é feita a devida verificação das respostas do utilizador, ou seja, se estiver a ser pedido um número e o utilizador escreve uma palavra. Dada a dimensão do projeto e o tempo disponível não se conseguiu verificar esta limitação nem concluir os requisitos que ficaram parcialmente executados, requisitos [55] e [57], e o requisito [56] que ficou por fazer.

5.3. Trabalho Futuro

Para trabalho futuro, é recomendável de forma a melhorar ainda mais o projeto, poderiam ser feitas todas as verificações e acabar/fazer os requisitos falados anteriormente que ficaram por fazer. Outra melhoria seria implementar por exemplo uma interface gráfica para proporcionar uma experiência visualmente mais apelativa ao utilizador e considerar otimizar o código de forma a melhorar o desempenho da aplicação.

6. Referências

Comissão Europeia (2024, janeiro, 04). *Soluções digitais durante a pandemia.* Disponível em: https://commission.europa.eu/strategy-and-policy/coronavirus-response/digital-solutions-during-pandemic_pt

Dupuiy, Y., & Roland, G. (1999). *Manual de Controlo de Gestão.* Mem Martins: Edições Cetop.

Jordan, H., Neves, J. C., Rodrigues, J.A. (2015). O Controlo de Gestão ao serviço da estratégia e dos gestores (10ª ed.). Lisboa: Áreas Editora.

Replit (2024, janeiro, 08). *How do you clear terminal in Java?*l Disponível em: https://replit.com/talk/ask/How-do-you-clear-terminal-in-Java/46341

Simplilearn (2024, janeiro, 09). *StringBuilder in Java Tutorial*. Disponível em: https://www.simplilearn.com/tutorials/java-tutorial/stringbuilder-in-java

Veloso, M. (2023). *Slides e Materiais de Apoio às Aulas*. [PDF de apoio à UC de Bases de Dados, lecionada na ESTGOH - IPC].

Visual Studio Code (2023, dezembro, 12). *Code editing*. Disponível em: https://code.visualstudio.com/

7. Anexo

Em anexo, externo a este documento, segue o Manual do Utilizador.