UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

TRABALHO DIRIGIDO 4

VICTOR JONES MESQUITA DE SOUSA – 511418

Escolhido o tema B – Sistema e-commerce de uma livraria.

SUMÁRIO:

INTRODUÇÃO:

No Trabalho Dirigido 4 vamos abordar o sistema de e-commerce de uma livraria. Foi feito os requisitos funcionais, não funcionais e criamos diagramas UML. Exploramos padrões de projeto (criacionais, estruturais, comportamentais) selecionando um de cada categoria, descrevendo como se aplicam ao projeto. Essa análise é crucial para garantir um sistema eficiente e escalável, além de ajudar no desenvolvimento e na apresentação do projeto para o cliente.

ANÁLISE DE REQUISITOS:

01) Quais são os requisitos funcionais, não funcionais e de domínio da aplicação?
R:

Os Requisitos funcionais são: É um sistema de e-commerce então ele deve cadastrar clientes de diversos tipos (física, jurídica, estrangeira), logo após fazer a validação desses dados dos clientes e aí sim “liberar” o cliente a Realizar pedidos, nos pedidos temos as informações do pedido tais como a data realizada e o cálculo do frete, bem como pode pedir o valor total do pedido. Cada pedido possui seus itens, que por sua vez são associados a livros, e o sua forma de pagamento que pode ser de diversas formas (separados em à vista ou a prazo).

Os Requisitos não funcionais são: Para cadastrar os clientes no sistema ele pode usar uma plataforma própria ou de terceiros. Temos APIs que fazem as validações dos dados automaticamente. A liberação do sistema pode ser feita através de um token. O cliente pode adicionar vários pedidos no carrinho e depois o sistema fazer o calculo de valor total e do frete. A associação dos Itens com os livros pode ser feita no tabelas de relacionamento no banco de dados.

O domínio da aplicação: São requisitos derivados do domínio da aplicação e descrevem características do sistema e qualidades que refletem o domínio. Portanto temos Como características um sistema linear, onde o usuário começa realizando o seu cadastro, depois realiza os pedidos e por fim faz o pagamento. E como qualidade temos um sistema relativamente seguro já que fazemos as verificações dos dados do cliente e só liberamos o acesso com um token.

PLANEJAMENTO:

O planejamento do projeto é feito pelo gerente de projeto, ou em algumas empresas pelos tech leads, onde ele é responsável pela gerencia da equipe, acompanhamento e realiza todo um backlog de tarefas a ser feitas. Também é o responsável por definir o tempo que determinas tarefas serão feitas, se é preciso adiar, se a tarefa está 100%, tudo passa pela aprovação dele e do cliente. Com isso é possível determinar o tempo execução de cada projeto, essa é uma parte fundamental, pois geralmente o cliente quer o projeto rodando no menor tempo possível e com o tempo previsto pode ser feito o marketing e divulgação do produto de forma eficiente.

02) Elaborar um cronograma com estimativa de pontos de complexidade (1 ponto = 00:30 min de trabalho) para a execução das tarefas. Este cronograma consiste em 3 colunas: requisito funcional, estimativa de complexidade e quem da equipe iria desenvolver esta atividade.

R: Dentro de Pessoas que iria desenvolver teríamos :

Desenvolvedores: Eles seriam os responsáveis pela codificação dos algoritmos e pela implementação das funcionalidades do sistema.

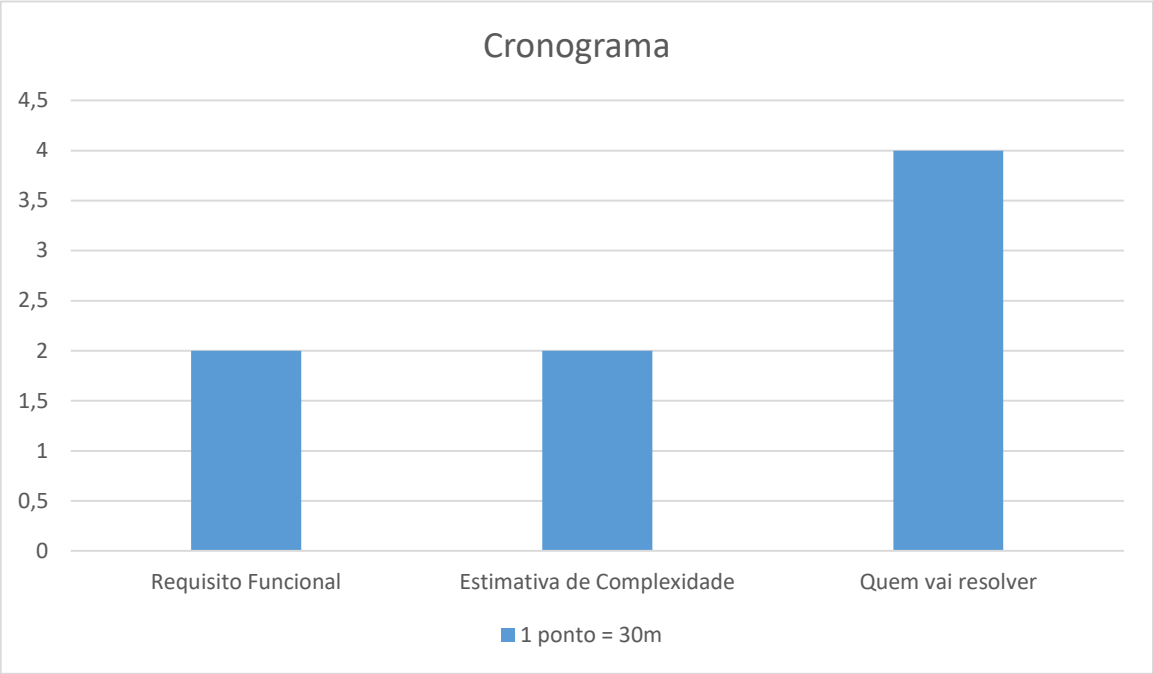
Analistas de sistemas: Eles seriam os responsáveis pela analise do projeto como um todo, então todos os requisitos, a analise da complexidade do projeto, qual é o melhor caminho para seguir, e o desenvolvimento de diagramas para auxiliar os desenvolvedores.

Analistas de qualidade: Eles seriam os responsáveis pela testagem/analise de qualidade do projeto, onde seu principal objetivo é levar o projeto ao limite para encontrar bugs e passar para a equipe de desenvolvedores resolver.

Lider de projetos : Ele seria o responsável pela gerência dessas pessoas, portanto ele iria marca reuniões e fazer o acompanhamento do projeto. Bem como sendo uma pessoa mais experiente, tiraria duvidas e resolveria os problemas mais complexos.

Acredito que no início do projeto cada um levaria umas 2 horas diárias para realizar os requisitos funcionais, umas 2 horas para estimar a complexidade e umas 4 horas de Trabalho no código. Somando 8 horas diárias.

Depois do início, poderíamos passar as horas de requisitos funcionais e complexidade, para os testes do código, para o review das funcionalidades, para deixar o sistema escalável e acessível para todos os tipos de usuários. Ou até mesmo passar mais hora para codificação e terminar o sistema em um prazo menor.

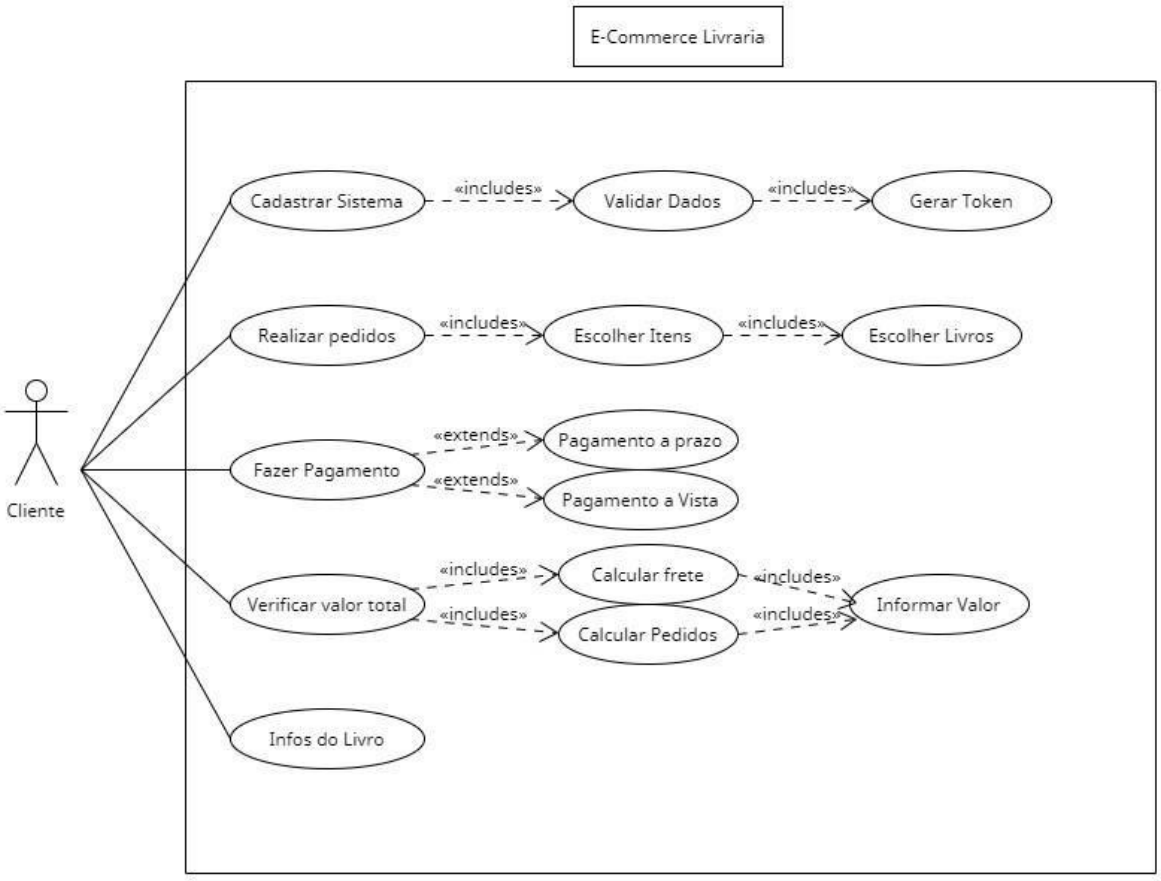


PROJETO DE SOFTWARE:

03) Quais são os casos de uso? Especifique-os em um Diagrama de Casos de Uso.

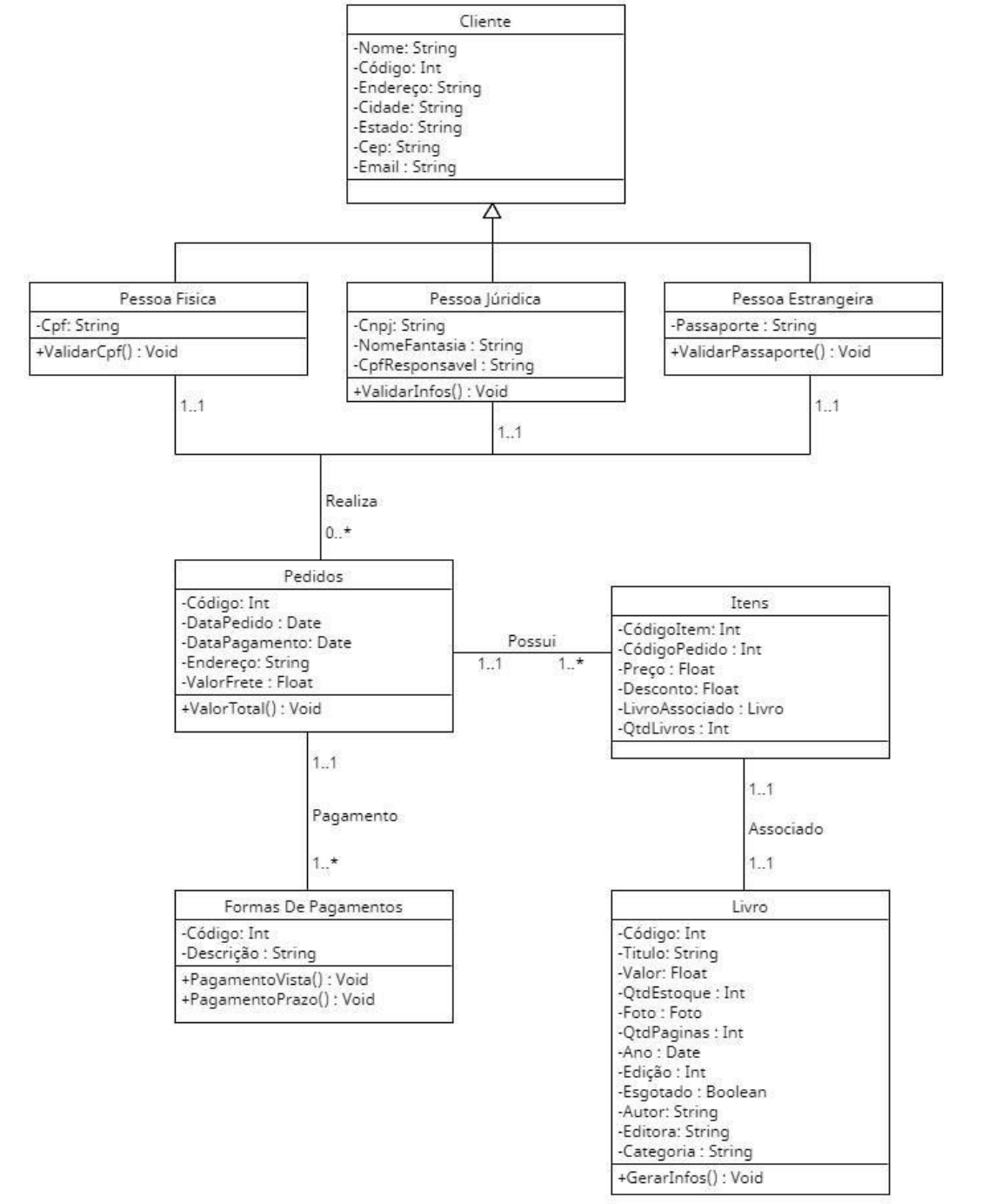
R:

Aqui temos o ator cliente que interage com o sistema de e-commerce, a partir disso temos alguns casos de uso tais como : Se cadastrar no sistema, realizar pedido, fazer pagamento, verificar valor total e informações do livro.



04) Quais são as entidades? Especifique-as em um Diagrama de Classes R:

No diagrama de classe vamos ter algumas entidades, começando pelo cliente, temos os atributos comuns a todos os tipos de cliente, e depois alguns específicos de clientes do tipo pessoa física, pessoa jurídica e pessoa estrangeira, portanto esses 3 herdam os atributos de cliente padrão. Vamos ter a entidade pedido, onde por sua vez interage com formas de pagamentos (para realizar o pagamento), ou Itens (Para escolher o item do pedido).
E por fim itens interage com Livro, que serve para escolher um tipo único e especifico de livro.



PADRÕES DE PROJETO:

05) Escolha 1 padrão de projeto criacional. Descreva detalhadamente como serão suas aplicações no seu projeto, especificando classes e funcionalidades envolvidas da sua aplicação.

Os Padrões de criação se preocupam com o processo de criação de objetos. Portanto, temos alguns padrões predefinidos pela GOF.

R: Singleton:
O Singleton é um padrão especifico para projetos que querem gerar/usar uma única instância de uma classe no projeto, seja por questão de memória ou por questões de necessidade. Portanto devido a sua importância temos que garantir que tenha somente uma instância da classe e fornecer um ponto global de acesso para ela.
Podemos utilizar isso no sistema de pedidos do e-commerce onde temos apenas 1 pedido por vez do cliente. Portanto quando o cliente termina de fazer o login criamos apenas uma instância da **classe pedido**, onde essa instância vai ser a responsável por administrar todos os **itens**(Livros) pedido pelo cliente, fazendo com que todos os itens estejam na mesma **instância** podemos calcular o valor do pedido inteiro, e a partir disso fazer o pagamento do mesmo. Portanto essa é uma forma de deixar o sistema mais organizado e confiável.

06) Escolha 1 padrão de projeto estruturais. Descreva detalhadamente como serão suas aplicações no seu projeto, especificando classes e funcionalidades envolvidas da sua aplicação.

Os Padrões estruturais lidam com a composição de classes ou de objetos. Portanto, temos alguns padrões predefinidos pela GOF.

R: Proxy:
O proxy fornece um objeto representante, ou um **marcador** de outro objeto, para **controlar** o acesso ao mesmo. Teoricamente o proxy pode ser conectado a qualquer objeto, ou seja, normalmente quando existe uma instância grande/complexa pode-se criar vários proxies, todos contendo uma única referência.

Podemos utilizar isso no sistema de estoque dos livros, onde teremos uma classe estoque e vários proxies com uma única referência para ela, cada proxy pode ser reservado a um livro, isso nos permitiria a utilização dos Proxy para fazer a busca no sistema se temos o livro em estoque ou não, já que ele é um **marcador** e podemos controlar o acesso ao mesmo.

07) Escolha 1 padrão de projeto comportamentais. Descreva detalhadamente como serão suas aplicações no seu projeto, especificando classes e funcionalidades envolvidas da sua aplicação.

Os Padrões comportamentais caracterizam as maneiras pelas quais classes ou objetos interagem e distribuem responsabilidades. Portanto, temos alguns padrões predefinidos pela GOF.

R: State:
O State permite que um objeto altere seu comportamento quando seu estado **interno muda**. Isso nos permite fazer varias **condições** que caso um determinado estado acontecer resulte em varias

mudanças nas classes do projeto. Essas mudanças podem ser leves ou bruscas, fazendo com que pareça que o objeto mudou de classe.

Podemos utilizar isso no Sistema de geração de token, onde o estado interno da **classe GerarToken** será **mudado** caso o cliente tenha seus dados validados ou não, caso for validado, teremos uma mudança no estado da classe, pois o cliente vai receber autorizações extras. Caso der invalido os dados do cliente, ele vai pedir para checar os dados e não vai dar autorização. Portanto podemos fazer essa "**condição**" com o state.

CONSIDERAÇÕES FINAIS:

Portanto temos a análise e documentação do projeto e-commerce de uma livraria, onde fizemos toda a parte de requisitos funcionais, não funcionais. Fizemos também toda a parte de diagramação do projeto com alguns diagramas da UML tais como diagrama de classe e casos de uso. Que serão de utilidade para o desenvolvimento do projeto. Por fim fizemos todo um estudo/descrição de como funcionam os padrões de projeto do GOF e como podemos utiliza-los no nosso projeto.

Em um projeto real, essa documentação é tão importante quanto a parte de codificação, já é comprovado que temos economia de tempo e de recursos, quando fazemos primeiro a documentação, entendemos todas as necessidades do projeto, e só depois passamos para a fase de desenvolvimento.