Curso: Mestrado em Sistemas de Informação [MSI]
Unidade Curricular: Engenharia de Software para Sistemas de Informação
Momento de avaliação: 3 | **Grupo:** 7

RESUMO

Para lidar com o atual mercado, que se está a tornar cada vez mais exigente e competitivo, as empresas estão a passar por momentos em que é necessário se reinventar constantemente, tomar decisões e fazer mudanças. Face a isso, os membros do grupo consideram pertinente a abordagem à inovação tecnológica de empresas de comércio tradicional (restauração, mercearias, e outros negócios ligados ao comércio), que estejam estagnados, mesmo que estejam inseridas numa sociedade de consumo. Estes ainda não têm a sua própria projeção na *web* ou então não possuem um sistema de informação que regularize o seu inventário para compras mais assertivas. Assumindo que as tecnologias da informação assumem um papel fundamental no que concerne à inovação de processos de uma organização, isto é, um papel estratégico face aos avanços científicos e tecnológicos, num mundo em transformação rápida e profunda, é crucial que o restaurante se mantenha atualizado relativamente às mudanças significativas que possam vir a ocorrer, consequência da magnitude da indústria tecnológica.

Palavras-chave: Tecnologias de informação, gestão de stocks, restauração

Considerações Iniciais

Este trabalho académico tem como finalidade uma das componentes de avaliação que foram delineadas no início do primeiro semestre na unidade curricular de Engenharia de Software para Sistemas de Informação do Mestrado em Sistemas de Informação, no ano letivo 2023-2024, sob orientação do Professor Doutor Ricardo Jorge Silvério Magalhães Machado. Consiste num relatório de grupo onde elenca uma abordagem académica a problema fictício, envolvendo a implementação de um software para melhorar os processos de gestão de um restaurante nigeriano. Tal como solicitado pelo docente, o problema em questão está inserido na área de "Engenharia de Software de Requisitos" do SWEBOK. O mesmo problema envolve a identificação, análise e especificação dos requisitos específicos do sistema que, posteriormente, será desenvolvido para atender às necessidades atuais do restaurante.

Em primeira instância, foi proposto pelo docente a definição do problema e os requisitos funcionais e não-funcionais. No presente relatório, a solução do problema será focalizada na ótica do utilizador, isto é, no manuseamento do *software* por parte dos *steakholders* do restaurante. No que concerne a requisitos funcionais irá contemplar o nível do registo de ingredientes, atualização de stock, auxílio na elaboração dos relatórios de stock e a gestão de fornecedores. Por outro lado, relativamente a requisitos não-funcionais, o relatório irá integrar a acessibilidade, usabilidade, desempenho, segurança e disponibilidade.

Definição do Problema Fictício

Ezekiel, um empresário nigeriano que possui um restaurante no Porto, está a ter um sucesso crescente. No entanto, o crescimento do seu negócio está a colocar desafios à gestão do restaurante, nomeadamente na gestão de stocks. O sistema manual que o restaurante utiliza é lento e impreciso, o que leva a erros, perdas de informação e insatisfação dos clientes. Face ao crescimento do negócio, a gerência considerou a necessidade de inovar os processos organizacionais, nomeadamente a gestão de stocks e a necessidade colmatar as falhas sinalizadas pelo dono e pelo gerente do restaurante como, os erros de

registo, ou seja, a gerência sinalizou que o staff ainda utiliza um bloco de notas e uma caneta para anotarem o stock da mercadoria, o que por vezes provocava erros de registo, como por exemplo, registos duplicados ou incorretos. Outra ocorrência verificada foram os cálculos incorretos de stock, o que levava a decisões inadequadas de compra e controlo de stock impreciso, ou seja, a ineficiente gestão do inventário do armazém resultava em custos adicionais, relacionados com o espaço de aprovisionamento, bem como a gestão de fornecedores, pois não existia nenhum mecanismo que armazenasse informações detalhadas de cada compra, nome dos fornecedores, contato, endereço, etc, salientando que a comunicação com os fornecedores era feita à base de chamadas telefónicas, o que também podia provocar erros de registo.

Nos dias de maior afluência de clientes no restaurante, ocorria a falta de ingredientes essenciais, o que impactava negativamente a qualidade dos serviços prestados pelo restaurante no que concerne à confeção de determinadas ementas.

Além dos problemas anteriormente mencionados, o dono possuía uma considerável quantidade de documentos físicos, como recibos, faturas e notas de encomendas que eram armazenados em diferentes locais, no escritório, na cozinha e no armazém (guardados em caixas ou dossiês sem rótulos ou etiquetas). Ou seja, eram catalogados de forma desordenada e sem qualquer sistema de organização, sendo que esta forma de armazenamento dificultava a consulta dos documentos, o que poderia levar a erros na gestão financeira do restaurante, ou seja, a falta de organização dos documentos pode levar a erros de cálculo, o que pode afetar os rendimentos do restaurante. Ressalva-se para além de perdas financeiras (desperdício significativo de alimentos que podem ultrapassar a validade ou deteriorar-se, desencadeia-se uma possível insatisfação dos clientes), ocorrem erros de faturação e, o custo de mão-de-obra torna-se mais dispendioso, na medida em que o staff gasta mais tempo a procurar os alimentos para a confeção. Evidencia-se assim uma clara desorganização no que concerne ao armazenamento de informação, ou seja, os documentos que são armazenados de forma desorganizada, podem ser perdidos ou deteriorados.

Posto isto, o Ezekiel, estava preocupado com o crescimento do seu restaurante. O negócio estava a crescer a um ritmo acelerado, mas os processos organizacionais rudimentares, morosos e manuais que o restaurante utilizava estavam a tornar-se ineficientes. Devido à sua inquietação o Ezekiel contactou uma software house para obter aconselhamento sobre como melhorar a gestão de stocks e a gestão de cadeia de abastecimento do seu restaurante. Após uma reunião com profissionais de tecnologias de informação, o senhor Ezekiel ficou convencido de que a implementação de um sistema de informação funcional era a solução ideal para os seus problemas.

Os consultores da software house propuseram que a implementação de um sistema de informação funcional para a gestão de stocks e fornecedores acarretaria várias vantagens para o restaurante, nomeadamente, a simplificação da gestão manual do stock, ou seja, o sistema de informação automatizaria a gestão do stock, o que simplifica o processo e reduz o risco de erros. Outras mais-valias concedidas seriam a diminuição do *lead time* do staff, ou seja, o tempo dos funcionários poderia ser melhor aproveitado em tarefas mais produtivas, como um atendimento mais personalizado ao cliente e melhoria dos processos da cozinha, a melhoria da eficiência operacional, ou seja, a gestão de stocks e fornecedores tornar-se-ia mais eficiente, o que podia levar a uma redução dos custos operacionais, bem como a uma melhor capacidade de tomar decisões estratégicas, porque o sistema de informação, seria capaz de fornecer dados mais precisos sobre o stock e informaçõs sobre os respetivos fornecedores, o que permitiria ao restaurante tomar decisões mais estratégicas, como por exemplo, a compra de ingredientes que sejam provenientes do país de origem do empresário.

Os indicadores, assim, culminam na automatização do processo de registo de stock, fornecendo dados precisos sobre o stock de ingredientes e permitindo gerir de forma eficiente o *layout* do armazém. Por outro lado, é necessário dar enfâse à lead time do staff a médio-longo, o que melhora sua eficiência

operacional e a satisfação dos clientes do restaurante. Ao rentabilizar o tempo, o staff poderá gerir melhor o tempo, na concretização de tarefas mais produtivas, como um atendimento mais personalizado ao cliente e na melhoria dos processos da cozinha. Relativamente aos funcionários que servem às mesas, salienta-se que um atendimento mais personalizado ao cliente pode ser realizado através de uma melhor compreensão das necessidades dos clientes e de uma maior disponibilidade dos funcionários para responder às suas questões. No que concerne aos funcionários que operam na cozinha, terão mais tempo a preparar os pratos, garantindo que são preparados com os ingredientes corretos e de acordo com as especificações do cliente (pois existem clientes que poderão ter gostos e preferências específicas num prato). Por outro lado, os funcionários poderão passar mais tempo a organizar e limpar o restaurante, garantindo que é um ambiente agradável para os clientes.

Em suma, ressalva-se que a eficácia dos processos organizacionais é fundamental para o sucesso dos estabelecimentos da área da restauração. Se a concretização destes se tornarem eficientes no controlo do stock e na gestão de fornecedores, os custos operacionais poderão ser reduzidos, assim como a experiência dos clientes tornar-se-á mais consistente e satisfatória. Em suma, face a este problema consta-se que os estabelecimentos de restauração, considerem abordagens inovadoras para impulsionarem o seu posicionamento, para atender à procura do mercado, mas também posicionar o restaurante para prosperar num ambiente empresarial em constante evolução.

Requisitos Funcionais e Requisitos Não-Funcionais

Em engenharia de software, requisitos funcionais (*functional requirements*) são declarações sobre o que o software deve fazer, e o que não deve fazer. Descrevem as funcionalidades que o software deve fornecer aos utilizadores, bem como as restrições sobre como essas funcionalidades devem ser implementadas no sistema. Os requisitos funcionais, além de serem uma componente essencial do processo de desenvolvimento de software, ajudam a garantir que este atende às necessidades dos utilizadores.

O requisitos-funcionais da eventual solução de inovação organizacional, será dirigido para os *steakholders.* Estes devem ser capazes de ler os itens da base de dados, alterar, adicionar e remover os mesmos, para que *software* seja acessível e intuitivo na ótica do utilizador, e que estes possam introduzir manualmente informações sobre os produtos, incluindo nome, descrição, quantidade inicial, data de validade e o seu respetivo fornecedor. Para além disso, o utilizador será capaz de visualizar informações acerca de cada fornecedor. Quando se depararem com o nível de stock baixo poderão encomendar a quantidade recomendável dos recursos pretendidos. Isto é, o sistema permite que o utilizador faça encomendas quando os níveis de stock de alimentos e ingredientes atingem um limite mínimo predefinido.

Por outro lado, o utilizador deverá conseguir atualizar as movimentações de stock, como compras, vendas, transferências entre locais, desperdício e perda. Estas funcionalidades irão auxiliar na elaboração de relatórios detalhados acerca quantidade exata do stock, bem como as movimentações, custos associados e futuras tendências de compra. Poderá, eventualmente, permitir que seja possível exportar a documentação para outros formatos.

No que concerne aos requisitos não-funcionais (not-*functional requirements*), estes definem condições impostas ao sistema e no âmbito das quais ele deve operar. Estes descrevem atributos de qualidade do *software*, como, acessibilidade, desempenho, segurança, usabilidade e disponibilidade. Os requisitos não-funcionais, sendo uma parte essencial do processo de desenvolvimento de software, ajudam a garantir que o software corresponde às expectativas dos *stakeholders*. São declarações que descrevem

qualidades e características do sistema que influenciam a experiência dos utilizadores. Para que o software seja acessível, o sistema deverá, por exemplo, corresponder da melhor forma, quando um funcionário estiver a manusear o programa, garantindo-lhe que as «janelas» carregam em menos de 2/3 segundos, traduzindo-se, assim, numa melhor experiência de utilização. Este processo tornar-se eficiente e concretizável por causa da interface do utilizador ser intuitiva (usabilidade).

No que concerne ao desempenho, o sistema deverá ser capaz de lidar com os diferentes *steakholders* em simultâneo (patrão e gerente, estes podem alterar o que entenderem e aceder sem qualquer restrição, a equipa operacional que lida diretamente com o stock, só poderá visualizar, adicionar ou remover). Relativamente à visualização da lista de fornecedores só quadros superiores da organização terão acesso. os dados do utilizador devem ser armazenados com criptografia" (segurança)

Os dados do utilizador serão protegidos por uma *password* de administrador que apenas o patrão e o gerente terão acesso. Este mecanismo permite que os dados do estabelecimento estejam salvaguardados de terceiros.

A disponibilidade do sistema ocorre no horário laboral, sendo que o primeiro contacto com o sistema é feito pelo patrão, podendo aceder a qual informação relacionada com stock a qualquer momento, incluindo durante as operações de maior afluência, sem interrupções no serviço.

Em jeito de conclusão, considera-se que ambos os requisitos são essenciais para o desenvolvimento de software, pois, garantem que o sistema atenda não apenas às expectativas e funcionamento, mas também às necessidades dos utilizadores.

Consonância entre a solução tecnológica e o SWEBOK Guide

O presente trabalho académico comtempla a prossecução da avaliação contínua da unidade curricular da unidade Engenharia de Software para Sistemas de Informação (ESSI) do Mestrado em Sistemas de Informação (MSI), no ano letivo 2023-2024. Contém o conteúdo considerado central no âmbito do segundo momento de avaliação, isto é, o enquadramento da solução tecnológica no SWEBOK e a “completude do planeamento do projeto”, constituído através do UML (*Unified Modeling Language*). Será elencado ilustrações de três pacotes de diagramas de casos de uso, bem como o seu refinamento e ilustrações de diagramas de sequência, de atividade e estados.

Para um desenvolvimento bem projetado da solução tecnológica (sistema de gestão de stocks do restaurante), alicerça-se na base os princípios e normas do guia SWEBOK Guide, proporcionando uma visão abrangente do corpo de conhecimento da Engenharia de Software (ES). Nas aulas de ESSI, através da orientação do Professor Doutor Ricardo Machado, o docente referiu que pelo facto do SWEBOK Guide, conter 15 áreas de conhecimento bastante vasto, não será possível que os alunos desenvolvam competências em todas as áreas de conhecimento e, espera-se que os alunos adquiram competências nas partes que consideram pertinentes para o desenvolvimento da solução tecnológica. Face à circunstância anteriormente mencionada, os membros do grupo optaram por escolher os requisitos e a conceção do *software*.

Visto que o tema central incide na resolução de problemas de gestão de stocks do restaurante, como por exemplo a segurança dos dados e maior fiabilidade acerca da quantidade de stock existente no restaurante, o enfoque será contrariar o dispêndio, visto que é uma rotina cansativa com muita documentação para verificar, com melhoria em vista a inovação os processos organizacionais. Assim, surge os requisitos de software, que são uma das principais áreas de conhecimento do SWEBOK, essenciais para garantir que o software atenda às necessidades dos seus utilizadores/*steakholders*.

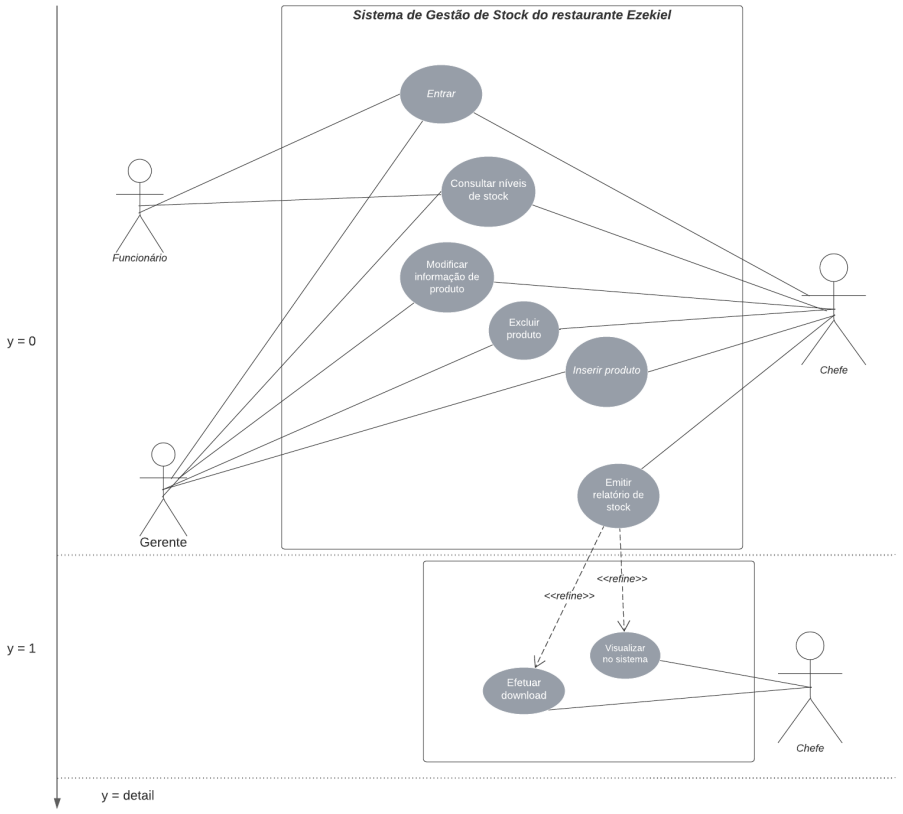
O processo de engenharia de requisitos do software envolve quatro atividades principais: elicitação, análise, especificação e validação. Visto que o restaurante deseja automatizar os processos alusivos à gestão de stocks, em primeira instância, ocorre a elicitação dos requisitos, isto é, começar por identificar os *stakeholders* envolvidos no processo. Neste caso, os principais *stakeholders* são o chefe, que é o responsável pelo negócio e tem acesso a todas as funcionalidades proporcionadas pelo sistema, o seu gerente, que para além de ser seu coadjuvante, tem a visão geral da funcionalidade do sistema.

No que concerne à concepção do *software*, é a atividade intermediária entre o levantamento de requisitos e a construção de software. Considera-se que é uma área que envolve as soluções de arquitetura, interface e outras características do sistema. É um processo interativo através do qual os requisitos são traduzidos num documento para construção do *software.* Este domínio dedica-se à análise meticulosa dos requisitos do *software* com o objetivo primordial de elaborar uma descrição robusta da estrutura interna do *software*, servindo como alicerce para a sua posterior construção. Esta área não se limita apenas à escolha de tecnologias ou à definição de estruturas de dados, pois abrange a modelagem holística do sistema. Posto isto, utiliza ferramentas como diagramas de classe, diagramas de sequência e outro tipo de representações.

No que concerne, à concepção do software para a sistema de gestão de stocks no restaurante, os engenheiros de software, focalizar-se-ão, nas atividades diárias que são feitas para os níveis de gestão de stock, como por exemplo, o registo de ingredientes, a atualização de stock e sua respetiva elaboração de relatórios.

Diagramas UML

Diagrama de caso de uso



(Figura 1)

Para a viabilização da solução tecnológica, os membros de grupo, inicialmente construíram um diagrama de casos de uso **(Figura 1),** que identifica os *steakholders* do restaurante: funcionários, gerente e o respetivo chefe. Salienta-se que o sistema deverá conceder diferentes acessos a cada ator consoante a sua responsabilidade na organização. Os funcionários, terão a possibilidade de aceder ao software e consultar níveis de stock, como também aceder a determinadas informações dos vários produtos alimentares existentes. Por outro lado, o gerente ao assumir o papel de coadjuvante do chefe, será capaz de reunir os dados essenciais a serem atualizados diariamente, como o número de produtos utilizados nesse dia, produtos e ingredientes em escassez, entre outros. Posto isto, terá acesso às opções de "modificar, adicionar ou remover produtos", como também às funcionalidades que um funcionário pode exercer. Por fim, o chefe poderá realizar todas as tarefas anteriormente mencionadas, para além dessas poderá emitir relatórios de stock, de forma a fazer balanços mensais dos consumos do restaurante.

Posteriormente, após a identificação dos atores, delineou-se as principais ações a serem realizadas no software de gestão do restaurante (casos de uso). As ações pretendidas são uma autenticação do sistema, consulta de níveis de stock, que serão realizadas por todos os *steakholders*. No que concerne à modificação de informação dos produtos (excluir, inserir, alterar) serão realizadas apenas pelo gerente e pelo chefe. Por fim, os membros do grupo identificaram o caso de uso "emitir relatório de stock" que o chefe será o único ator que terá acesso à funcionalidade.

O caso de uso "Emitir relatórios de stock" é refinado porque tem duas variantes: download e visualização no sistema. No caso de download, o sistema gera um relatório em formato PDF e o envia para o utilizador. Enquanto, que no caso de visualização no sistema, o sistema exibe o relatório numa janela do sistema. Por outro lado, optou-se por não refinar os restantes casos de uso devido ao facto de serem simples e de fácil compreensão e não terem várias variantes subentendidas.

No que concerne à construção de ilustrações de outros diagramas UML, para modelação do software utilizado na solução, foi decidido de forma unânime, o agrupar algumas funcionalidades do sistema em «pacotes». Optou-se por criar três pacotes de diagramas, também baseados na relação entre os casos de uso, levando em consideração o número de atores associados a casos de uso:

Em primeiro plano, surge o "**Pacote de Acessibilidade**", que constitui a autenticação dos três tipos de atores no sistema e o primeiro contacto com as informações comtempladas na base de dados. Integra os casos de uso "Entrar" e "Consultar níveis de stock". Posteriormente surge, o pacote de "**Edição de informação**", que representa a edição dos dados. No que concerne ao acesso deste pacote, só será possível realizar apenas por dois atores, o gerente e o chefe. Abrange os casos de uso "Modificar informação de produto", "Excluir produto" e "Inserir produto". Por fim, o pacote de **emissão de relatórios,** que traduz a ação de emissão de relatórios, ou seja, o caso de uso "Emitir relatório de stock", que pode ser apenas realizada pelo chefe.

Diagramas de interação

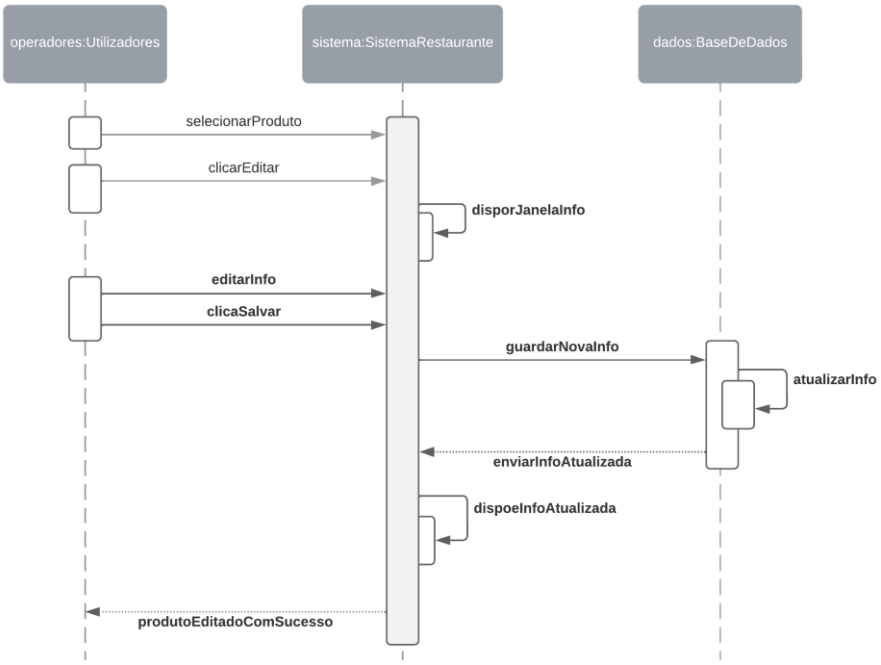
Os diagramas de interação representam o fluxo do controlo de um sistema. Estes diagramas são utilizados para modelar o comportamento dinâmico de um sistema. Têm como finalidade apresentar as atividades que são executadas, os objetos envolvidos e as «mensagens» trocadas entre os respetivos objetos (utilizador, sistema de gestão de stocks e a base de dados).

Estes diagramas são usados para “formalizar” uma série de cenários que o engenheiro de requisitos concebe como formas típicas do funcionamento de um sistema. Estes cenários são baseados nos requisitos funcionais e representam as interações entre os *steaksholders* e o sistema.

Existem dois tipos de diagramas de interação: os diagramas de colaboração e os diagramas de sequência.

Os membros do grupo optaram pelos diagramas de sequência, pela formalização de formas típicas de funcionamento do software. No entendimento da equipa, este tipo de diagrama consegue ser mais legível e intuitivo na clara explicitação da troca de «mensagens» entre os objetos, face aos diagramas de colaboração.

No âmbito da elaboração dos diagramas de sequência, os membros do grupo orientaram-se por selecionar o caso de uso no diagrama de casos de uso, detalhar como se opera o caso de uso, identificar os atores que interagem com esse objeto, identificar as mensagens entre os objetos e a sua sequência e, por fim o desenho do diagrama.



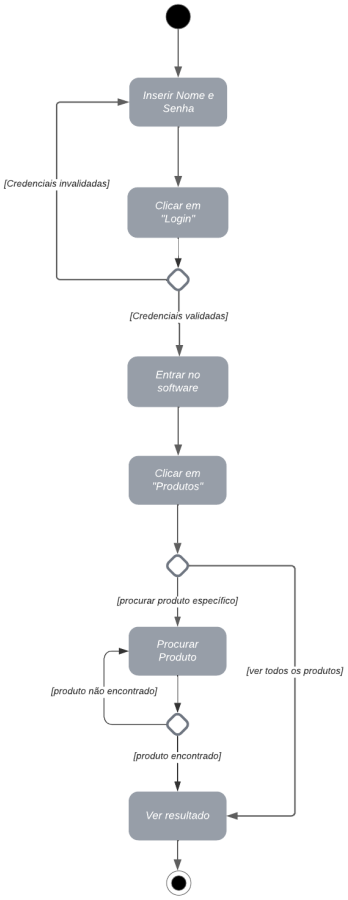
(Figura 2)

A **Figura 2** será apresentado, um dos diagramas de casos de uso elaborados, que corresponde ao pacote de “edição de informação”. Este pacote terá como propósito a edição de um produto que já esteja registado na base de dados. Inicialmente, o ator começa por selecionar o produto que deseja alterar. Consequentemente, o sistema abre uma janela com as informações do mesmo, onde é possível alterar os dados do produto, como o nome, descrição, preço unitário, etc. Após serem efetuadas as alterações desejadas, o utilizador deve pressionar no botão "salvar". O sistema envia uma «mensagem» à base de dados com as alterações efetuadas e atualiza as informações do produto. Em simultâneo, envia a informação atualizada de volta para o sistema e, posteriormente mostrará a nova alteração na interface com a tabela dos produtos.

Diagrama de atividades

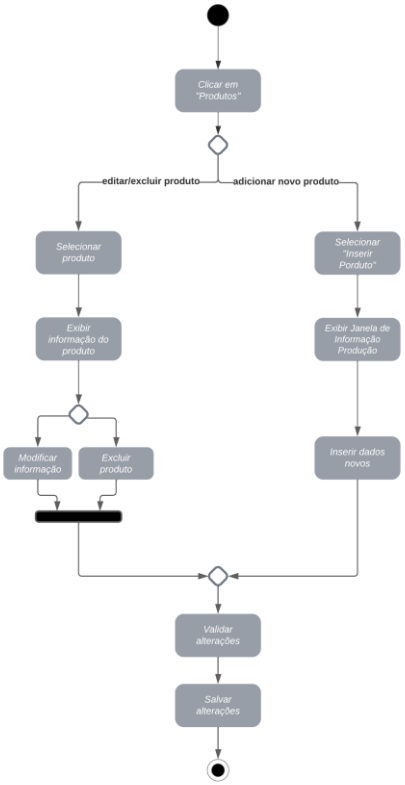
Os diagramas de atividades UML são utilizados para modelação do comportamento dinâmico de um software, onde são indicadas as atividades realizadas e a ordem pela qual estas são realizadas. Estes diagramas contêm as atividades que representam ações que são realizadas, as transições que representam o fluxo de controlo entre as atividades e os eventos que representam ocorrências que podem ocorrer durante o fluxo de controlo.

Entende-se que para a criação do diagrama, este deverá consistir em três etapas fundamentais, sendo que a primeira, incide na identificação dos atores e as suas responsabilidades, a segunda é identificação dos casos de uso e a terceira, o mapeamento o fluxo de controle, com base nos casos de uso.



(Figura 3)

A **figura 3** evidencia a entrada na aplicação e começa com o utilizador, a introduzir o nome de utilizador e a respetiva senha. Se as credenciais forem válidas, o utilizador é autenticado e autorizado a entrar no software. Em seguida, uma vez dentro do software, o utilizador pode consultar os níveis de stock clicando em “Produtos”, onde será disposto uma tabela com os dados dos produtos. Caso o utilizador pretenda procurar um produto em específico, pode fazê-lo através da barra de pesquisa que o sistema apresenta na janela de produtos. Se o produto for encontrado na base de dados, o sistema exibe as informações relativas. Caso contrário, o sistema exibe uma mensagem de erro.



(Figura 4)

A ilustração supracitada, representa a **figura 4**, que ilustra uma representação visual do fluxo de trabalho no sistema de gestão de produtos. É iniciado ao pressionar o botão "Produtos". Após este acontecimento, ocorre uma bifurcação que dá aos utilizadores a opção de editar/excluir um produto existente ou adicionar um novo produto.

No que concerne à caso de edição/exclusão, os utilizadores ao selecionarem o produto desejado, o sistema exibe as informações. Posteriormente, na ocorrência deste acontecimento podem ser realizadas duas atividades distintas: modificar informações ou excluir o produto pelo utilizador. No entanto, se o utilizador pretender adicionar um novo produto, deverá selecionar a opção "Inserir Produto". O sucedido irá desencadear a exibição de uma janela com informações acerca do produto, prosseguindo-se por uma atividade de inserção de dados do mesmo. É importante salientar que todas as atividades anteriormente mencionadas culminam na convergência das duas atividades finais. Salienta-se que, a validação das alterações (antes de salvar), garante a consistência e integridade dos dados no sistema de gestão de produtos.

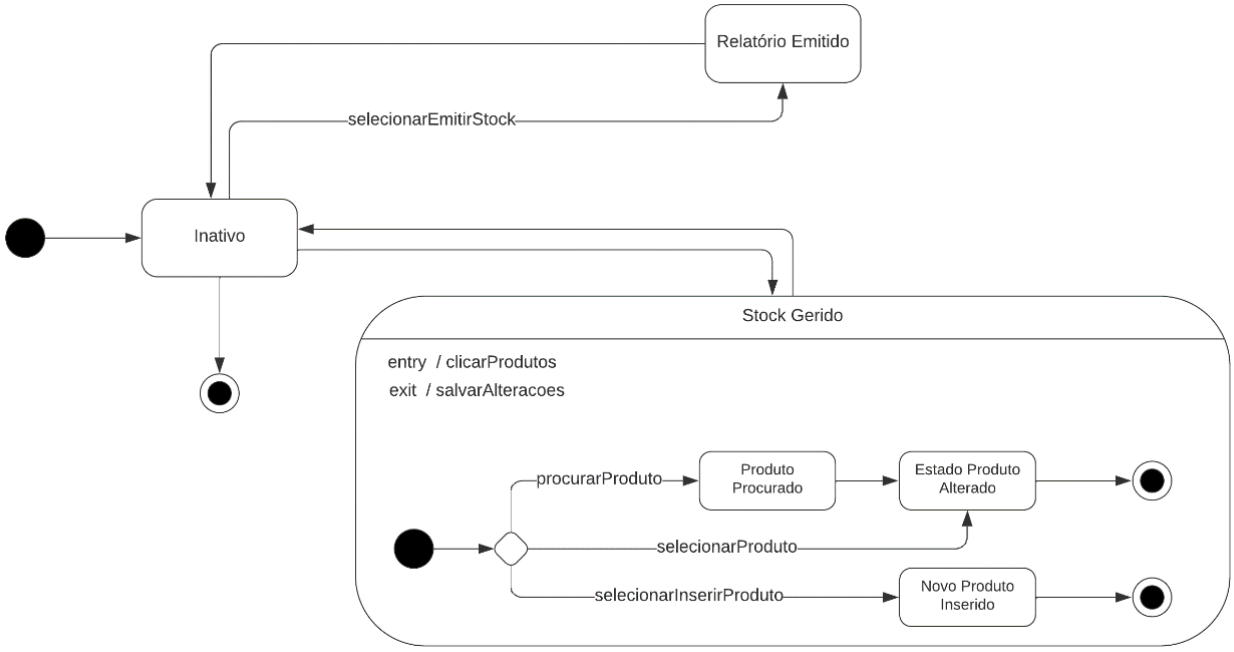
Diagrama de Estados

O diagrama de estados é uma representação visual que expõe vários estados entre os quais um sistema ou objeto pode existir, juntamente com as transições entre estados nos quais são desencadeados por eventos ou ações. Este diagrama integra caixas e círculos para denotar estados e utiliza setas para ilustrar as transições, indicando desta forma as mudanças comportamentais. Os diagramas de estado são usados tanto em engenharia de *software* como em modelação de sistemas e oferecem um meio conciso que permite a compreensão e conceção do comportamento dinâmico de processos e sistemas complexos.

Estes diagramas igualmente auxiliam na previsão e gestão de respostas do sistema relativamente a *inputs* e cenários diferentes, melhorando a adaptabilidade e fiabilidade do sistema, tornando o mesmo mais robusto e eficiente.

No que concerne à elaboração do diagrama, o grupo decidiu representar o sistema como um todo, uma vez, que não se justificaria a divisão destes pelos pacotes escolhidos nos diagramas de atividades e de sequência, dado que a representação dividida se tornaria redundante ao não apresentar características que permitem a compreensão e conceção do comportamento dinâmico de processos e sistemas complexos, e desta forma não cumpriria o objetivo deste tipo de representação visual.

Abaixo (**Figura 5**), encontra-se representado o diagrama de estados do software a ser desenvolvido.



(Figura 5)

Em primeira instância o diagrama evidencia a transição da entrada do software no qual o mesmo se encontra no seu estado inicial. Posteriormente haverá uma transição para o estado inativo da plataforma aquando da entrada do utilizador na mesma. A partir desse estado, poderá ser acionada uma transição ao ser selecionada a funcionalidade de emitir o relatório de stock. Após emitido o relatório, uma nova transição será desencadeada para retornar o programa ao seu estado inativo.

Outra transição que pode ser acionada a partir do estado inativo, desencadeia-se aquando de pressionar no botão “Produtos”, verificando-se o estado da funcionalidade fundamental da plataforma, alusiva à gestão de stock do restaurante.

Este estado de gestão do stock pode tomar três fluxos distintos. O primeiro será alusivo à alteração do estado do produto (edição e exclusão do mesmo), realizada após uma procura por um produto em específico. Este fluxo apresenta bastante similitude em relação ao segundo, pelo facto de este incidir igualmente pela alteração do estado do produto. O mesmo desta vez não será procurado, mas sim selecionado na lista apresentada após acionada a transição para o estado da gestão do stock. O terceiro (fluxo) corresponde ao estado da inserção de um novo produto, na qual se encontra separado da edição e

da exclusão por via de possuir dinâmicas distintas das demais onde, por exemplo, o estado não é acionado após a procura de um produto em específico. Para desencadear o término do estado da gestão do stock e consequente transição para o retorno do estado inativo do programa, a ação de salvar as alterações terá de ser realizada. Por fim, a partir do estado inativo do programa poderá prosseguir para o estado final do mesmo, onde será encerrado.

Considerações finais do momento 2 (M2) e perspetivas para o momento 3 (M3)

Após a elaboração dos diagramas UML mencionados ao longo do presente relatório, a próxima etapa engloba a parte introdutória do terceiro momento de avaliação. Esta será focalizada na apresentação da solução tecnológica (concretização dos dados, controlo e interface), aliada à concretização do diagrama de classes e as consciências metodológicas (modelação e processo) dos discentes no âmbito do desenvolvimento de sistemas de informação. Em suma será elencado, as aprendizagens consolidadas após a elaboração das etapas do segundo momento, concluiu-se do mapeamento gerir um projeto de
No que concerne, ao desenvolvimento da solução tecnológica, o grupo irá realizar a projeção do esquema da eventual base de dados. Consequentemente, será realizada a escolha das tecnologias a serem implementadas para o desenvolvimento do respetivo software, nomeadamente o Java para a programação e o MySQL para a base de dados, em consideração a combinação e compatibilidade entre as duas tecnologias. Após esta etapa, será realizado a configuração do *backend e a frontend* da solução tecnológica, a realização de testes de prototipagem e a sinalização de bugs que podem ocorrer aquando da realização de testes.

Enquadramento Bibliográfico do SWEBOK

Requisitos do Software

No que concerne aos requisitos, esta área de conhecimento do SWEBOK, (…) pretende auxiliar na definição de requisitos, como tornar possível a resolução de problemas reais, desde tipos produto vs processo e funcional vs não funcional. (…) É também possível analisar modelos de processo, os atores envolvidos, suporte, gestão e qualidade do processo. (Talib, M., Khelifi, A., & Jololian, L., 2010, pp. 4-6)

Por outro lado, existem ainda diferentes diretrizes que são seguidas. A primeira destaca a necessidade de negociação de compensações, de forma a se encontrarem com as limitações técnicas, orçamentos, visto que nem sempre é possível satisfazer todos as exigências das partes interessadas. Quanto à segunda diretriz, esta refere-se à conexão entre as atividades do processo com os recursos necessários para a sua realização (custos, recursos humanos e formação). Por fim, a terceira destaca a melhoria dos processos, utilizando modelos e padrões para satisfazer o cliente. (Talib, M., Khelifi, A., & Jololian, L., 2010, pp. 4-6)

É importante salientar que os requisitos de software são uma área crucial para o ciclo de vida do desenvolvimento de sistemas. Inclui a elicitação e análise de requisitos, denominando as necessidades dos stakeholders de forma a se tornarem mais claras e evidentes. (Bourque, P., & Fairley, R. E. (Eds.), 2014, pp. 36). Para a gestão de requisitos, são estabelecidos processos para monitorizar possíveis mudanças durante o ciclo de vida do software. Para além disso, a validação e verificação são importantes para

assegurar a correta implementação do software e, que este atenda às necessidades das partes interessadas. (Bourque, P., & Fairley, R. E. (Eds.), 2014, pp. 42-43)

Finalmente, a gestão de mudanças, pressupõe alterações dos requisitos que acontecem no decorrer do processo de desenvolvimento, auxiliando assegurar a integridade do sistema. Para o funcionamento claro de todos estes parâmetros é essencial manter uma comunicação eficaz de forma que todas as partes interessadas se encontrem informadas das diferentes ocorrências. (Bourque, P., & Fairley, R. E. (Eds.), 2014, pp. 44-45). Dentro desta finalidade, também se destacam a rastreabilidade de requisitos, prototipagem e engenharia de requisitos como ferramentas valiosas para validação e refinamento. Os requisitos não funcionais, que abordam atributos como desempenho, usabilidade e segurança, são igualmente cruciais. (Bourque, P., & Fairley, R. E. (Eds.), 2014, pp. 43)

Uma compreensão sólida desses elementos é essencial para garantir a integridade e qualidade do software, visto que o conhecimento das diretrizes do SWEBOK pode aprimorar a eficácia dos projetos de engenharia de software. (Bourque, P., & Fairley, R. E. (Eds.), 2014, pp. 32)

Implementação no restaurante

A utilização de um sistema de gestão de stock é uma componente essencial num restaurante para assegurar a eficiência operacional e a qualidade dos serviços a serem prestados. Assim, para ser possível atender as necessidades do restaurante, foi desenvolvido um software que integra alguns princípios do SWEBOK. Em primeira instância, é efetuado um levantamento das necessidades do restaurante, de forma a entender as suas necessidades, bem como a sua procura. Por meio de análises é possível identificar as necessidades de inventário para assegurar corretos níveis de stock.

A gestão de requisitos permite a adaptação contínua do sistema às mudanças, garantindo a precisão e confiabilidade dos pedidos, bem como a atualização dos níveis de stock e a conformidade com normas. Posto isto, o software de gestão de stock proporciona um ambiente eficaz e adaptável, alinhado à procura específica do restaurante. Esta abordagem garante não apenas a precisão do controlo de stock, mas também facilita a análise estratégica para otimização de processos operacionais.

Conceção do Software

No que concerne à conceção do software, esta considera-se que é uma atividade intermediária entre o levantamento de requisitos e a construção de software. Considera-se que é uma área que envolve as soluções de arquitetura, interface e outras características do sistema. É um processo interativo através do qual os requisitos são traduzidos num documento para construção do software. Este domínio dedica-se à análise meticulosa dos requisitos do software com o objetivo primordial de elaborar uma descrição robusta da estrutura interna do software, servindo de alicerce para a sua posterior construção. Esta área não se limita apenas à escolha de tecnologias ou à definição de estruturas de dados, pois abrange a modelagem holística do sistema. Posto isto, utiliza ferramentas como diagramas de classe, diagramas de sequência e outro tipo de representações. Ao abordar a conceção de um sistema de gestão de stocks, a prioridade é traduzir os requisitos específicos do setor que sirva como alicerçar para a construção do software.

Esta fase não se restringe à escolha de tecnologias ou à definição de estruturas de dados. Pelo contrário, abrange a modelagem holística do sistema, incorporando elementos cruciais como a dinâmica de

gestão de stocks, a interação com sistemas de fornecedores e a integração com a gestão de pedidos e faturação. (A. Silberschatz, P.B. Galvin, and G. Gagne,Operating System Concepts, 8th ed., Wiley, 2008, pp. 76).

Para concretizar esta conceção, recorreu-se à modelação UML e foi concebido diagramas de caso de uso, diagramas de interação, diagramas de atividades, diagramas de estado e diagramas de classes. Os fluxogramas anteriormente mencionados não só facilitam o seu desenvolvimento, mas também permitem uma análise mais clara e pormenorizada do contexto específico do restaurante. (P. Clements et al., Documenting Software Architectures: Views and Beyond, 2nd ed., Pearson Education, 2010, pp. 73). Salienta-se que na prática, a qualidade desta fase é crucial para o sucesso do software e uma compreensão correta e uma modelagem apropriada das particularidades que asseguram um sistema eficiente e adaptado às necessidades do restaurante.

Descrição do Protótipo Java MySQL

Para viabilizar o planeamento da construção do protótipo, ressalta-se a importância da descrição da solução. Esta incide num sistema de Java e SQL. A escolha da combinação Java e SQL para implementar no sistema de gestão de stocks do restaurante é pelo facto da linguagem de programação Java, segue o paradigma orientado a objetos, facilitando a criação de código legível e modular. Por outro lado, a escolha da linguagem SQL foi pela sua facilidade de uso, especialmente para iniciantes, visto que a maior parte dos membros de grupo não detém *background* tecnológico. Ressalva-se que os discentes, estiveram em contacto com a linguagem SQL, ou seja, foram introduzidos ao manuseamento de bases de dados noutra unidade curricular integrada no plano de estudos. Para a linguagem SQL (Structured Query Language), esta é utilizada para interagir com o programa MySQL, sendo esta intuitiva e amplamente adotada.

Criação da Base de Dados

A criação da base de dados ocorre durante a fase inicial do desenvolvimento do protótipo. Esta é procedida no sistema MySQL para dar suporte a um software de gestão de stocks. Nesse sentido, definiu-se quatro entidades: categoria, produto, utilizador e fornecedores. Salienta-se que a estrutura da base de dados foi meticulosamente planeada para abranger as necessidades específicas de cada entidade.

No que concerne à entidade "categoria", esta foi projetada com dois atributos principais: “id" (chave primaria) para identificação única e "descricao" para a descrição da categoria em questão. Posteriormente criou-se uma entidade "utilizadores", onde encontram-se atributos como “id” (chave primária) , ”nome", "tipo_utilizador", "senha", "estado" e "ultimo_login". As chaves primárias anteriormente mencionadas foram incorporadas para representar informações relevantes sobre os utilizadores do sistema.

Por outro lado, a categoria “produtos”, foi modelada com uma tabela que contêm atributos como “id" (chave primaria), "nome", "descrição", "preco_unitario", "quantidade", "categoria_id" (chave estrangeira da tabela categorias), "fornecedor_id" (chave estrangeira da tabela fornecedores), "utilizador_id" (chave estrangeira da tabela utilizadores) e "data_hora_registo".

Em referência à entidade "fornecedores", incorporam-se atributos como “id" (chave primaria), "designação", "contato", "contribuinte", "codPostal", "localidade", "morada", "eMail" e "telefone". Posto isto, a estrutura da base de dados encontra-se estabelecida. Consequentemente, concebeu-se a fase de povoamento, onde foram inseridos dados relevantes para cada entidade.

Definição de entidades e respetivas tabelas na base de dados

No que concerne à descrição da primeira etapa, salienta-se através da programação é expectável que o software seja acessível e intuitivo na ótica do utilizador e, que estes consigam introduzir manualmente informações sobre os produtos, incluindo nome, descrição, quantidade inicial, data de validade e o seu respetivo fornecedor. Posto isto, definiu-se as entidades e as suas respetivas tabelas, com a finalidade de os utilizadores lerem, alterarem, adicionarem e removerem os itens da base de dados. Para além disso, os utilizadores devem ser capazes de visualizar informações acerca de cada fornecedor. Consequentemente, o sistema permite que o utilizador faça encomendas aquando os níveis de stock de alimentos e ingredientes atingem o limite mínimo predefinido.

Por outro lado, também definiu-se que o sistema deve proporcionar a funcionalidades ao utilizador para que este consiga atualizar as movimentações de stock, como compras, vendas, transferências entre locais, desperdício e perda. Estas funcionalidades permitem o auxílio na elaboração de relatórios detalhados acerca da quantidade exata do stock, bem como as movimentações, custos associados e futuras tendências de compra. Neste permite que seja possível exportar a documentação para outros formatos.

Para a concretização da acessibilidade, ou seja, para que o software seja acessível, no momento de programação, foi levado em consideração a facilidade de o sistema corresponder da melhor forma aos utilizadores. Um objetivo na hora da programação, foi quando um funcionário estiver a manusear o programa, e este garantir-lhe que as «janelas» carreguem em menos de 2 segundos, traduzindo-se assim numa melhor experiência de utilização. Este processo tornar-se eficiente e concretizável por causa da interface do utilizador ser intuitiva.

Assim sendo (…) o sistema deverá ser capaz de lidar com os diferentes *stakeholders* em simultâneo (patrão e gerente, estes podem alterar o que entenderem e aceder sem qualquer restrição, a equipa operacional que lida diretamente com o stock, só poderá visualizar, adicionar ou remover). Em referência, à visualização da lista de fornecedores só quadros superiores da organização terão acesso, porque as credenciais de utilizador estão armazenadas com criptografia.

No que concerne à criptografia, as credenciais dos diferentes utilizadores (funcionários, gerente, chefe) serão protegidos por uma *password* de administrador que apenas o patrão e o gerente terão acesso, ou seja, este mecanismo permite que os dados do estabelecimento estejam salvaguardados de terceiros.

Salienta-se que a disponibilidade do sistema ocorre no horário laboral, sendo que o primeiro contacto com o sistema é feito pelo patrão, podendo aceder a qual informação relacionada com stock a qualquer momento, incluindo durante as operações de maior afluência, sem interrupções no serviço.

Criação da Interface

Após a criação da base de dados, o enfoque será voltado para a construção da interface. Esta será na linguagem de programação Java, que se conectará com uma base de dados em SQL, permitindo ao utilizador, criar, ler inserir, consultar novos produtos, atualizar informações do stock e puder apagar produtos da base de dados. A interface será intuitiva e fácil de usar, para que os três tipos de utilizadores possam realizar as operações necessárias de forma rápida e eficiente. Por outro lado, para a concretização do código Java será utilizado programa NetBeans (ambiente de desenvolvimento integrado - IDE). Para que o programa NetBeans consiga conectar-se à base de dados, é necessário criar uma “interface de java” onde é utilizado o `Connection` da biblioteca de java `java.sql.Connection;` que estabelece a ligação entre o `java` e o `sql`.

É importante salientar que a interfaces não são classes, mas são apenas declarações sintáticas de expressão de comportamento. Não estão na mesma hierarquia das classes e possuem uma hierarquia própria de herança múltipla. Este mecanismo deverá orientar o comportamento das classes, que adotarem a interface. Neste sentido, estas classes desempenham funções essenciais na manipulação de dados das tabelas presentes na base de dados. Em contrapartida, invés do uso direto de *queries* MySQL, a responsabilidade pela edição, inserção e remoção de dados, será das classes Java, que possibilitará a execução dessas ações pelo programa. Relativamente à uniformidade estrutural das classes, estas mantêm-se consistente em todos os contextos, como na gestão de utilizadores, categorias, entre outros. A distinção entre elas reside na invocação de classes específicas e nas instruções SQL armazenadas em variáveis do tipo *string*. Neste segmento focalizou-se a implementação do pacote DAO, responsável pela intermediação dos dados com a base de dados, inicialmente desenvolvendo a classe UtilizadorDAO para capturar informações e encaminhá-las à base de dados.

Fundamentação teórica das linguagens utilizadas

MySQL

Pereira & Poupa (2011), citados por António (2013) referem que (…)"O MySQL50 é um sistema de gestão de base de dados relacionais, ou seja, um sistema informático com o objetivo de armazenar dados de forma bem organizada e com baixa redundância" . Tal como os outros sistemas de base de dados atuais, funciona como um sistema cliente-servidor, onde o programa servidor é responsável por armazenar e permitir a manipulação de dados. Os programas cliente possibilitam o acesso à base de dados do ponto de vista lógico, e comunicam com o servidor utilizando a linguagem SQL51, concretamente o ANSI/ISO SQL standard, para manipulação dos dados e das estruturas onde são armazenados (Pereira & Poupa, 2011).

Java

De acordo com Horstmann e Cornell (2010), Java é uma linguagem de programação orientada a objetos, integrante da plataforma Java, livre e gratuita, que começou a ser desenvolvida em 1991 por um grupo de engenheiros da Sun Microsystems, liderados por Patrick Naughton e James Gosling. Salienta-se que a utilização do paradigma orientado a objetos traz como benefício o reuso de artefactos produzidos no desenvolvimento de softwares. Esse paradigma é utilizado na linguagem de programação Java, agregando simplicidade, robustez, segurança, portabilidade e ótimo desempenho.

Fundamentação teórica do padrão de conceção arquitetural

Conceptualização de MVC

É um padrão de conceção arquitetural amplamente utilizado no desenvolvimento de software. O MVC organiza o código-fonte de uma aplicação em três componentes principais, cada um com responsabilidades específicas. As componentes são constituídas pelo **modelo (model)**, que representa a camada de dados e lógica de negócios da aplicação, além de gerir o acesso e a manipulação dos dados, respondendo a consultas sobre o estado do sistema e atualizando-se conforme necessário. A segunda componente refere-se à **visão (view),** que representa a camada de apresentação da aplicação. Esta

componente, também é responsável por exibir as informações ao utilizador e interagir com ele. É importante salientar que a view obtém os dados do modelo e os apresenta de uma maneira compreensível para o utilizador, além de encaminhar as interações do utilizador para o **controlador (controller).** Este constituiu a terceira componente e atua como intermediário entre o modelo e a visão. Ele recebe eventos da interface do usuário (por exemplo, cliques em botões) da visão, processa esses eventos e, se necessário, atualiza o modelo de acordo. O **Controller** também pode receber informações do Modelo e atualizar a Visão de acordo.

Modelação UML - Diagrama de classes

Um diagrama de classes descreve uma estrutura estática das classes existentes num sistema, através da representação das entidades e de classes usadas no protótipo. No entanto, o mesmo descreve os comportamentos e os vários tipos de relacionamentos que as classes possuem entre si (UML Toolkit). Para alcançar a concretização da solução tecnológica, será utilizada uma linguagem de programação orientada a objetos, como o Java. Para o desenvolvimento da mesma solução tecnológica, deve ser implementado uma classe diretamente, a fim de tornar o respetivo diagrama num dos diagramas-chave para tornar a concessão de uma solução tecnológica exequível e viável. No que concerne à criação de um diagrama de classes, será necessário realizar uma identificação e descrição dos itens mais relevantes. Posteriormente será necessário definir os relacionamentos entre ambas. (UML Toolkit 90).

Ressalva-se que as classes podem relacionar-se entre si das seguintes formas: através da associação (conexão entre si), dependência (uma classe depende de outra ou utiliza outras para o seu funcionamento), especialização (uma classe é uma especialização de outra) ou da agregação (agregadas entre si como uma unidade) (UML Toolkit 25).

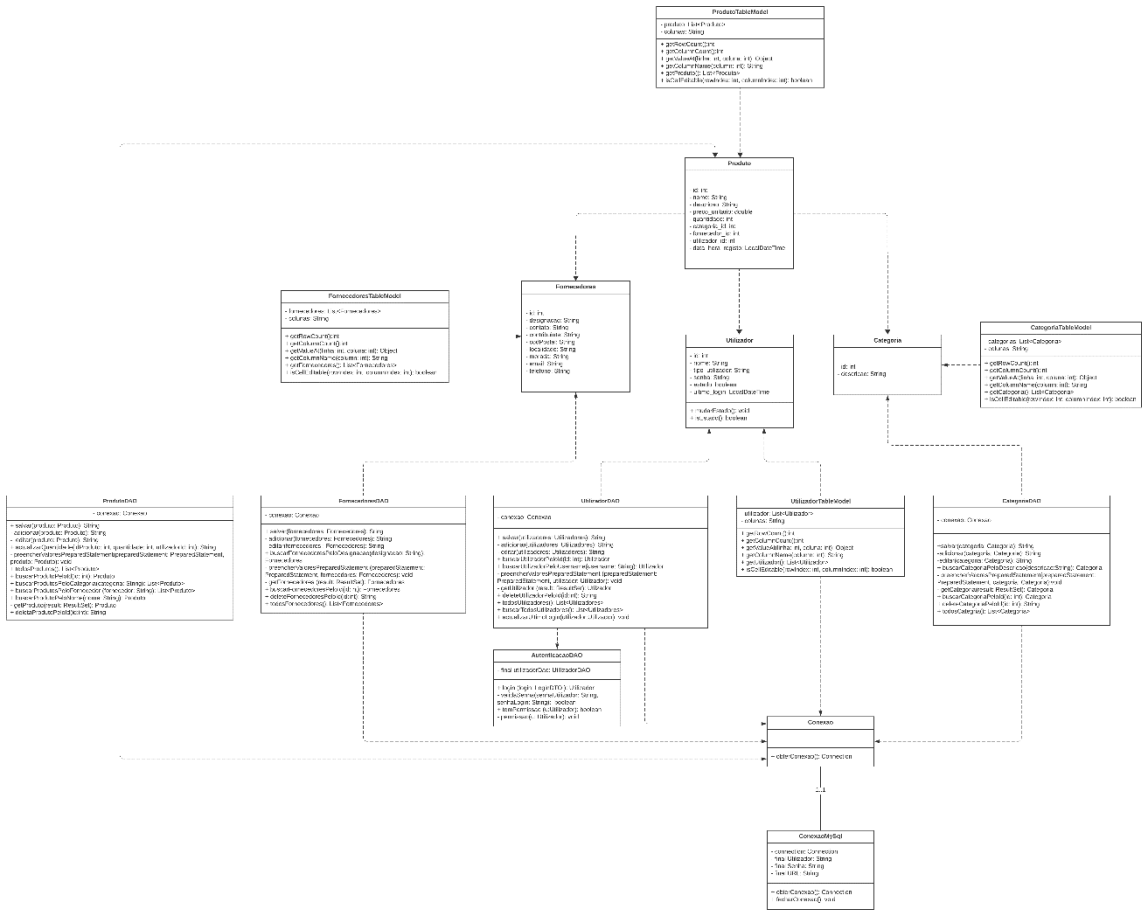
Posto isto, verifica-se que estes relacionamentos são representados no diagrama de classes, juntamente com a estrutura interna das classes. O diagrama é considerado estático na medida em que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema. Cada sistema possui tipicamente um número de diagramas de classes – não há a inserção de todas as classes do sistema num só diagrama – e uma classe pode participar em diversos diagramas de classes (UML Toolkit 25).

No presente relatório, foi dividido os diagramas em 3 pacotes, seguindo a lógica do padrão de conceção arquitetural MVC *(Model-View-Controller)*.

Diagrama de Classe Model

No primeiro diagrama, visualiza-se o diagrama de classe “Model”, onde se encontra representada a gestão do acesso e a manipulação dos dados, através das classes de domínio, de manipulação (…DAO), as classes TabelModel e as classes modelo representadas neste diagrama, que estabelecerão a ligação com a base de dados.

Abaixo, (**Figura 6**) é ilustrado a representação diagramática das classes Model no protótipo a desenvolver.



(Figura 6)

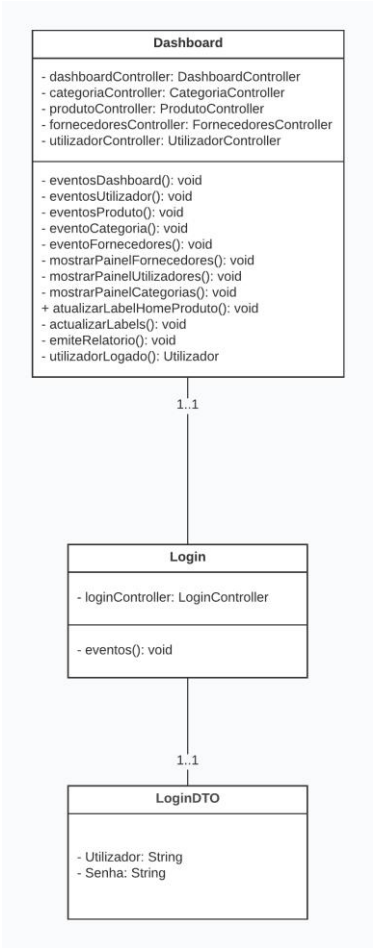
O centro do diagrama é constituído pelas classes de Domínio (Fornecedores, Categoria, Utilizador, Produto), que representam as tabelas que a base de dados (BD) contém. O relacionamento entre ambas é estabelecido através da classe Produto, na qual é dependente de todas as outras classes de Domínio. Posteriormente verifica-se que classes DAO, que são responsáveis pela manipulação dos dados das tabelas através de métodos. O relacionamento entre as classes é estabelecido pela dependência que as mesmas possuem com as classes de Domínio, nas quais farão posteriormente a manipulação. No caso da classe “Autenticacao DAO”, esta é dependente de outra classe DAO (Utilizador DAO).

Em referência às classes TabelModel, verifica-se que fornecem um modelo de dados em forma de tabela para representar e exibir informações sobre as respetivas tabelas na interface gráfica. Da mesma forma como as classes DAO, estas são dependentes da sua respetiva classe de Domínio.

Por fim, evidencia-se as classes Modelo que estabelecem a conexão com a BD desenvolvida em MySQL (Conexao, ConexaoMySql). Estas relacionam-se com as restantes classes através do relacionamento de dependência que as classes DAO (FornecedoresDAO, CategoriaDAO, UtilizadorDAO, ProdutoDAO) possuem, sobre a classe Conexao e do relacionamento de associação entre as classes Conexao e ConexaoMySql.

Diagrama de Classe View

No seguinte diagrama, representa o diagrama de classe “View”. Encontra-se exibido tudo aquilo que o utilizador visualiza no ecrã através das classes de formulário e da classe LoginDTO.
Abaixo **(figura 7)**, é ilustrado a representação diagramática das classes “View” no protótipo a desenvolver.



(Figura 7)

Relativamente aos relacionamentos, as classes de formulário (Dashboard e Login), responsáveis pela criação e funcionamento das janelas da interface que o utilizador irá manipular, relacionam-se entre si através de um relacionamento de associação. Relativamente à classe LoginDTO, que é utilizada para representar dados de entrada no sistema através da interface de login, a mesma relaciona-se com a classe Login através de uma associação.

Diagrama de Classe Controller

O diagrama de classe “controller”, representa a gestão de fluxo e a interação entre o modelo e a visão. É responsável por gerir a lógica de controlo associada às respetivas entidades e têm como principais responsabilidades a gestão de eventos de interface do utilizador, interação com a interface gráfica, comunicação com o modelo, validação e manipulação de dados, atualização das tabelas e controlo de permissões.

Abaixo, **(figura 8)** verifica-se a representação diagramática das classes Controller no protótipo a desenvolver.

(Figura 8)

Os relacionamentos neste diagrama de classes são estabelecidos através do relacionamento de dependência das Classes FornecedoresController, CategoriaController, RelatórioController e ProdutoController em relação à classe UtilizadorController e pelo relacionamento de dependência que a Classe DashboardController possui com todas as restantes classes.

V&V | Verificação e Validação dos requisitos funcionais e Não-Funcionais

Requisitos funcionais:

Figura 9 – Verifica-se que o sistema permite que os utilizadores criem novos produtos"

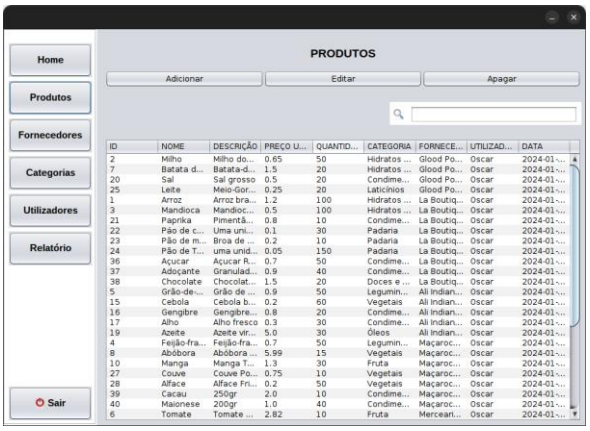
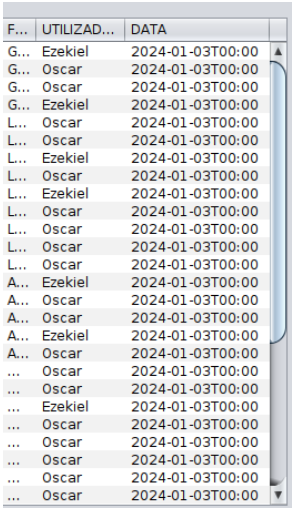


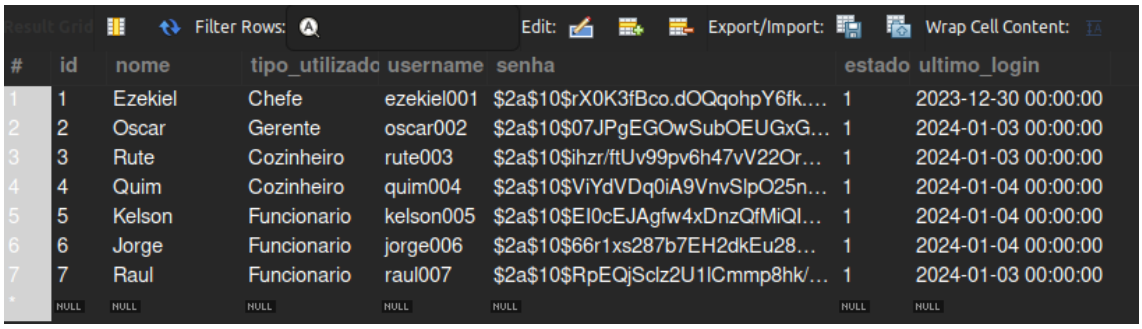
Figura 10 – Verifica-se que o sistema permite que os utilizadores leiam, atualizem e excluam novos produtos



F...	UTILIZAD...	DATA
G...	Ezekiel	2024-01-03T00:00:00
G...	Oscar	2024-01-03T00:00:00
G...	Oscar	2024-01-03T00:00:00
G...	Ezekiel	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
L...	Ezekiel	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
L...	Ezekiel	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
L...	Oscar	2024-01-03T00:00:00
A...	Ezekiel	2024-01-03T00:00:00
A...	Oscar	2024-01-03T00:00:00
A...	Oscar	2024-01-03T00:00:00
A...	Ezekiel	2024-01-03T00:00:00
A...	Oscar	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00
...	Ezekiel	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00
...	Oscar	2024-01-03T00:00:00

Figura 11 – Verifica-se a funcionalidade de visualizar o utilizador que editou as informações, bem o como o dia e as horas.

Requisitos não funcionais:



#	id	nome	tipo_utilizado	username	senha	estado	ultimo_login
1	1	Ezekiel	Chefe	ezekiel001	$2a$10$rX0K3fBco.dOQqohpY6fk....	1	2023-12-30 00:00:00
2	2	Oscar	Gerente	oscar002	$2a$10$07JPgEGOwSubOEUGxG...	1	2024-01-03 00:00:00
3	3	Rute	Cozinheiro	rute003	$2a$10$ihzr/ftUv99pv6h47vV22Or...	1	2024-01-03 00:00:00
4	4	Quim	Cozinheiro	quim004	$2a$10$ViYdVDq0iA9VnvSlpO25n...	1	2024-01-04 00:00:00
5	5	Kelson	Funcionario	kelson005	$2a$10$El0cEJAgfw4xDnzQfMiQI...	1	2024-01-04 00:00:00
6	6	Jorge	Funcionario	jorge006	$2a$10$66r1xs287b7EH2dkEu28...	1	2024-01-04 00:00:00
7	7	Raul	Funcionario	raul007	$2a$10$RpEQjSclz2U1lCmmp8hk/...	1	2024-01-03 00:00:00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 12 – Verifica-se que o sistema consegue crifar as informações confidenciais dos funcionários, do gerente e do chefe

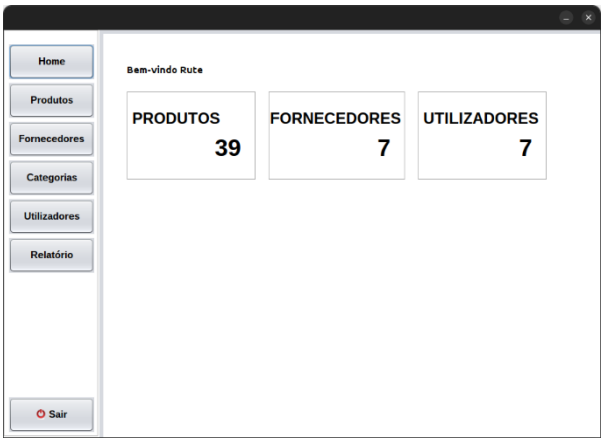
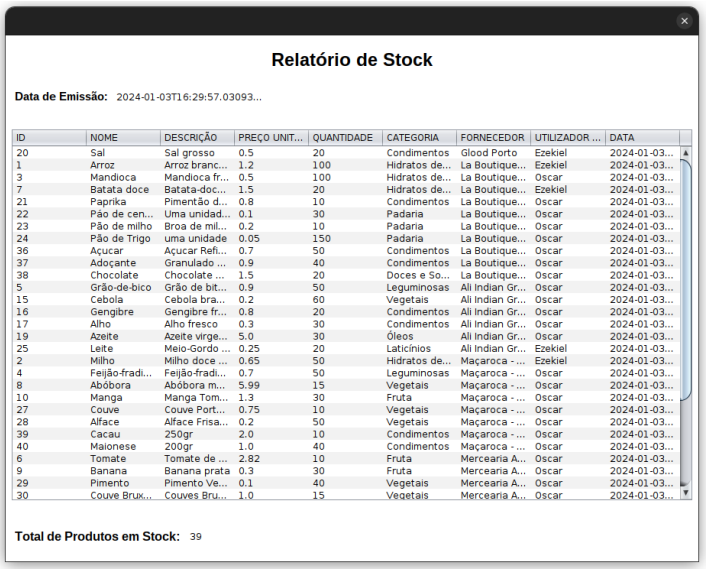


Figura 13 – Verifica-se que o programa é fácil de usar. (Ressalva-se que facilidade de utilização advém de o programa ser intuitivo de manusear e, sua a estética ser simples na ótica do utilizador)



ID	NOME	DESCRIÇÃO	PREÇO UNIT...	QUANTIDADE	CATEGORIA	FORNECEDOR	UTILIZADOR ...	DATA
20	Sal	Sal grosso	0.5	20	Condimentos	Glood Porto	Ezekiel	2024-01-03...
1	Arroz	Arroz branc...	1.2	100	Hidratos de...	La Boutique...	Ezekiel	2024-01-03...
3	Mandioca	Mandioca fr...	0.5	100	Hidratos de...	La Boutique...	Oscar	2024-01-03...
7	Batata doce	Batata-doc...	1.5	20	Hidratos de...	La Boutique...	Ezekiel	2024-01-03...
21	Paprika	Pimentão d...	0.8	10	Condimentos	La Boutique...	Oscar	2024-01-03...
22	Pão de cen...	Uma unidad...	0.1	30	Padaria	La Boutique...	Oscar	2024-01-03...
23	Pão de milho	Broa de mil...	0.2	10	Padaria	La Boutique...	Oscar	2024-01-03...
24	Pão de Trigo	uma unidade	0.05	150	Padaria	La Boutique...	Oscar	2024-01-03...
36	Açucar	Açucar Refi...	0.7	50	Condimentos	La Boutique...	Oscar	2024-01-03...
37	Adoçante	Granulado ...	0.9	40	Condimentos	La Boutique...	Oscar	2024-01-03...
38	Chocolate	Chocolate ...	1.5	20	Doces e So...	La Boutique...	Oscar	2024-01-03...
5	Grão-de-bico	Grão de bit...	0.9	50	Leguminosas	Ali Indian Gr...	Oscar	2024-01-03...
15	Cebola	Cebola bra...	0.2	60	Vegetais	Ali Indian Gr...	Oscar	2024-01-03...
16	Gengibre	Gengibre fr...	0.8	20	Condimentos	Ali Indian Gr...	Oscar	2024-01-03...
17	Alho	Alho fresco	0.3	30	Condimentos	Ali Indian Gr...	Oscar	2024-01-03...
19	Azeite	Azeite virge...	5.0	30	Óleos	Ali Indian Gr...	Oscar	2024-01-03...
25	Leite	Meio-Gordo ...	0.25	20	Laticínios	Ali Indian Gr...	Ezekiel	2024-01-03...
2	Milho	Milho doce ...	0.65	50	Hidratos de...	Maçaroca - ...	Ezekiel	2024-01-03...
4	Feijão-fradi...	Feijão-fradi...	0.7	50	Leguminosas	Maçaroca - ...	Oscar	2024-01-03...
8	Abóbora	Abóbora m...	5.99	15	Vegetais	Maçaroca - ...	Oscar	2024-01-03...
10	Manga	Manga Tom...	1.3	30	Fruta	Maçaroca - ...	Oscar	2024-01-03...
27	Couve	Couve Port...	0.75	10	Vegetais	Maçaroca - ...	Oscar	2024-01-03...
28	Alface	Alface Frisa...	0.2	50	Vegetais	Maçaroca - ...	Oscar	2024-01-03...
39	Cacau	250gr	2.0	10	Condimentos	Maçaroca - ...	Oscar	2024-01-03...
40	Maionese	200gr	1.0	40	Condimentos	Maçaroca - ...	Oscar	2024-01-03...
6	Tomate	Tomate de ...	2.82	10	Fruta	Mercearia A...	Oscar	2024-01-03...
9	Banana	Banana prata	0.3	30	Fruta	Mercearia A...	Oscar	2024-01-03...
29	Pimento	Pimento Ve...	0.1	40	Vegetais	Mercearia A...	Oscar	2024-01-03...
30	Couve Brux...	Couves Bru...	1.0	15	Vegetais	Mercearia A...	Oscar	2024-01-03...

Figura 14 – Verifica-se que que o sistema fornece relatórios ao chefe

De acordo com *screenshots* supramencionados, os requisitos funcionais foram preenchidos na integra e o programa está robusto, pois comporta-se com imensa razoabilidade. Por outro lado, no que concerne aos requisitos não-funcionais foram preenchidos quase na totalidade, exceto na emissão de *download* dos relatórios.

A ideia inicial de emissão de relatório pensada pelo grupo seria feita em duas versões: emissão através de uma janela do programa ou *download* do relatório assim que este fosse emitido. Porém, devido a vários impasses que foram surgindo ao longo do desenvolvimento do programa (como erros de código, falhas nos testes, verificação de funcionalidades, entre outros), a elaboração da funcionalidade da emissão do relatório foi deixada para último plano, colidindo com os dias finais para a realização do trabalho. Assim, a questão da emissão por *download*, bem como certas informações que deveriam constar do relatório, não foram conseguidas, sendo que respetivo requisito não foi atendido na totalidade.

Relato do percurso do desenvolvimento do software

O percurso do desenvolvimento do software integra as etapas da definição de entidades e respetivas tabelas na base de dados, povoamento da base de dados, conexão com a base de dados do MySQL, criação das Classes de Modelo (ClassesDAO, Classes de Domínio, Classes TabelModel, Classe Conexao e Classe ConexaoMySql), criação de Classes de visão (classe LoginDTO), criação das Classes de controlo (classes Controller), proteção das senhas dos utilizadores, autenticação e permissão, formulário de Login e formulário da Dashboard.

É importante salientar que o desenvolvimento do software se consta em anexo. Ressalva-se que o desenvolvimento do software foi baseado numa arquitetura MVC. No processo de prototipagem foi necessário a criação de classes que pertencessem às componentes **modelo (model), visão (view)** e **controlo (controller).** A componente modelo é responsável por gerir o acesso e a manipulação dos dados, respondendo a consultas sobre o estado do sistema e atualizando-se conforme necessário. As classes que foram criadas pelos membros de grupo, alicerçam-se nas **classes de domínio**, ou seja, aludem à representação das tabelas que a base de dados contém (Utilizador, Produto, Categoria e Fornecedores).

Por outro lado, verifica-se as classes DAO são responsáveis pela manipulação dos dados dessas tabelas (UtilizadorDAO, CategoriaDAO, ProdutoDAO e FornecedoresDAO) através de métodos criados como o editar(), salvar(), adicionar(), etc.

Em referência às **Classes TabelModel** (UtilizadorTableModel, FornecedoresTableModel, ProdutoTableModel, CategoriaTableModel), estas estendem a classe “AbstratctTableModel” e são utilizadas para fornecer um modelo de dados em forma de tabela para representar e exibir informações sobre as respetivas tabelas na interface gráfica. Por outro lado, as classes de modelo permitem estabelecer a conexão com a base de dados desenvolvida no programa MySQL (Conexao.java e ConexaoMySql.java).

Em referência à componente **visão,** esta representa a camada de interface do utilizador, ou seja, aquilo que o utilizador visualiza no ecrã. Para esta camada foram desenvolvidas as **classes de formulário** (Login.java e Dashboard.java) que são responsáveis pela criação e funcionamento das janelas da interface que o utilizador irá manipular e a **Classe LoginDTO** é utilizada para representar os dados de entrada no sistema através da interface de login (Login.java), possuindo como instâncias o nome de utilizador (username) e a senha.

No que concerne à componente **controlo**, estas classes caracterizam-se pela gestão o fluxo e pela interação entre o modelo e a visão. A componente controlo responde a eventos e atualiza a visão com base nas mudanças na componente modelo. As classes que foram desenvolvidas para esta componente foram UtilizadorController, ProdutoController, FornecedoresController, CategoriaController, LoginController, e DashboardController. As classes anteriormente mencionadas são responsáveis por gerir a lógica de controlo associadas às respetivas entidades (como por exemplo, a classe UtilizadorController gere a lógica de controlo associada à entidade Utilizador) e têm como principais responsabilidades a gestão de eventos de interface do utilizador, interação com a interface gráfica, (no caso deste protótipo é a Dashboard), comunicação com o modelo, validação e manipulação de dados, atualização das tabelas e controlo de permissões.

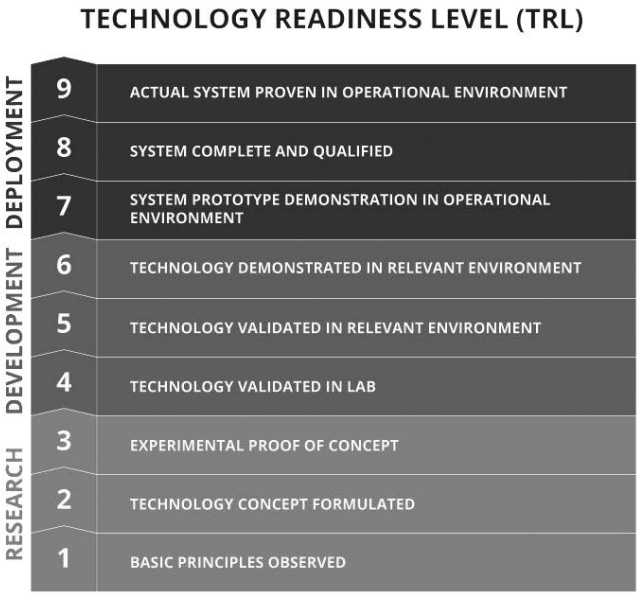
Technology Readiness Level (TRL)

Com os ganhos de performance alcançados através da utilização de uma nova tecnologia, igualmente advém um grau de risco e incerteza associada às capacidades, limitações e trajetória de desenvolvimento (Olechowski,2015). Para que se consiga colmatar o alastramento de consequências negativas provenientes da não prontidão a tempo das tecnologias, uma melhor compreensão do estado da

maturidade da tecnologia é fulcral na tomada de boas decisões relativas a injeção, ao desenvolvimento e à integração destas tecnologias em projetos complexos de engenharia (Olechowski,2015).

A escala do Nível de Prontidão da Tecnologia (*TRL – Technology Readiness Level*) foi introduzido pela NASA nos anos 70 como uma ferramenta de suporte para a avaliação da maturidade de tecnologias ao decorrer do desenvolvimento de sistemas complexos. Atualmente, esta escala tornou-se numa norma para a avaliação e supervisão de tecnologias em várias indústrias, incluindo a de tecnologias de informação, na qual o presente projeto se enquadra. (Olechowski,2015)

Abaixo (figura 15) é representado o TRL adaptado pela União Europeia nos seus projetos de inovação, na qual se considerou que se encontra mais adaptado à realidade e indústria do protótipo em estudo do que a escala originalmente utilizada pela NASA.



(Figura 15)

Tendo por base todos os processos de desenvolvimento praticados anteriormente a este capítulo do relatório, concluiu-se que o protótipo em questão se enquadra no nível 6, que é relativo à prontidão da sua tecnologia, uma vez que o sistema prototipado, encontrou-se produzido e demonstrado num ambiente simulado – internamente com os membros envolvidos na conceção e desenvolvimento do sistema.

Conclusão: Competências adquiridas ao longo da UC

Ao longo do presente semestre na Unidade Curricular de Engenharia de Software de Sistemas de Informação, o docente transmitiu conhecimentos tecnológicos e fomentou nos alunos fundamentos teóricos para que estes percebessem o funcionamento da UC. Salienta-se que enquanto equipa obteu-se aprendizagens enriquecedoras para aplicar no projeto, acerca do SWEBOK Guide, como o desenvolvimento do software e o seu ciclo de vida, mitos e realidades de um gestor de projetos, qualidade no Software, princípios da engenharia de software e a modelação com o UML. Estas aprendizagens foram uma mais-valia para o ingresso no mercado de trabalho no âmbito da integração de tecnologias e sistemas de informação nas organizações.

No que concerne às principais dificuldades sinalizadas na elaboração do projeto em grupo poderá ter sido a gestão de tempo. Considera-se que esta pode sido ser uma dificuldade, especialmente quando os membros do grupo têm outras responsabilidades e compromissos. No entanto, marcaram-se prazos e datas de entrega e reuniões para a divisão e distribuição de tarefas.

Os membros do grupo tiveram ideias e opiniões diferentes sobre os diferentes diagramas UML que deveriam ser criados, no entanto, com a orientação do docente, delineou-se um fio condutor e de forma como os diagramas deveriam ser construídos com coerência sobre como representar um determinado elemento do sistema. Assim os membros do grupo puderam aprender sobre as diferentes possibilidades de representação. Ou seja, verifica-se que processo de construção dos diagramas UML permitiu que os alunos desenvolvessem *softkills* de resolução de problemas e de pensamento crítico, porque ao analisar os diferentes elementos do sistema, foi necessário identificar as relações entre eles e encontrar a melhor forma de representá-las.

O processo de construção dos diagramas UML foi uma oportunidade valiosa para os membros do grupo aprenderem sobre uma ferramenta importante para o desenvolvimento de software, bem como os pressupostos de cada diagrama e representá-los de forma clara e concisa, salientando que através desta experiência, os discentes adquiriram novos conhecimentos e habilidades que serão úteis em projetos futuros.

Em referência a outras aprendizagens desenvolvidas verifica-se que os membros de grupo adquiriram mais conhecimento no âmbito da programação, que outrora não possuíam, o que se traduz em uma maior capacidade de compreender as temáticas alusivas à engenharia de software, para além de fomentar o grau de autonomia para aplicar técnicas de programação.

Porém, foram sinalizadas dificuldades sentidas pelo grupo ao longo de todo o processo de programação. Evidenciaram-se dificuldades no âmbito da compreensão conceitos complexos de programação, como orientação a objetos, algoritmos e na conexão nas estruturas de dados, devido à falta de experiência prática em programação, desencadeando assim erros e ineficiências no processo de escrever código. No entanto o desafiante processo de programação foi superado quase na totalidade através de esforço e dedicação, mas ainda assim evidenciou-se como uma lacuna relativa à pressão do tempo para o terceiro momento de avaliação, porém reforça-se que foi uma oportunidade de aprendizagem e crescimento para os membros do grupo.

No entendimento de todos os membros do grupo, os mecanismos utilizados para a superação das dificuldades, foi a importância de estabelecer uma comunicação aberta e eficaz, definir claramente as responsabilidades de cada membro e estabelecer prazos realistas e acompanhar de perto o progresso do trabalho. Além disso, foi importante ter flexibilidade para lidar com desafios inesperados e estar aberto a ajustes e mudanças conforme necessário. Para além disso, ressalva-se que através da comunicação e do diálogo aberto, destacou-se a concordância e o alinhamento entre todos os membros, relativamente aos métodos utilizados na prototipagem da solução tecnológica, na elaboração do suporte de apresentação e na estrutura da elaboração do documento do relatório de grupo, culminando assim num entendimento total e num sucesso para a apresentação do projeto final.

Ressalva-se que, o facto de a metodologia de avaliação ser realizada em grupo, foi uma valência, pois dotou os membros do grupo de competências, das quais gestão de tempo, a gestão de tarefas e aprimorou a comunicação entre todos os alunos. Para além da aquisição de *softskills,* o projeto em equipa aprimorou as *hardskills* dos discentes. Evidencia-se que a produção de software cresce devido aos avanços científicos e tecnológicos e as empresas tomam a decisão de automatizar os seus processos, em busca de competitividade, redução de custos e melhorias nos processos de negócio e, face a estes aspetos ressalva-se a acentuação da uma melhor preparação dos alunos para que num futuro próximo exerçam funções com

destreza e eficiência numa equipa de projeto para desenvolver software para um determinado contexto organizacional.

Ao longo dos diferentes momentos de avaliação verifica-se que os alunos foram capazes de conceber o problema fictício, identificar os requisitos do restaurante e dos *stakeholders*, realização de testes para verificar a viabilidade da qualidade no software, isto é, se o protótipo foi concebido com exequibilidade, e se os requisitos estipulados no momento 1 foram efetivamente concretizados, como também revisão formal do código. Por outro lado, adquiriu-se domínio no âmbito da conceção de diagramas UML (de casos de uso, atividades, sequência, estados, classes), bem como a sua validação no que concerne à arquitetura de software, isto é, se este é capaz de responder às necessidades funcionais e não-funcionais do problema do restaurante e estas foram verificadas e validadas e existiu adequação da solução técnica ao problema a resolver através de testes).

Posto isto, concluiu-se que face às aquisições técnicas, anteriormente mencionadas, os alunos, num futuro próximo estarão melhor preparados para o ingresso no mercado de trabalho enquanto profissionais de TI. Deverão ser capazes, de acordo com uma dada necessidade, aplicar uma abordagem sistemática, disciplinada e quantificável conducente ao desenvolvimento, operação e manutenção de um sistema de software que satisfaça os requisitos funcionais de uma organização na área da restauração ou similar, possuindo competências para analisar os pontos fortes e fracos de um protótipo para fomentar a mudança nas organizações.

ANEXOS

Tabelas

- Categoria (de produtos) → id, descrição.

```
CREATE TABLE Categoria(
    id INT NOT NULL,
    descricao VARCHAR(75) NOT NULL,
        PRIMARY KEY (id)
);
```

- Produtos → id, nome, descrição, categoria, quantidade, data validade, fornecedor (sugestão do bard: unidade de medida do produto, preço unitário, stock_minimo, stock_maximo)

```
CREATE TABLE Produto (
    id INT NOT NULL,
    nome VARCHAR(50) NOT NULL UNIQUE,
    descricao VARCHAR(200),
    preco_unitario DECIMAL(10,2) NOT NULL,
    quantidade INT NOT NULL,
    categoria_id INT NOT NULL,
    utilizador_id INT NOT NULL,
    data_hora_registo DATETIME DEFAULT current_timestamp,
        PRIMARY KEY (id),
        FOREIGN KEY (categoria_id) REFERENCES Categoria(id),
        FOREIGN KEY (utilizador_id) REFERENCES Utilizador(id)
);
```

- Utilizadores →id, tipo de utilizador (se é o chefe, gerente, ou funcionário), nome, senha, ultimo login no sistema.

```
CREATE TABLE Utilizador(
    id INT NOT NULL,
    nome VARCHAR(100) NOT NULL,
    utilizador VARCHAR(75) NOT NULL UNIQUE,
    senha VARCHAR(100) NOT NULL,
    tipo VARCHAR(75) NOT NULL,
    estado BOOLEAN NOT NULL DEFAULT TRUE,
    ultimo_login DATETIME DEFAULT '0001-01-01 00:00:00',
        PRIMARY KEY(id)
);
```

- Fornecedores

```
CREATE TABLE Fornecedores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    designação VARCHAR(100) NOT NULL,
    contato VARCHAR(200) NOT NULL,
    contribuinte VARCHAR(100),
    codPostal VARCHAR(75) NOT NULL,
    localidade VARCHAR(75) NOT NULL,
    morada VARCHAR(100) NOT NULL,
    eMail VARCHAR(150),
    telefone VARCHAR(50) NOT NULL
);
```

Povoamento da base de dados

```sql
INSERT INTO Categoria
  (id, descricao)
  VALUES
    ('01','Padaria'),
    ('02','Laticínios'),
    ('03','Óleos'),
    ('04','Vegetais'),
    ('05','Peixe e Marisco'),
    ('06','Carne e ovos'),
    ('07','Leguminosas'),
    ('08','Hidratos de Carbono'),
    ('09','Fruta'),
    ('10','Congelados Diversos'),
    ('11','Vinhos, Refrigerantes e Águas Minerais'),
    ('12','Doces e Sobremesas'),
    ('13','Condimentos');


INSERT INTO Utilizador
  (id, nome, tipo_utilizador, senha, estado, ultimo_login)
    VALUES
    ('001','Ezekiel', 'Chefe', 'FR34858', FALSE, '2023-12-30'),
    ('002', 'Oscar', 'Gerente', 'H1235', FALSE, '2024-01-03'),
    ('003', 'Rute', 'Cozinheira', 'Z0942', FALSE, '2024-01-03'),
  ('004', 'Quim',  'Cozinheiro', 'G3425', FALSE, '2024-01-04'),
  ('005', 'Kelson', 'Funcionario', 'X1343', FALSE, '2024-01-04'),
    ('006', 'Jorge', 'Funcionario', 'F3245', FALSE, '2024-01-04'),
  ('007', 'Raul Meireles', 'Funcionario', 'Q2334', FALSE, '2024-01-03');
```

```sql
INSERT INTO Produto
  (id, nome, descricao, preco_unitario, quantidade, categoria_id, fornecedor_id, utilizador
  VALUES
    ('01', 'Arroz','Arroz branco de grão fino', '1.20', '100', '08', '02', '002', '2024-01
    ('02', 'Milho', 'Milho doce enlatado', '0.65', '50', '08', '01', '002', '2024-01-03'),
    ('03', 'Mandioca', 'Mandioca fresca', '0.50', '100', '08', '02', '002', '2024-01-03'),
    ('04', 'Feijão-fradinho', 'Feijão-fradinho preto', '0.70', '50', '07', '04', '002', '2
    ('05', 'Grão-de-bico', 'Grão de bito seco', '0.90', '50', '07', '03', '002', '2024-01-
    ('06', 'Tomate', 'Tomate de rama com 3 unidades', '2.82', '10', '09', '05', '002', '20
    ('07', 'Batata doce', 'Batata-doce laranja', '1.50', '20', '08', '01', '002', '2024-01
    ('08', 'Abóbora', 'Abóbora moranga', '5.99', '15', '04', '04', '002', '2024-01-03'),
    ('09', 'Banana', 'Banana prata', '0.30', '30', '09', '05', '002', '2024-01-03'),
    ('10', 'Manga', 'Manga Tommy Atkins', '1.30', '30', '09', '04', '002', '2024-01-03'),
    ('11', 'Frango', 'Frango inteiro', '2.34', '10', '06', '06', '002', '2024-01-03'),
    ('12', 'Carne bovina', 'Carne bovina moída', '4.49', '10', '06', '06', '002', '2024-01
    ('13', 'Carne de porco', 'Cachaço inteiro sem osso', '4.61', '5', '06', '06', '002', '
    ('14', 'Peixe', 'Tilápia fresca', '1.50', '4', '05', '07', '002', '2024-01-03'),
    ('15', 'Cebola', 'Cebola branca', '0.20', '60', '04', '03', '002', '2024-01-03'),
    ('16', 'Gengibre', 'Gengibre fresco', '0.80', '20', '13', '03', '002', '2024-01-03'),
    ('17', 'Alho', 'Alho fresco', '0.30', '30', '13', '03', '002', '2024-01-03'),
    ('18', 'Óleo', 'Óleo de girassol', '1.60', '30', '03', '06', '002', '2024-01-03'),
    ('19', 'Azeite', 'Azeite virgem extra', '5.00', '30', '03', '03', '002', '2024-01-03'),
    ('20', 'Sal', 'Sal grosso', '0.50', '20', '13', '01', '002', '2024-01-03'),
    ('21', 'Paprika', 'Pimentão doce', '0.80', '10', '13', '02', '002', '2024-01-03'),
    ('22', 'Pão de centeio', 'Uma unidade de pão', '0.10', '30', '01', '02', '002', '2024-
    ('23', 'Pão de milho', 'Broa de milho uma unidade', '0.20', '10', '01', '02', '002', '
    ('24', 'Pão de Trigo','uma unidade','0.05','150', '01', '02', '002', '2024-01-03'),
    ('25', 'Leite','Meio-Gordo 1lt','0.25', '20', '02', '01', '002', '2024-01-03'),
    ('26', 'Presunto','9 meses cura 200gr', '3.50', '15', '06', '07', '002', '2024-01-03'),
    ('27', 'Couve', 'Couve Portuguesa 1un', '0.75', '10', '04', '04', '002', '2024-01-03'),
    ('28', 'Alface',  'Alface Frisada 1un', '0.20', '50', '04', '04', '002', '2024-01-03'),
    ('29', 'Pimento', 'Pimento Verde 1u', '0.10', '40', '04', '05', '002', '2024-01-03'),
    ('30', 'Couve Bruxelas', 'Couves Bruxelas 0.5kg', '1.00', '15', '04', '05', '002', '20
    ('31', 'Bróculos', 'Bróculos 1kg', '1.25', '10', '04', '05', '002', '2024-01-03'),
    ('32', 'Sumo Laranja', '1 litro', '0.90', '50', '11', '07', '002', '2024-01-03'),
    ('33', 'Água Mineral','Garrafas de 1 litro', '1.00', '100', '11', '07', '002', '2024-0
```

```sql
INSERT INTO Fornecedores
  (id, designação, contato, contribuinte, codPostal, localidade, morada, eMail, telefone)
  VALUES
  ('01', 'Glood Porto', 'Fernando Silva', NULL, '4050-438', 'Porto', 'Rua de Oliveira Mont
  ('02', 'La Boutique Africaine', 'Maria Domingos', NULL, '4350-014',  'Porto', 'PC Afonso
  ('03', 'Ali Indian Groceries', 'Crisley Santos', NULL, '4000-171', 'Porto', 'R. de Cimo
  ('04', 'Maçaroca - Mercearia Viva', 'Deborah Giovana', NULL, '4100-247', 'Porto', 'R. Sã
  ('05', 'Mercearia Africana Maya', 'Ada Maya', NULL, '4430-999', 'Vila Nova de Gaia', 'R.
  ('06', 'Halal Fresh meat And asian shop', 'Victor Boniface', NULL, '4000-160', 'Porto',
  ('07', 'ATL', 'Sandro Chioma', NULL, '4369-005', 'Porto', 'R. Chaves de Oliveira, 181',
```

```java
package com.restauranteezequiel.stock.restaurante.modelo.conexao;

//Bibliotecas a importar
import java.sql.Connection;
import java.sql.SQLException;

public interface Conexao {
    //Método que as classes que adotarem este tipo de interface têm de implementar
    public Connection obterConexao() throws SQLException;
}
```

```java
package com.restauranteezequiel.stock.restaurante.controller;

import com.restauranteezequiel.stock.restaurante.modelo.conexao.Conexao;
import com.restauranteezequiel.stock.restaurante.modelo.conexao.ConexaoMySql;
import java.sql.SQLException;
import java.sql.ResultSet;

public class Main {

    public static void main(String[] args) throws SQLException{
      //guarda numa string o comando sql para ver a tabela
      String sql = "select * from Categoria";
      //cria uma variavel para manipular a conexão
      Conexao conexao = new ConexaoMySql();
//resultado de executar a conexão através do "obterConexao()"
// e de preprar o comando sql definido em cima e por fim executar esse comando (query)
      ResultSet result = conexao.obterConexao().prepareStatement(sql).executeQuery();

      //enquanto encontrar descrições para imprimir, imprime.
      //Sai do ciclo quando não houve mais nada a imprimir
      while (result.next()) {
          System.out.println(result.getString("descricao"));
      }

    }
```

```java
package com.restauranteezequiel.stock.restaurante.modelo.conexao;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.DriverManager;

public class ConexaoMySql implements Conexao {

    //Instancia que guarda o "estado" da conexão
    private Connection connection;
    //Representa o nome de utilizador da base de dados, no caso no pc da Sara
    private final String Utilizador = "root";
    //Representa a senha que utilizo para aceder à base de dados
    private final String Senha = "Sara123_";
    //URL da base de dados, para que o java possa conectar
    private final String URL = "jdbc:mysql://localhost/restaurante_Ezekiel?useTimezone=true&s

    public Connection obterConexao() throws SQLException {
        if(connection == null) {
            connection = DriverManager.getConnection(URL, Utilizador, Senha);
        }
        return connection;
    }

    public void fecharConexao() throws SQLException {
        if(connection != null)
            connection.close();
    }

}
```

```java
    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getTipo_utilizador() {
        return tipo_utilizador;
    }

    public void setTipo_utilizador(String tipo_utilizador) {
        this.tipo_utilizador = tipo_utilizador;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }
```

```java
public LocalDateTime getUltimo_login() {
    return ultimo_login;
}

public void setUltimo_login(LocalDateTime ultimo_login) {
    this.ultimo_login = ultimo_login;
}

//Método hashCode
@Override
public int hashCode() {
    int hash = 3;
    hash = 89 * hash + this.id;
    hash = 89 * hash + Objects.hashCode(this.nome);
    hash = 89 * hash + Objects.hashCode(this.tipo_utilizador);
    hash = 89 * hash + Objects.hashCode(this.senha);
    hash = 89 * hash + (this.estado ? 1 : 0);
    hash = 89 * hash + Objects.hashCode(this.ultimo_login);
    return hash;
}

//Método Equals
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Utilizador other = (Utilizador) obj;
    if (this.id != other.id) {
```