Escola Superior
de Tecnologia
e Gestão
Politécnico de Coimbra

Relatório de Projeto

Programação Aplicada

Avaliação Periódica

Autor(es):

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

Data: Março de 2024

Resumo

Este projeto pressupõe o desenvolvimento de uma aplicação para gestão de obras literárias em Java, utilizando os princípios da Programação Orientada a Objetos (POO). O objetivo principal é a construção de uma solução abrangente que integre dados inerentes a obras literárias, autores, revisores e respetivos processos de revisão. Para o desenvolvimento foram implementadas classes que permitiram representar os diferentes elementos da aplicação (e.g., obras, autores) e os processos de revisão (i.e., revisão). A gestão e articulação entre os objetos definidos levou à constituição de classes para gestão (e.g., gereUtilizadores, gereRevisoes) que, contendo diferentes métodos para concretização de ações (e.g., pesquisa, listagem, verificação), permitiram o eficaz cumprimento dos objetivos. A manipulação de bases de dados relacionais através de Java Database Connectivity (JDBC) e o uso de ficheiros para armazenamento de forma persistente (e.g., das propriedades para ligação à Base de Dados) foram alguns dos requisitos da aplicação. Os resultados destacam a eficaz gestão de autores, obras e processos de revisão e 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, Autores, Obras Literárias, Processo de Revisão, Revisores, Utilizadores.

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

O presente projeto pressupõe o desenvolvimento de uma aplicação que permita a gestão de obras literárias e respetivos processos de revisão. A correta operacionalização irá possibilitar a comunicação entre autores, gestores e revisores, nomeadamente, para gestão de processos de revisão, permitindo a submissão de obras para análise, o acompanhamento de processos de revisão e o pagamento que deverá culminar com a publicação da obra pela editora.

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 Aplicada, 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, tratamento de exceções, heranças e polimorfismo, manipulação de ficheiros para armazenamento persistente dos dados, manipulação de bases de dados relacionais através de Java Database Connectivity (JDBC) e construção de uma interface (em modo texto) para interação com o utilizador (Veloso, 2024).

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 direcionadas à gestão de revisões de documentos encontra-se explorado em diferentes softwares. Todavia, versões desenhadas à medida, que sejam específicas para obras literárias e permitam a gestão completa das obras, do respetivo processo de revisão e não apresentem funcionalidades desnecessárias é algo que não se encontrou no mercado.

Neste sentido, a abordagem implementada tentou ir ao encontro das necessidades do cliente (i.e., uma Editora Portuguesa conceituada), preenchendo uma lacuna no mercado e oferecendo uma solução integrada que inicia com a submissão de obras, apresenta um acompanhamento do processo e permite a coordenação e viabilização das revisões em curso.

A incorporação de funcionalidades como listagem de obras, pesquisa de autores, notificações relativas a processos, interação entre revisores e autores, proporcionou uma comunicação efetiva entre os diferentes agentes do serviço. A colocação dos dados numa base de dados e potencial articulação online confere ao sistema uma versatilidade que permite a sua utilização em qualquer local que ofereça acesso remoto.

A abordagem é necessária e relevante mediante a ausência de soluções adequadas existentes, preenchendo uma lacuna no campo da gestão de obras literárias e nos processos de revisão.

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, concretamente:

DESCRIÇÃO GLOBAL DA APLICAÇÃO

Gestão de acesso e utilizadores

[R01] Permitir aos utilizadores registarem-se e autenticarem-se na aplicação.

[R02] Permitir o acesso à aplicação por 3 tipos de utilizadores: gestores da plataforma (ou administradores), autores e revisores.

[R03] Os utilizadores são caracterizados pelos atributos nome, login, password, estado (ativo/inativo), email e tipo (gestor, autor ou revisor).

[R04] Tanto o atributo login como o email devem ser únicos.

[R05] O email deve apresentar um formato válido ([designação] @ [entidade] . [domínio]).

[R06] Cada utilizador apenas pode alterar a sua própria informação, não podendo alterar ou visualizar dados de outros utilizadores, ou criar novos utilizadores.

[R07] A exceção ao requisito anterior são os gestores, que podem visualizar e alterar dados de todos os utilizadores e criar novos utilizadores, nomeadamente outros gestores.

[R08] Tanto os autores como os revisores têm adicionalmente os parâmetros número de contribuinte, contacto telefónico e morada. Os autores possuem ainda estilo literário (e.g. drama, ficção, thriller) e data de início de atividade. Os revisores possuem, adicionalmente, área de especialização (e.g. ciências, literatura, artes) e formação académica.

[R09] O atributo número de contribuinte deve ser único e possuir 9 dígitos.

[R10] O atributo contacto telefónico deve possuir 9 dígitos e iniciar pelos dígitos 9, 2 ou 3.

[R11] Os autores e revisores devem registar-se na plataforma. Os gestores são criados por outros gestores.

[R12] Todos os pedidos de registo de novos utilizadores devem ser notificados aos gestores através da aplicação.

[R13] Os gestores aprovam ou rejeitam os pedidos de registo dos utilizadores. Todos os pedidos devem ser aprovados antes de poderem ser usados para autenticação.

[R14] Caso uma conta seja reprovada pelo gestor ou ainda não tenha sido analisada, quando o utilizador tentar usar essas credenciais deve surgir uma mensagem informativa.

[R15] Em qualquer momento um gestor pode inativar ou ativar uma conta de um utilizador.

[R16] Um utilizador pode solicitar que a sua conta seja removida do sistema. Neste caso deve surgir uma notificação aos gestores que podem aceitar ou recusar o pedido. Se o pedido for aceite, a informação pessoal do utilizador deve ser omitida do sistema, embora os registos associados ao utilizador devem ser mantidos.

[R17] Caso não existiam utilizadores criados na primeira execução, a aplicação deve solicitar credenciais para criar uma conta gestor.

[R18] Após a autenticação, a aplicação deve apresentar a mensagem
"Bem-vindo [nome utilizador]".

[R19] Quando a aplicação estiver a encerrar, deve apresentar a mensagem
"Adeus [nome utilizador]".

Ações

(Intervenientes e responsabilidades)

[R20] Os revisores gerem (aprovam e finalizam) o processo de revisão.

[R21] Os autores submetem obras para revisão, que devem ser revistas por um ou mais revisores, e realizam o respetivo pagamento.

[R22] Os revisores são responsáveis rela realização da revisão.

[R23] Cada obra para revisão é caracterizada pelo autor, título, subtítulo (opcional), estilo literário (e.g. drama, ficção, thriller), tipo de publicação (e.g. capa dura, de bolso) número de páginas,

número de palavras, código ISBN, número de edição, data de submissão e data de aprovação. O código ISBN deve possuir um valor único.
[R24] Uma revisão está associada a uma obra, autor, gestor e revisor responsável.
[R25] Uma revisão é caracterizada por um número de série, que é composto por um número sequencial (a cada pedido o número incrementa), seguido da data no formato *AAAAMMDDHHMMSS*. Por exemplo, se já ocorreram 95 pedidos até ao momento, e às 10h20m10s do dia 21 de Fevereiro de 2023 surge um novo pedido, o mesmo terá o número 9620230221102010.
[R26] Adicionalmente, uma revisão inclui a data da realização, o tempo decorrido (calculado automaticamente desde o início até ao fim da revisão) uma listagem de anotações realizadas, observações genéricas inseridas pelo revisor, um custo do processo e o seu estado.
[R27] As anotações são caracterizadas por uma descrição, a página e o parágrafo onde ocorrem e a data em que foram realizadas.

(Processo)
[R28] O processo inicia com o autor, que solicita uma revisão para uma das suas obras.
Cada pedido recebe um número ISBN aleatório entre 1 e 1 000 000.
[R29] O pedido de revisão tem de ser aprovado pelo gestor. O pedido pode ser aceite ou rejeitado. Se aceitar o pedido, o gestor deve atribuir um ou mais revisores. Deve sempre existir um revisor responsável. Se o pedido for rejeitado o autor deve ser notificado e o processo arquivado.
[R30] Quando um processo de revisão é atribuído a um revisor este deve ser notificado.
O revisor pode aceitar ou rejeitar o processo.
[R31] Caso um revisor rejeite um processo, o gestor responsável pelo processo deve ser notificado e selecionar outro revisor. Na nova listagem de possíveis revisores a atribuir, não deve surgir o(s) revisor(s) que rejeitou (rejeitaram) anteriormente o pedido.
[R32] Após aceitar o processo, o revisor inicia a revisão. Quando concluir a tarefa, encerra essa revisão. A revisão não tem de ter terminada numa única sessão da aplicação.
[R33] No final o autor realiza o pagamento do processo e o gestor será notificado.
[R34] Após a confirmação do autor, o gestor arquiva o processo.
[R35] Tanto o autor como o gestor podem consultar o estado de uma revisão:
a) iniciada (autor submeteu pedido, a aguardar autorização do gestor),
b) aceite (gestor aceitou o pedido, em espera de execução);
c) decorrer (revisor iniciou a revisão);
d) finalizada (revisor terminou a revisão);
e) arquivado (o gestor encerra o processo após o autor pagar o processo).
[R36] Cada autor pode possuir diversas obras e cada obra só está associada a um autor.
[R37] Sempre que uma nova obra é inserida no sistema deve-se verificar se o título já existe, bloqueando a sua inserção em caso de duplicação.
[R38] Uma obra pode possuir várias revisões.
[R39] Para garantir que as obras não contêm plágio, são utilizadas aplicações específicas. Estas aplicações possuem um número limitado de licenças, com prazo de utilização (usualmente um ano). Cada revisão pode usar uma ou mais licenças.
[R40] O gestor é responsável pela inserção de licenças no sistema e atualização do seu número.

Notificações
[R41] Quando um novo utilizador se registou na plataforma o gestor deve ser notificado.
[R42] Sempre que existe um pedido de revisão o gestor deve ser notificado.
[R43] Se um pedido de revisão for rejeitado o autor deve ser notificado.
[R44] Quando um pedido de revisão é atribuído a um revisor este deve ser notificado.
[R45] Se um revisor rejeitar um pedido, o gestor deve ser notificado.
[R46] O gestor deve ser notificado quando um processo de revisão for confirmado/pago pelo autor.
[R47] Sempre que um processo de revisão passou 10 dias sem ser finalizado o gestor deve ser notificado.

Listagens e pesquisas
[R48] Todas as listagens devem permitir visualizar os resultados por páginas de 10 registos cada.
[R49] Todas as listagens devem permitir ordenar os resultados, no sentido ascendente ou descendente.

[R50] Todas as listagens devem permitir realizar pesquisas para filtrar os resultados.
[R51] Os gestores podem listar todos os utilizadores, ordenando por nome.
[R52] Os gestores podem pesquisar utilizadores por nome, login ou tipo.
[R53] Os gestores podem listar todas os pedidos de revisão ordenados por data de criação, título de obra ou por autor.
[R54] Os gestores podem listar todas os pedidos de revisão ainda não finalizados, ordenados por data.
[R55] Os gestores podem pesquisar pedidos de revisão por identificador, estado ou autor.
[R56] Os gestores podem pesquisar pedidos de revisão submetidos dentro de um intervalo temporal (período entre duas datas a indicar pelo utilizador).
[R57] Os gestores podem listar os processos de revisão de qualquer obra a partir do título.
[R58] Os autores podem listar os seus pedidos de revisão ordenando por data de criação ou por número de série.
[R59] Os autores podem pesquisar os seus pedidos de revisão por data de criação, por título ou estado.
[R60] Os autores podem listar as suas obras ordenadas por data de submissão ou por título.
[R61] Os autores podem pesquisar os seus obras por data de registo ou matrícula.
[R62] Os revisores podem listar os seus pedidos de revisão ordenando por data de criação ou por título.
[R63] Os revisores podem pesquisar os seus pedidos de revisão por data de criação, por título ou estado.
[R64] Devem ser implementados mecanismos de pesquisa avançada, 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").

MANIPULAÇÃO E GESTÃO DE BASES DE DADOS RELACIONAIS
[R65] Acesso a uma base de dados relacional que permita gerir toda a informação necessária para a execução da aplicação.
[R66] O acesso à aplicação deve ser restringido com credenciais (login/password), informação que deverá ser armazenada numa base de dados relacional.
[R67] Os parâmetros de acesso à base de dados (IP, porto, nome da base de dados, login e password) devem ser armazenados num ficheiro de texto (Properties), sendo disponibilizada uma interface que possibilite a sua alteração antes do arranque da aplicação (antes do processo de autenticação). Estes parâmetros não necessitam de ser definidos em todas as execuções da aplicação, apenas quando o utilizador solicitar.

GESTÃO GERAL DA APLICAÇÃO
Interação com o utilizador
[R68] Disponibilizar uma interface em modo texto onde o utilizador possa interagir e controlar a aplicação. A organização da interface e escolha de componentes fica ao critério do aluno. Soluções eficientes de menus, respeitando as regras de interação com o utilizador serão bonificadas.
[R69] Calcular o tempo que a aplicação demora a executar a aplicação, desde o arranque até ao utilizador selecionar a opção de saída, apresentando, no final do processo, a seguinte informação:

Início do processo: Terça-Feira; 2023-03-01 11:21:11
Fim do processo: Terça-Feira; 2023-03-01 11:21:56
Tempo de execução: 45132 Milissegundos (45 Segundos; 0 Minutos; 0 Horas)

Monitorização de acessos
[R70] Deverá ser mantida a informação genérica do sistema, como por exemplo a data da última execução da aplicação e o número total (até ao momento) de execuções do sistema.
[R71] Deve existir um registo de ações (log) dos utilizadores no formato:
<data> <hora> <utilizador> <ação>. O log deve ser armazenado numa base de dados relacional.
[R72] Deve ser possível listar o conteúdo do log através da aplicação.
[R73] Deve ser possível pesquisar registos por utilizador no log.
Programação Orientada a Objetos
[R74] A aplicação deve estar corretamente estruturada, tendo em conta o paradigma Orientado a Objetos, recorrendo à linguagem Java.

[R75] Implemente as estruturas de armazenamento necessárias, procurando otimizar os recursos utilizados. Validação de dados e notificações
[R76] Valide todas as leituras de dados do utilizador (e.g. verifique se os nomes são únicos).
[R77] 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 nas Editoras de Livros.

A metodologia implementada teve várias etapas de execução. A primeira visou a construção dos Diagramas Entidade Relacionamento, classes, objetos, atributos necessários. A segunda, a construção do script para criação da base de dados e a definição dos métodos que respondessem às funcionalidades previstas. A terceira, o inicio da conceção do menu (respetiva class main) e ajustes necessários ao correto funcionamento. A última fase foi para testagens e correções do sistema que permitissem a 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 ao uso de diversas plataformas para dar resposta às necessidades de desenvolvimento e modelagem. O Eclipse foi o Ambiente de Desenvolvimento Integrado (IDE) selecionado. A opção cingiu-se ao facto de ser aplicação de código aberto que oferece uma ampla gama de recursos, incluindo suporte a várias linguagens de programação, como Java. Além disso a flexibilidade e extensibilidade tornam-no numa escolha sólida para a configuração deste projeto.

Importa salientar que na fase inicial, para agilizar o progresso e facilitar a colaboração, foi utilizada a plataforma online Replit. O Replit oferece um ambiente de desenvolvimento online com um IDE integrado que permite aos desenvolvedores a programação simultânea diretamente no navegador. Esta plataforma dispõe de recursos de colaboração em tempo real, tornando-a especialmente útil na implementação de softwares construídos em equipa.

Por fim, para a construção dos Diagramas Entidade Relacionamento e obtenção do script de criação da base de dados, foi utilizada a plataforma Onda. Esta plataforma é uma ferramenta dedicada à modelação de dados que permite facilmente criar os diagramas conceptuais e físicos das aplicações em causa dispondo ainda de uma funcionalidade que permite gerar automaticamente o script SQL capaz de criar a base de dados. A sua interface intuitiva e recursos avançados levaram a que fosse considerado neste projeto.

Todos os softwares utilizados foram selecionados com base nas necessidades do projeto e por se considerarem ferramentas robustas, eficientes e colaborativas que atendem às exigências e requisitos considerados.

3.2. Planeamento

O planeamento foi alicerçado nas etapas definidas e apresentadas na metodologia com objetivo de suprir os requisitos e as funcionalidades estabelecidas. A calendarização inicialmente prevista não se mostrou díspar face à execução propriamente dita.

Tabela 1 - Planeamento Previsto

	14/02	01/03	15/03	25/03	29/03	01/04	Previsão Tempo
1ª Fase							15 horas
2ª Fase							50 horas
3ª Fase							40 horas
4ª Fase							20 horas
Relatório							8 horas

A realização do projeto dada a especificidade e os detalhes solicitados foi um desafio. Todavia não se verificou nenhum comprometimento a nível da execução uma vez que todos os requisitos foram implementados, embora alguns não estejam diretamente apresentados no menu da aplicação.

A divisão do trabalho entre os desenvolvedores foi devidamente articulada e considerou plataformas colaborativas online para que o processo fosse mais célere e cada interveniente pudesse apoiar, intervir, contactar, ou analisar o código conjeturado pelo outro desenvolvedor.

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

4. Trabalho Desenvolvido

Neste capítulo será apresentada uma visão ao detalhe daquilo que foi desenvolvido no projeto, descrevendo não só o que foi realizado em termos de classes e métodos, como o pensamento para a sua implementação. Destacar que não foi incluído diagrama de classes neste capítulo, uma vez que esse conteúdo programático só irá decorrer este semestre na Unidade Curricular de Engenharia de Software e até à data de fecho deste trabalho ainda não foram concluídos os conteúdos inerentes à concretização desta recomendação.

4.1. Requisitos Implementados

Os requisitos apresentados no capítulo objetivos e metodologias foram todos implementados, com exceção dos relativos ao plágio, que embora tenham sido considerados na construção da base de dados não foram totalmente implementados. Importa referir que algumas listagens e pesquisas não estão diretamente implementadas no menu, mas em ações do utilizador é possível encontrar alguns desses métodos a serem chamados. O método para alteração do ficheiro com as BaseDados.Properties também foi criado, mas não foi aplicado no menu da aplicação.

4.2. Classes e *Packages*

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", responsável pela execução do programa. Dentro do método "main", 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 "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".

Classe "Revisores.java"

Esta classe herda os atributos da classe "Utilizadores", sendo a classe filha, associada à superclasse "Utilizadores". Foi necessário recorrer à herança uma vez que nesta classe além dos atributos específicos dos "Gestores", existem cinco atributos adicionais (i.e., nif, morada e telefone, área de especialização e formação académica). Tal como nas classes anteriores esta também apresenta um construtor, para inicializar o objeto, 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 "Autores.java"

Esta classe, tal como a classe "Revisores" herda os atributos da classe "Utilizadores". Nela foram considerados cinco atributos adicionais (i.e., nif, morada e telefone, estilo literário e início da atividade). Conforme a classe revisores há um construtor, métodos para ler e atualizar certos atributos e um método "toString" para obter uma frase com certos dados do objeto.

Classe "Revisao.java"

Esta classe modela as revisões no sistema, com as informações inerentes ao pedido de revisão em causa. Tal como as classes anteriores este objeto reúne um conjunto de atributos característicos (e.g., idRevisao, nSerie, dataRealizacao, dataFinal) e integra métodos para obter informações sobre um atributo, fazer-lhes atualizações e o método "toString" com o formato desejado para imprimir mensagens na consola.

Classe "Obras.java"

Nesta classe consideram-se os atributos que caracterizam e integram o objeto relativo às obras literárias (e.g., idObra, titulo, subtítulo, nPaginas, codISBN) e tal como nas classes anteriores existe o construtor e métodos de leitura e atualização dos atributos. É também estabelecida a formatação para o método toString que faz uma representação escrita das informações a serem apresentadas ao utilizador.

Classe "Anotacoes.java"

Nesta classe são representadas as anotações associadas a uma obra literária, nomeadamente as que derivam dos processos de revisão. Na estrutura da classe encontra-se apenas o construtor uma vez que a implementação desta funcionalidade não foi completamente introduzida na interação com revisores e autores.

Classe "Logs.java"

Esta classe é onde se integram os atributos necessários ao armazenamento das informações de log (i.e., ações e data). Este método relacionado com a classe utilizadores irá permitir consolidar informações relevantes sobre a ação dos utilizadores no sistema.

Classe "LigacaoBD.java"

Esta classe integra os métodos responsáveis pela ligação à base de dados. Na sua estrutura encontramos um método para carregar as propriedades da ligação do ficheiro BaseDados.Properties, conectar à base de dados, desconectar, atualizar as propriedades de ligação e reescrevê-las no ficheiro.

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:

- "adicionarInfoAutorBD" – método que permite adicionar um novo autor à base de dados;
- "verificaLoginUtilizador" – método implementado no momento da autenticação, que verifica se o utilizador com um determinado login e password inseridos pelo utilizador, existe na base de dados. 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, retorna ao menu inicial sem acesso às áreas restritas.
- "listarUtilizadoresPendentes" – método que retorna a listagem com todos os utilizadores pendentes para o gestor aprovar.

Classe "GereRevisoes"

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

- "adicionarPedidoRevisao" – método que permite adicionar um novo pedido de revisão à base de dados para posterior análise pelo gestor;
- "geraNSerie" – método que permite gerar um número de série único para os pedidos de revisão com base na concatenação de um número sequencial com a data do pedido;
- "listarPedidosRevisaoDeAutor" – método que retorna a listagem com todos os pedidos de revisão de um determinado autor.

Classe "GereObras"

Classe responsável por gerir, através dos métodos criados nesta, as várias obras literárias da aplicação. Alguns dos métodos criados e implementados foram por exemplo:

- "pesquisarObrasDeAutorDataSubmissao" – método que permite pesquisar obras de um autor considerando a data de submissão;
- "adicionarObra" – método para inserir uma nova obra literária na base de dados;
- "criaVerificaISBN" – método para criar e verificar a existência de um ISBN na base de dados.

Classe "GereNotificacoes"

Classe responsável pela gestão das notificações do sistema. Alguns dos métodos criados e implementados foram por exemplo:

- "adicionarNotificacao" – método que permite adicionar uma notificação à base de dados;
- "listarNotificacoesGestores" – método para listar as notificações que dizem respeito aos gestores;
- "darVistaNotificacoes" – método para marcar como vista as notificações de um determinado utilizador.

Classe "GereLogs.java"

Classe responsável pelo registo e apresentação das ações que os utilizadores desenvolvem no sistema. Exemplos dos métodos criados são:

- "logAcao" – método que permite adicionar um log à base de dados;
- "listarLogsUtilizador" – método para listar os logs de um determinado utilizador.

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

Class GereUtilizadores

Ex. Métodos
verificaLogin()
verificaFormatoEmail()
verificaEmail ()
verificaNIF()
verificaFormatoTelefone()
listarUtilizadores()
listarUtilizadoresPendentes()
pesquisaPorTipo()
…

Class Revisores (Classe Filha)

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

Ex. Métodos
getMorada()
setNif()
…

Class GereRevisoes

Ex. Métodos
listarPedidosRevisao()
pesquisarPedidosRevisaoIdEstado()
listarPedidosRevisaoTitulo()
listarPedidosRevisaoEstados()
alteraEstadoRevisao()
concluirPedidoRevisao()
pesquisarPedidosRevisaoDeAutorEstado()
idUltimoPedido()
…

Class Obras

```
private int idObra;
private Autores autor;
private String titulo;
private String subTitulo;
private String estiloLiterario;
private int tipoPublicacao;
private int nPaginas;
private String codISBN;
private int nEdicao;
private String dataSubmissao;
private String dataAprovacao;
```

Ex. Métodos
setId()
getTitulo()
…

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 Autores (Classe Filha)

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

Ex. Métodos
getMorada()
setNif()
…

Class Revisao

```
private int idRevisao;
private String nSerie;
private String dataRealizacao;
private String tempoDecorrido;
private String observacoes;
private float custoProcesso;
private int estado;
private Obras obra;
private Autores autor;
private Utilizadores gestor;
private Revisores revisor;
private Vector <Anotacoes> listaAn…;
```

Ex. Métodos
setId()
getId()
…

Class GereObras

Ex. Métodos
listarObrasDeAutor()
pesquisarObrasDeAutorDataSubmissao()
adicionarObra()
ultimoIdObra()
atribuirAutorObra()
…

4.3. Algoritmos

Na realização deste projeto, foi necessário desenvolver algoritmos de forma a que o sistema tivesse um funcionamento correto, tendo em conta as diversas operações de gestão. De seguida estarão representados alguns dos algoritmos implementados no trabalho:

- O algoritmo de registo e de autenticação de forma a poder validar se determinados atributos estão disponíveis para se utilizar e se as credenciais introduzidas pelo utilizador na hora da autenticação estão corretas.
- A aprovação de registos, no qual o gestor pode aprovar os registos de novos utilizadores, validando neste caso as informações deste.
- A possibilidade de ativar ou desativar um utilizador, a pedido de este ou devido a outros fatores.
- A gestão de revisões, no qual é necessário a criação, a aprovação e a execução de revisões, que envolvem todos os utilizadores da aplicação, de forma a garantir uma maior eficiência nos seus processos.
- A gestão das obras, para poder adicionar, atualizar e listá-las.
- A gestão dos utilizadores, de forma a poder adicionar, atualizar e listar os utilizadores.
- O algoritmo de gestão de notificações para os utilizadores poderem visualizar as suas notificações referentes a alguns passos do processo de revisão entre outras.
- E o algoritmo de gestão de logs de forma a ser possível a visualização de um certo utilizador em casos de necessidade.
- Para guardar os dados necessários ao longo dos processos, um dos processos mais importantes, recorreu-se a uma base de dados que garantem que as informações dos utilizadores, das revisões, das obras, das notificações e dos logs sejam armazenados corretamente e recuperados também de forma certa.

Com este tipo de algoritmos mantém-se um correto funcionamento do programa, de forma a que este funcione de forma acertada de forma geral.

4.4. Armazenamento de Dados

Para este projeto de forma a garantir a persistência dos dados, foi utilizada uma base de dados, com várias tabelas e os atributos de acordo com o que se pretendia caracterizar. Também se utilizou um ficheiro de texto, de propriedades, de forma a armazenar os dados de acesso à base de dados.

BaseDados.properties: Este ficheiro guarda os dados para que se consiga aceder à base de dados de forma a inserir, visualizar ou alterar os dados dependendo daquilo que se pretenda.

Base de Dados: base de dados onde é armazenada toda a informação necessária para o programa.

A construção da base de dados teve por princípio o diagrama Entidade Relacionamento, onde se estrutura as informações relevantes do sistema e a forma como se relacionam. As tabelas a implementar na base de dados, as variáveis e suas especificidades (e.g., se é primary key, chave forasteira) são esquematizadas e sintetizadas quer no diagrama concetual quer no diagrama físico. Seguidamente apresentam-se os diagramas.

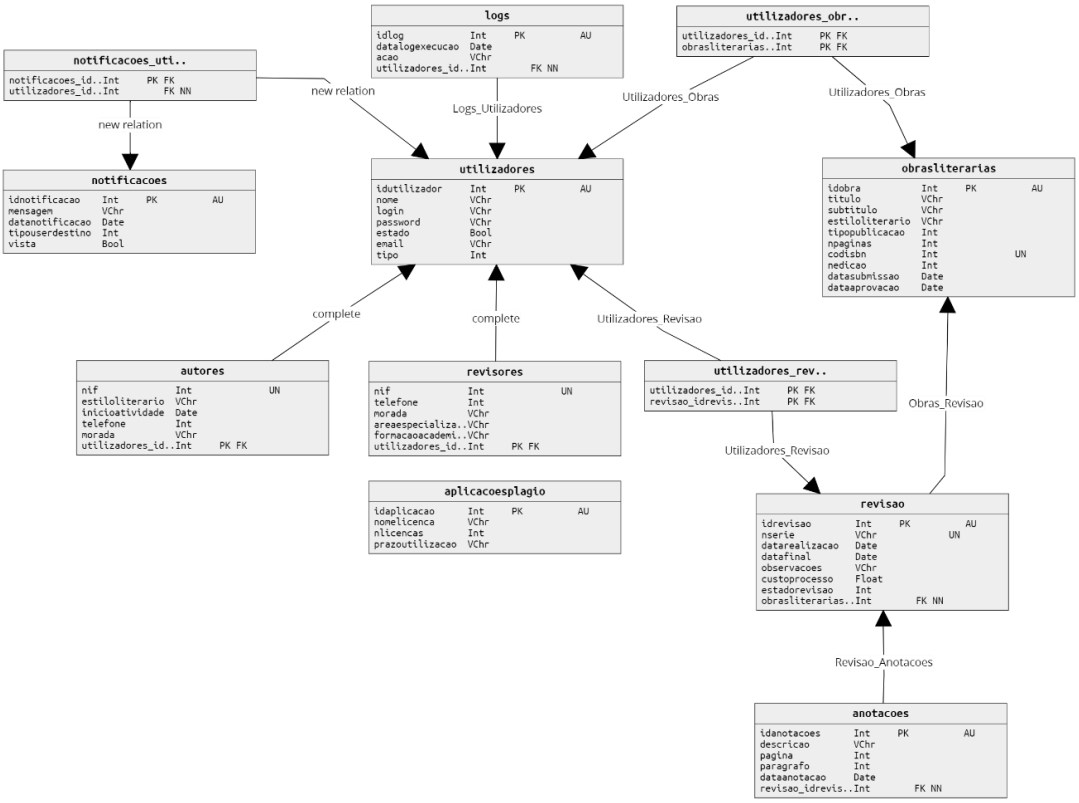

Figura 4-1 – *Diagrama Conceptual.*

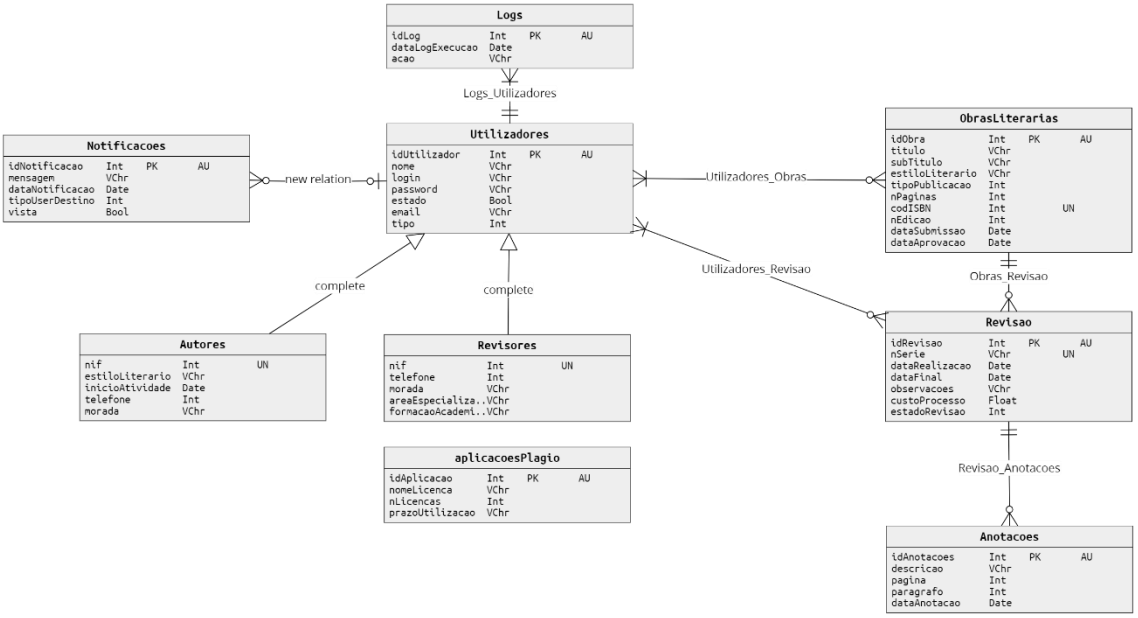

Figura 4-2 – *Diagrama Físico.*

4.5. Procedimentos de Teste

Os procedimentos implementados e os testes realizados serviram para garantir o correto funcionamento da aplicação e também uma boa qualidade e otimização do código em grande parte do programa. No âmbito deste projeto, foram realizados testes unitários (i.e., relativos a cada opção do menu) para análise das funcionalidades, validação do funcionamento e articulação dos métodos feitos. Também foram realizados testes para garantir que a interação com o utilizador fosse acessível e não criasse grande confusão para com este, de forma a evitar erros. Ainda durante os testes, foram verificados se as informações estavam corretas sem qualquer erro entre diferentes execuções.

Estes procedimentos foram considerados de forma sistemática ao longo do desenvolvimento, contribuindo para a robustez e confiabilidade do código.

5. Conclusões

Neste primeiro projeto, o objetivo foi a criação de uma aplicação em linguagem Java dedicada a auxiliar uma editora no processo de revisão de obras literárias. Ao longo da implementação deste sistema, foi necessário utilizar conceitos de programação orientada a objetos, a manipulação de uma base de dados relacionais utilizando JDBC e o uso de ficheiro para armazenamento de dados. Além disso, o desenvolvimento de interfaces para interação com o usuário em modo texto foi uma parte crucial do processo.

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.

Desta forma, podemos concluir que o projeto atingiu os objetivos inicialmente propostos, tirando um ou outro requisito, que pode ter sido implementado indiretamente e não diretamente com o utilizador. Este projeto proporcionou uma oportunidade para aplicar e aprimorar os conhecimentos e competências adquiridos ao longo do primeiro semestre nas Unidades Curriculares de Programação e de Bases de Dados.

De seguida estará representada uma tabela 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, 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. A principal limitação encontrada ao longo do trabalho foi por exemplo, devido à

dimensão do projeto e o tempo disponível não foi possível verificar todos os pormenores que podiam ser melhorados, no entanto são pormenores.

5.3. Trabalho Futuro

Para trabalho futuro, é recomendável de forma a melhorar ainda mais o projeto, 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

Stackoverflow (2024, março, 18). *Regular Expression for Mobile*. Disponível em: https://stackoverflow.com/questions/45704925/regular-expression-for-mobile-and-starting-with-09-or-04

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

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

Veloso, M. (2024). *Slides e Materiais de Apoio às Aulas*. [PDF de apoio à UC de Programação Aplicada, lecionada na ESTGOH - IPC].

Velykozhon, D. (2024, março, 18). *Simple Email Validation in Java*. Disponível em: https://mailtrap.io/blog/java-email-validation/