Wings

E-Commerce Wings
Documento de Arquitetura de Software

Versão <7.0>

Grupo 2:
André Lopes
Brendan Buckley
Lucas Guarnelli Scherpel
Nicolas Atmatzides
Vitor Rocha

UFF - Universidade Federal Fluminense
IC - Instituto de Computação

Disciplina: Projeto de Software
Professor: Paulo Pires

Instituto de Computação

Histórico da Revisão

Data	Versão	Descrição	Autor
15/11/2021	1.0	Primeira versão	Lucas Scherpel
15/12/2021	2.0	Segunda versão	Brendan Buckley e Vitor Rocha
21/01/2022	3.0	Todos os tópicos completos	Brendan Buckley
23/01/2022	4.0	Atualização de diagramas	Vitor Rocha
26/01/2022	5.0	Atualização de diagramas	Vitor Rocha
31/01/2022	6.0	Atualização de diagramas	André Lopes
01/02/2022	7.0	Atualização de diagramas	Nicolas Atmatzides

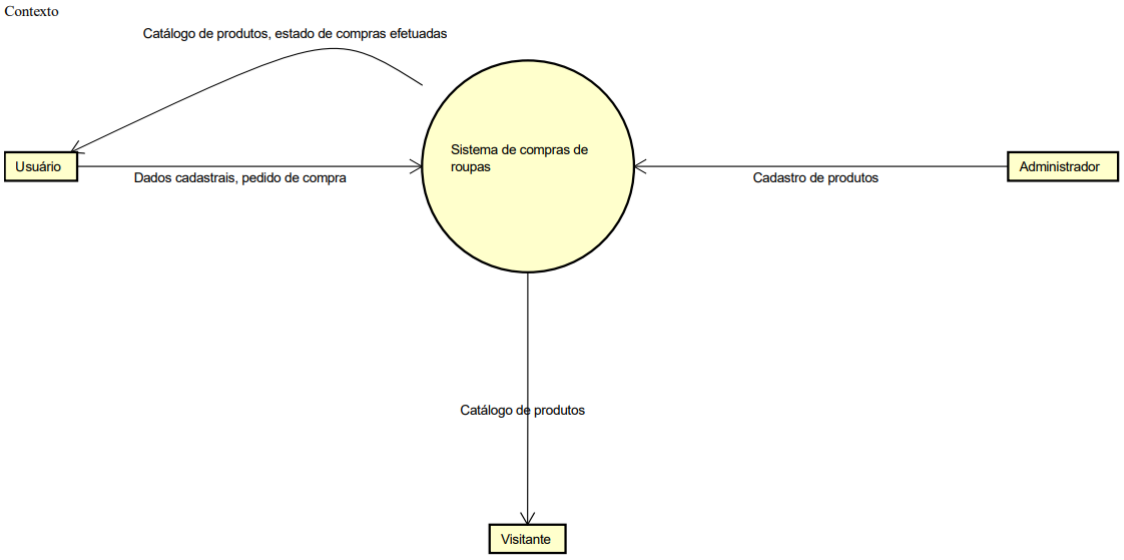
Índice Analítico

Documento de Arquitetura de Software

1. Introdução

Este documento explica de forma detalhada as decisões arquiteturais na construção de um sistema de e-commerce de roupas. O foco principal do projeto é o aprendizado e a aplicação dos conceitos de projeto de software, por isso não é esperado que todas as funcionalidades e requisitos estejam implementados no produto final, sendo este, um sistema unissex de venda de roupas chamado Wings.

Diagrama de Contexto do Sistema



1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

1.2 Escopo

O escopo deste documento é oferecer o maior nível de detalhamento possível em relação a estrutura do projeto tanto em alto nível como a divisão de camadas e o contexto, quanto em baixo nível como os métodos mais importantes das classes e quais as dependências entre elas.

1.3 Referências

O material didático usado como referência para a construção deste documento foi principalmente o livro: VALENTE, Marco. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.

1.4 Visão Geral

Em geral este documento foi focado para os desenvolvedores e usa em sua maioria linguagem específica de software. Porém também foram adicionados diagramas para facilitar o entendimento em geral dos conceitos abordados.

2. Metas e Restrições da Arquitetura

O sistema será dirigido por funcionalidades simples e de fácil entendimento e implementação, tentando seguir os padrões já existentes no mercado para e-commerce. O sistema utiliza tecnologias atuais como o React e precisa fazer integração com outras tecnologias que também são atuais e funcionam bem em conjunto, então não existe necessidade de integração com sistemas legados. O desempenho que se espera é de atualização e respostas em tempo real, e as tecnologias que estão sendo usadas tem suporte para esse tipo de resposta. A construção do sistema deve facilitar futuras mudanças ou upgrades mesmo que elas não estejam no escopo atual.

As metas do sistema são garantir que uma possível mudança de banco de dados seja simples de implementar, diminuindo as dependências e classes que seriam afetados. Além disso, é importante que seja considerado o uso em dispositivos móveis, e possivelmente uma conexão de internet instável, por isso os requisitos de hardware devem ser baixos e as operações e transações devem ficar num estado seguro e salvar o progresso em situações de queda de conexão.

3. Suposições e Dependências

O projeto será realizado de novembro de 2021 até 2 de fevereiro de 2022 quando será entregue, e conta com uma equipe de 6 estudantes universitários de Sistemas de informação. Devido ao curto período e pequena alocação de tempo, o escopo foi reduzido principalmente na implementação do código. As tecnologias usadas são novas para a maioria dos integrantes, que estão aprendendo enquanto produzem.

4. Requisitos Arquiteturalmente Significativos

4.1 MINIMUNDO

A **Wings** é uma loja unissex virtual que vende diversos tipo de roupa. O cliente se cadastra no site usando e-mail e uma senha à sua escolha, e assim fará o login no sistema.
O usuário vai navegar no sistema, e adicionar no carrinho de compras tudo que ele quiser comprar. O site separa os produtos por categoria e tem uma caixa de busca para os produtos. Após o cliente terminar de escolher seus produtos, ele seleciona o carrinho no canto da tela e clica na opção de finalizar escolha de produtos, assim o site deve direcioná-lo para a página de pagamento.

O site tem uma parte para cadastro de administradores, que por sua vez, farão o cadastro dos produtos no site, vão habilitar/desabilitar os anúncios dependendo de sua disponibilidade ou não no estoque, vão cadastrar categorias e cadastrar outros usuários administradores no sistema. O site tem uma parte para configurações de login para os usuários, uma parte para alteração de dados cadastrais, uma parte para acompanhamento de compras já realizadas, e alteração de dados em relação ao login, como troca de senha e mudança de e-mail.

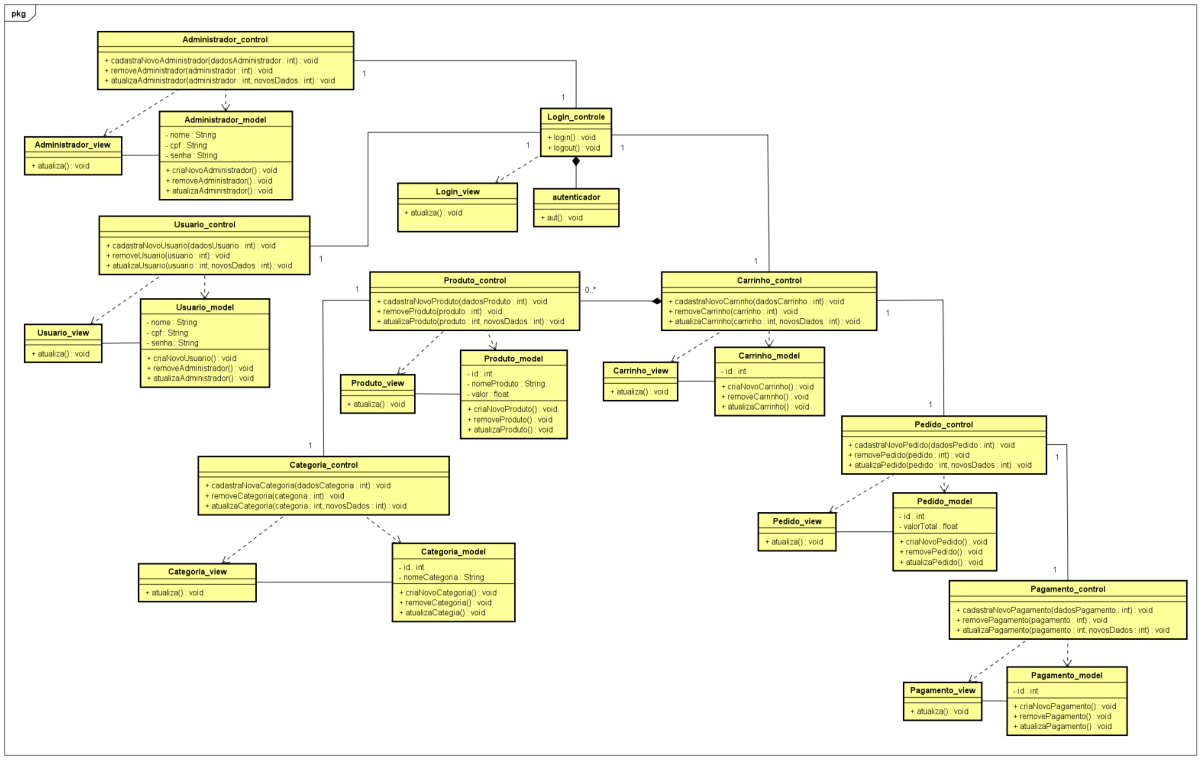
4.2 Requisitos funcionais e não-funcionais

RF1 - Um usuário deve conseguir comprar um produto.
RF2 - Um usuário deve ser capaz de encontrar qualquer produto por meio de pesquisa ou navegando por categorias.
RF3 - Um usuário deve ter um carrinho de compras para seus pedidos.
RF4 - Um usuário deve ser capaz de se cadastrar e editar suas informações de cadastro.
RF5 - Um usuário cadastrado deve ser capaz de acompanhar a situação dos seus pedidos.
RF6 - Um administrador deve poder criar, listar, editar e excluir (CRUD) anúncios de roupas.
RF7 - Um administrador deve poder criar outros administradores e gerenciar suas permissões.
RF8 - Um administrador deve poder acessar dados sobre o estoque dos produtos.

RNF1 - O site deve ser responsivo na interface para telas dos tamanhos Extra large ≥1200px e Small ≥576px (usabilidade)
RNF2 - A integridade dos dados de usuários e administradores deve ser garantida. (segurança)
RNF3 - A Lei Geral de Proteção de Dados Pessoais deve ser considerada e respeitada no desenvolvimento do projeto (desenvolvimento/segurança)
RNF4 - O código do projeto deve ser colocado no github.com (desenvolvimento)
RNF5 - O sistema deve ter estar disponível 95% do tempo 24 horas do dia (disponibilidade)
RNF6 - Uma consulta aos produtos deve ocorrer em menos de 3 segundos 95% do tempo (eficiência)

5. Decisões, Restrições e Justificativas

Diagrama de Classes:



A primeira decisão tomada foi a adoção do modelo MVC, já que ele é bastante comum em aplicações de e-commerce e faz bastante sentido neste contexto de desenvolvimento que lida tanto com front-end quanto conexão com banco de dados. Esse modelo norteou várias outras decisões arquiteturais como os padrões de projeto Controller e Creator, além de implementar o conceito de camadas e melhorar a organização e leitura do código.

Sobre as restrições do projeto, é relevante esclarecer que principalmente por estar sendo utilizada tecnologia atual, é importante ter em mente atualizações futuras e possíveis mudanças e upgrades de linguagem, ou de banco de dados. Além disso, é importante facilitar o possível crescimento do projeto, caso ele faça muito sucesso, por exemplo.

5.1 Faça / Não Faça

- O frontend será feito em React, pois alguns integrantes do grupo já tem experiência com a linguagem e ela permite um aplicativo de página única.
- O código deve obedecer os paradigmas da orientação a objetos, pois facilita a manutenção e a divisão das tarefas pelo time.
- A escrita do código deve respeitar as boas práticas de programação, e utilizar o snake_case. Dessa forma o entendimento do código pelos outros integrantes do grupo fica facilitado.
- Não dar “commit” diretamente caso esteja num código de outro integrante do grupo, invés disso criar uma nova “branch” com a versão nova e discutir as mudanças.

6. Mecanismos Arquiteturais

Persistência

A persistência dos dados é garantida pelo banco de dados em nuvem firestore.

Autenticação

A autenticação de usuários é feita através do autenticador Google, pois ele é seguro, de fácil implementação e já possui integração com o firestore.

Front-End

O front-end ficará na camada View e será responsável por interagir com o usuário do site e mostrar informações do banco de dados trazidas pela camada Controller. O front-end será feito em React.

Build

A montagem do projeto será feita pela IDE Visual Studio Code, pois já é conhecida pelos integrantes, além de ser de fácil utilização.

Deploy

O plano para deployment do projeto é hospedar em um host gratuito como o github pages ou heroku.

6.1 Padrões de projeto GOF

Fachada

O padrão de fachada é usado para facilitar o uso de classes complexas. Ele é usado na classe conexão, que simplifica as operações necessárias para conectar com o DB e retorna objetos mais simples.

Observador

O padrão de observador é usado para atualizar objetos que mudam de estado. Na nossa aplicação esse padrão é utilizado indiretamente por meio de hooks do react, pois eles observam os objetos dados como parâmetro e atualizam a página sempre que o objeto muda. Dessa forma a meta de atualização em tempo real é suprida.

6.2 Padrões de projeto GRASP

Controlador

O padrão controlador é usado para separar a camada de interação com usuário da camada de sistema, e é normalmente utilizada para implementar os casos de uso. Como o nosso sistema já utiliza o padrão MVC o controlador é usado em todos os casos de uso, e um mesmo controlador pode resolver vários casos de uso, como a classe produto_controle que faz o CRUD e todas as operações que envolvem produtos.

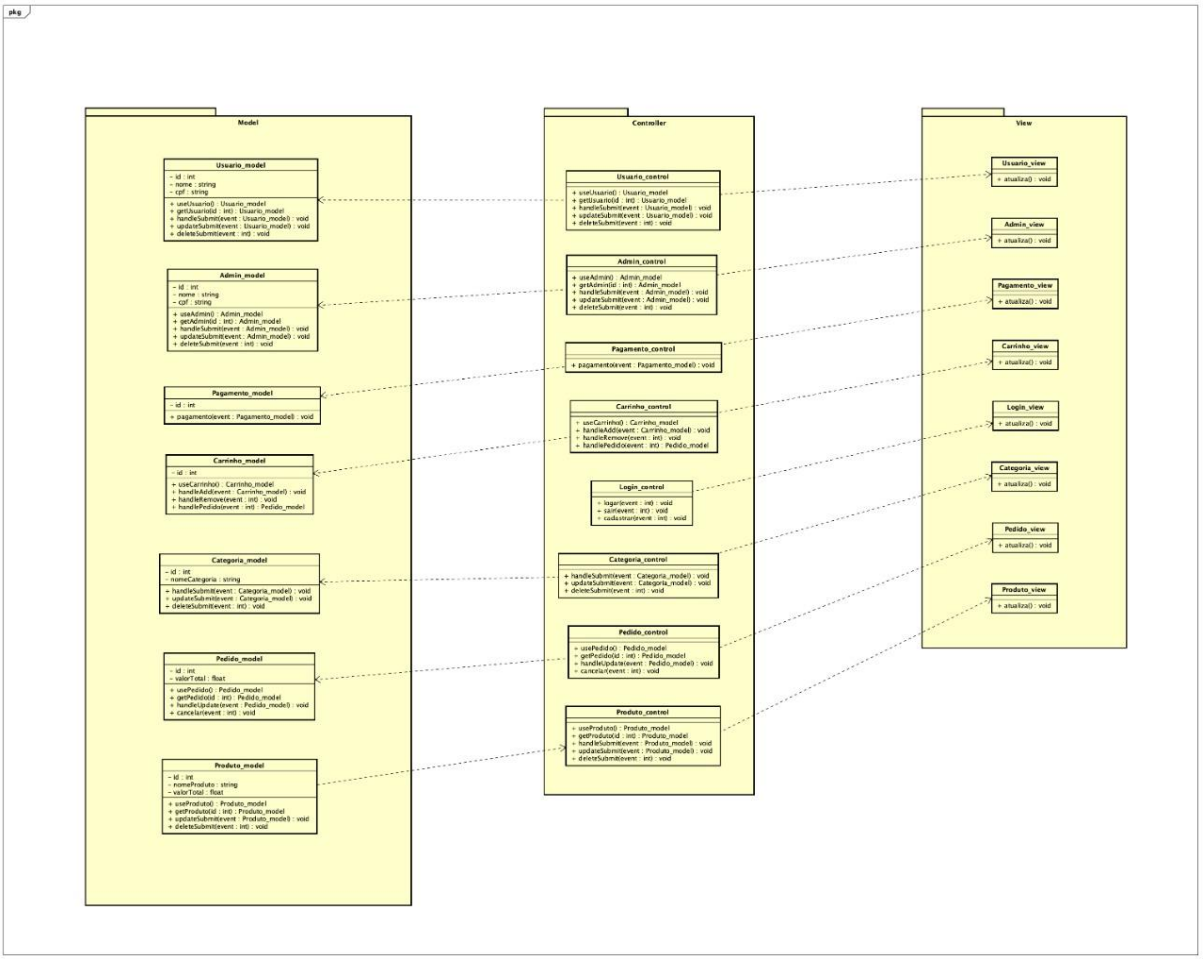
Indireção

O padrão de indireção é usado para reduzir o acoplamento entre duas classes e garantir maior reuso por meio de uma classe intermediária que faz a ponte entre eles. Novamente, esse padrão já é usado no MVC, pois as classes controle fazem justamente essa ponte entre os modelos e as views.

Alta Coesão

Esse padrão representa um dos princípios de OO, a ideia de que uma classe deve representar o mundo real e ser focada e fácil de compreender. Buscamos seguir esse padrão em todo o código, principalmente a questão do foco, por isso existem um número grande de classes que são em geral bem pequenas.

7. Camadas da Arquitetura



Arquitetura em camadas:
Esse padrão arquitetural se trata de hierarquia e organização. Com esse padrão uma camada de hierarquia mais alta pode utilizar serviços de uma com hierarquia mais baixa.
Nesse projeto o padrão arquitetural em camadas será usado para instanciar classes e objetos nos arquivos .JS dentro do projeto feito na linguagem React.

Cliente-Servidor:
Nesse padrão arquitetural o cliente requisita um serviço ao servidor e aguarda resposta.
No projeto Wings, esse padrão será utilizado nas requisições ao banco de dados, utilizando como comunicação o padrão **REST** e o servidor **Firebase**.

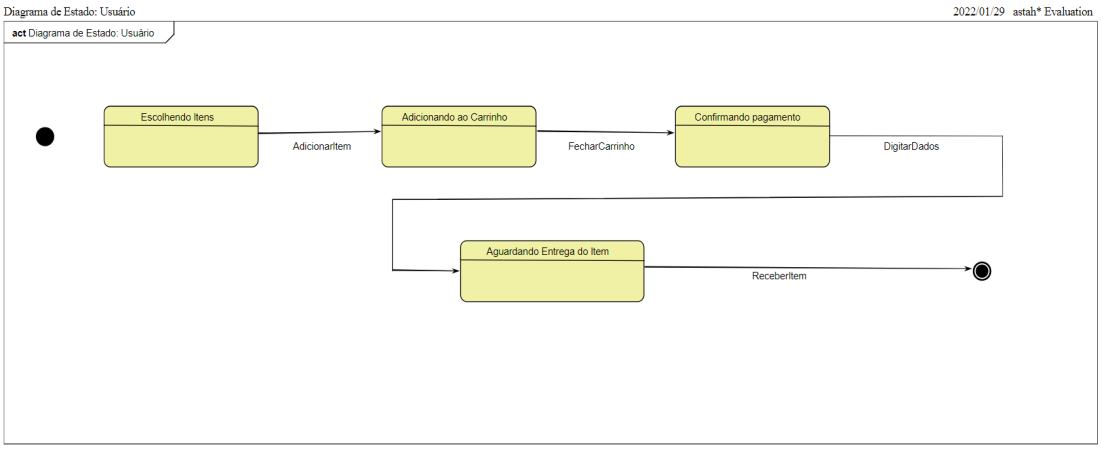
Single Page Application (SPA):
A plataforma web será feita em **React**, um framework JavaScript para criação de interfaces de usuário. Usaremos os métodos de roteamento do React como o router e redux, para implementar o padrão de 'aplicação de página única'. Existe apenas o arquivo index para representar todas as páginas do site, estas serão implementadas em scripts separados, e o react atualizará a página index por meio de rotas com o conteúdo de cada script.

Model - View - Controller (MVC):
O projeto será dividido em 3 grandes pastas: Model, View e Controller. Na pasta View estará o index, o CSS e todos os scripts referentes às páginas do site; essa camada é responsável pela interface com o usuário, ou o front-end do projeto. Já na pasta Model, estarão principalmente os códigos javascript para acesso e manipulação de banco de dados Firebase, além da comunicação com o nosso gateway de pagamento Stripe. A pasta Controller vai conter arquivos de comunicação entre as camadas, funcionando como uma ponte que permite a interação entre as camadas que serão independentes.

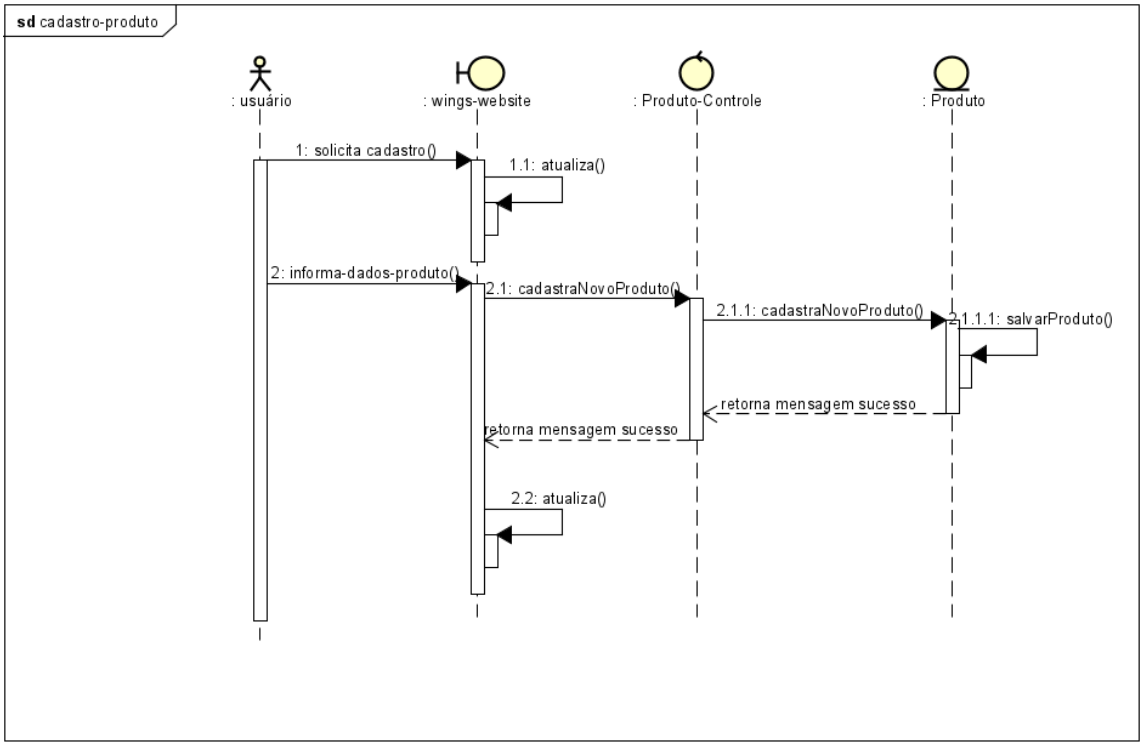
8. Visões da Arquitetura

8.1 Lógica

Diagrama de estado do objeto usuário:



Diagramas de Sequência:



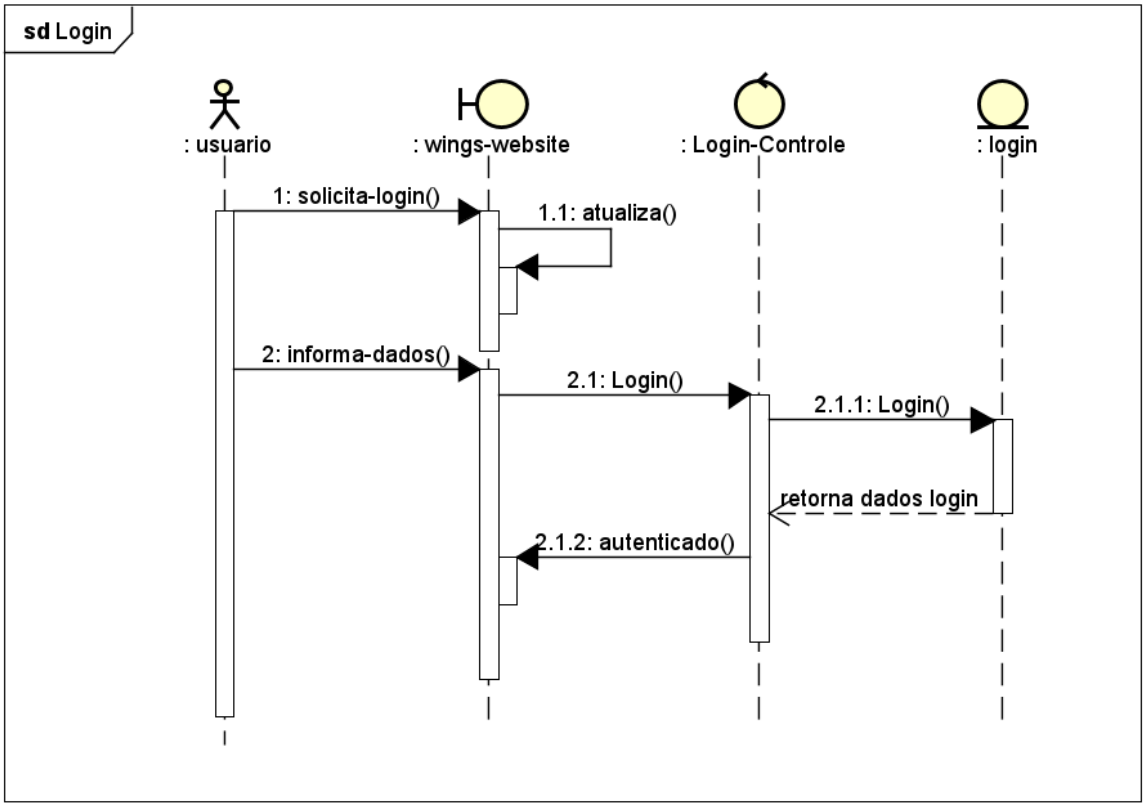
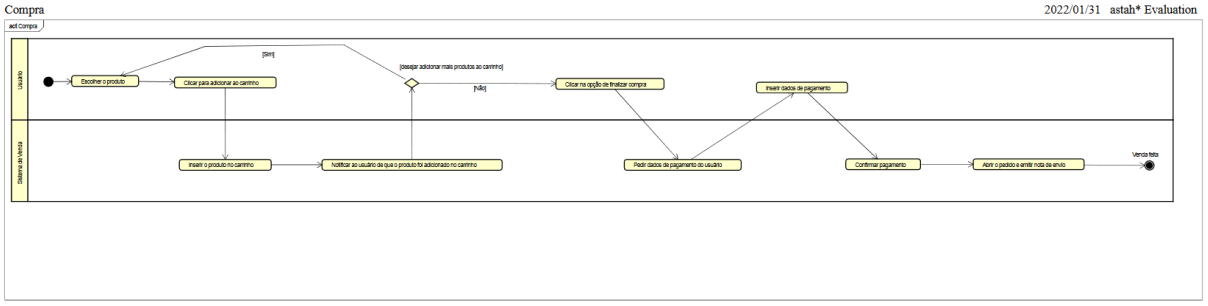


Diagrama de atividades da atividade compra:



Link para visualização do arquivo:

https://drive.google.com/file/d/1avFyzcquiORytv_h9nxKADL7WH4J1wub/view?usp=sharing

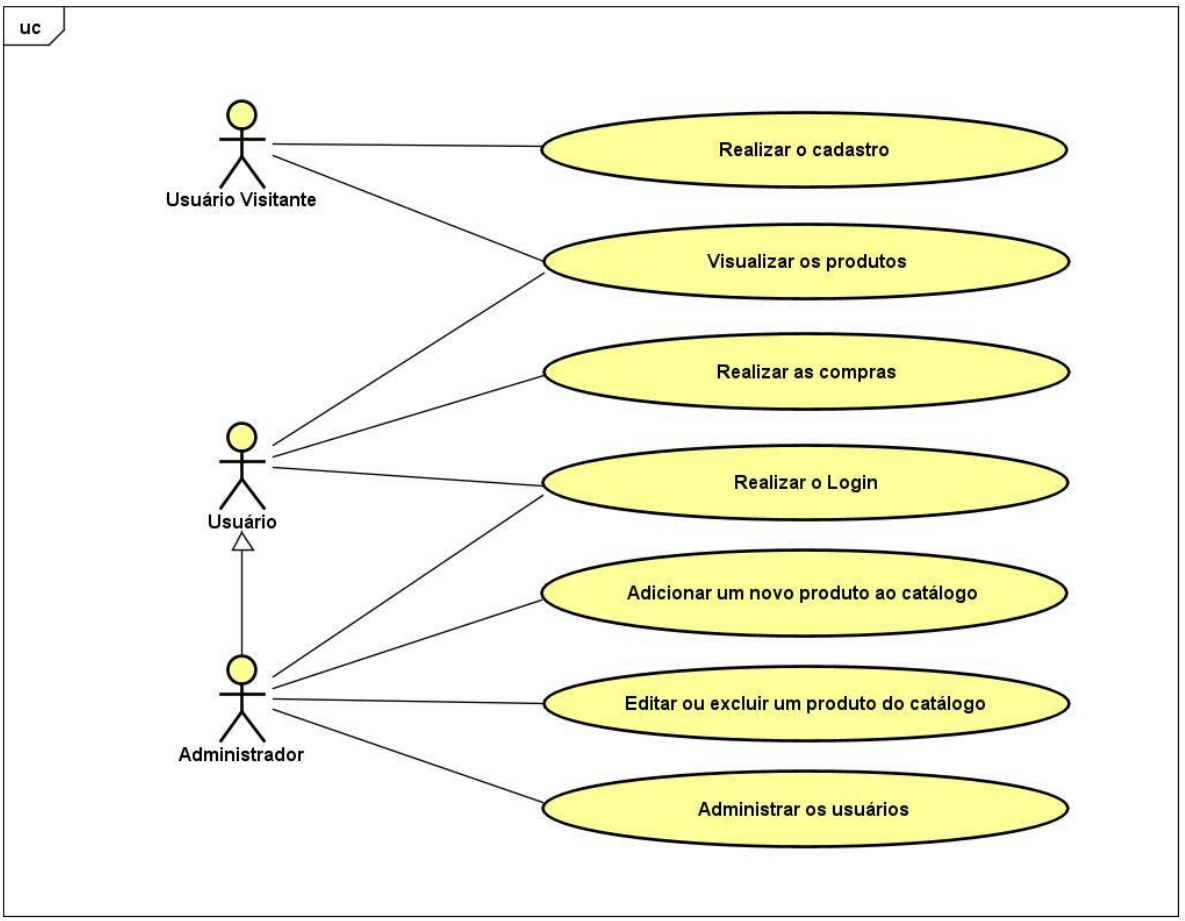
O sistema está estruturado em 4 principais pacotes, Model, View, Controller, e Core. A View é responsável pela interação com o usuário e se comunica apenas com o Controller, que é responsável por fazer a "ponte" entre os pacotes. O Model é responsável pela comunicação com o banco de dados e repassa esses dados para o controller. O Core é um pacote auxiliar com classes e funções que dão suporte aos outros pacotes, como Conexão, Roteamento, Configurações ou Templates.

Dentro da View também existem conexões importantes, pois uma única página geralmente possui vários componentes, como navbar, footer, carroussel, além de diversos tipos de conteúdo como tabelas, texto e imagem. E para garantir mais organização, modularidade e reuso de componentes, estes são divididos em classes diferentes, e a View é responsável por unir todos os componentes necessários e chamar o Controller caso precise de dados.

8.2 Casos de Uso

Ator	Papel
Usuário	Usuário cadastrado no sistema, pode adicionar itens no seu carrinho, comprar roupas e acompanhar seus pedidos.
Administrador	Adicionar ou remover produtos do site. Tem controle sobre o estoque e pedidos dos clientes.
Visitante	Usuário não cadastrado no site, pode visualizar todo o catálogo, porém não pode efetuar a compra sem antes se cadastrar.
Stripe	A Stripe é um sistema que auxilia o nosso, e atua como gateway de pagamento para efetuar as compras.

Diagrama de Caso de Uso



Descrição Detalhada dos Casos de Uso

Nome do caso de uso	UC1: Realizar Cadastro
Ator(es)	Visitante
Descrição	O visitante entra no site, pesquisa produtos, realiza buscas, checa preços, e por fim decide comprar. Então o visitante é direcionado para realizar um cadastro no site e se tornar um usuário, para assim conseguir finalizar as compras.
Gatilho	Clicar em “cadastrar”, ou em “comprar” sem estar logado.
Pré-condições	Nenhuma
Pós-condições	O visitante agora é um usuário cadastrado no sistema.
Fluxo principal	1- O visitante preenche o campo de email. 2- O visitante preenche o campo de senha. 3- O visitante preenche o campo de confirmar senha. 4- O visitante clica em “Enviar” e finaliza o formulário de cadastro. 5- O sistema cadastra o novo usuário e loga o visitante no site.
Fluxo alternativo	1.1 Email inválido. 1.1.1 O sistema informa o visitante para preencher um email válido.

Nome do caso de uso	UC2: Visualizar Produtos
Ator(es)	Visitante, Usuário
Descrição	O visitante ou usuário acessa o site e confere na tela principal as principais categorias. O usuário clica em uma categoria e visualiza todos os produtos daquela categoria. O usuário acessa a página “produtos” na barra de navegação para ver todas as categorias e todos os produtos. O usuário seleciona um produto e confere as informações e preço dele.
Gatilho	Acessar o site
Pré-condições	nenhuma
Pós-condições	O visitante ou usuário conhece o catálogo da loja e está na página do produto que lhe interessa.
Fluxo principal	1- O visitante/usuário seleciona e acessa uma categoria recomendada. 2- O visitante seleciona um produto. 3- O visitante confere fotos, descrições e preço do produto.
Fluxo alternativo	1.1 Sem interesse pelas categorias recomendadas 1.1.1 O visitante/usuário clica em “produtos” na barra de navegação e confere todas as categorias e produtos disponíveis.

Nome do caso	UC3: Realizar as Compras
Ator(es)	Usuário
Descrição	O usuário cadastrado, após selecionado o produto desejado, ou o carrinho de compras, é direcionado para a página de compra, onde seus dados são confirmados, ou atualizados caso necessário. É informada a forma de envio e de pagamento e ele é direcionado para a Stripe.
Gatilho	Clicar em comprar ou no carrinho de compras
Pré-condições	Usuário cadastrado, produtos desejados selecionados ou carrinho pronto.
Pós-condições	Pedido criado, e em estado: aguardando pagamento.
Fluxo principal	1- O usuário seleciona do catálogo a roupa desejada. 2- O usuário clica em comprar e adiciona o item ao carrinho. 3- O usuário clica no carrinho e finaliza o pedido.. 4- O usuário verifica seus dados e informa endereço de envio. 5- O usuário seleciona a forma de pagamento. 6- O sistema redireciona o usuário para o Stripe. 7- O sistema informa ao usuário que o pedido foi feito com sucesso e está aguardando pagamento.
Fluxo alternativo	

Nome do caso de uso	UC4: Realizar Login
Ator(es)	Usuário, Administrador
Descrição	O usuário ou administrador clica em entrar e é direcionado para o formulário de login. O usuário preenche seu email e senha e clica em enviar. O usuário agora está logado.
Gatilho	O usuário ou administrador clica em “entrar” na barra de navegação.
Pré-condições	Estar cadastrado no sistema.
Pós-condições	O usuário ou administrador está logado no site e pode acessar áreas restritas, realizar compras etc.
Fluxo principal	1- O usuário preenche o campo de email. 2- O usuário preenche o campo de senha. 3- O usuário clica em “Enviar” e finaliza o formulário de login. 4- O sistema loga o usuário no site.
Fluxo alternativo	2.1 senha incorreta. 2.1.1 O sistema informa ao usuário e envia um email com os passos para a criação de uma nova senha.

Nome do caso	UC5: Adicionar um novo produto ao catálogo
Ator(es)	Administrador
Descrição	O administrador acessa a página de novo produto, procura a categoria mais adequada para a roupa nova ou cria uma nova categoria. No template do anúncio ele informa os dados necessários como valor, estoque, tamanhos, cores. E por fim faz upload de fotos da roupa e clica em finalizar.
Gatilho	Clicar em ‘novo produto’.
Pré-condições	Administrador cadastrado no sistema e categoria existente para a roupa nova.
Pós-condições	Novo produto adicionado ao catálogo.
Fluxo principal	1- O administrador seleciona a categoria adequada. 2- O administrador preenche dados sobre o produto. 3- O administrador faz upload das fotos.
Fluxo alternativo	1.1 Não existe categoria adequada. 1.1.1 O administrador cria uma nova categoria.

Nome do caso de uso	UC6: Editar ou excluir um produto do catálogo
Ator(es)	Administrador
Descrição	O administrador acessa a página de editar/excluir, e procura pela lista de todos os produtos cadastrados. O administrador usa a busca para encontrar rapidamente o produto desejado. O administrador seleciona o botão excluir e confirma. O administrador seleciona outro produto e agora usa o botão editar. O sistema gera um formulário de edição para o administrador confirmar e alterar os dados. O administrador confirma as edições.
Gatilho	Clicar em “editar/excluir”.
Pré-condições	Produtos existentes no catálogo.
Pós-condições	Produto atualizado ou excluído.
Fluxo principal	1- O administrador navega pela lista de produtos 2- O administrador usa a busca pelo nome do produto desejado e o seleciona 3- O administrador clica em editar e preenche o formulário com as novas informações; ou clica em excluir. 4- O sistema atualiza o catálogo.
Fluxo alternativo	2.1 nome inexistente no banco de dados 2.1.1 O sistema avisa ao administrador que o produto não existe

Nome do caso de uso	UC7: Administrar usuários
Ator(es)	Administrador
Descrição	O administrador acessa a área restrita com a lista de todos os usuários cadastrados e seus pedidos. O administrador pesquisa por um usuário informando seu nome. O administrador confere todos os pedidos do usuário selecionado e seus estados.
Gatilho	Clicar em “usuários”.
Pré-condições	Administrador logado e usuários cadastrados.
Pós-condições	Administrador tem informação sobre usuários e pedidos.
Fluxo principal	1- O administrador navega pela lista de usuários 2- O administrador pesquisa pelo nome de um usuário específico e o seleciona. 3- O administrador confere todos os pedidos referentes ao usuário e seus estados. 4- O administrador seleciona um pedido e confere seus detalhes.
Fluxo alternativo	3.1 Nenhum pedido existente 3.1.1 O sistema informa ao administrador que este usuário ainda não tem pedidos vinculados.

9. Qualidade

Portanto, concluímos que o projeto arquitetado da forma exposta neste documento fornece maior manutenibilidade principalmente em relação ao modo de persistência de dados e mudanças de banco, além de facilitar o crescimento e expansão no futuro usando como base os padrões arquiteturais feitos ou facilmente trocando uma parte modular do código por outra que supra as novas necessidades. Outro aspecto importante que ressalta a qualidade do código estruturado é a facilidade de integrar novos programadores no time, pois o código fica mais legível e em caso de dúvida se pode recorrer a este próprio documento.

Enfim, a qualidade de um software é algo difícil de ser julgada, principalmente quando em dois programas distintos todos os requisitos e funcionalidades são supridos, mas tendo em vista todos os benefícios citados neste documento, é possível enxergar com mais clareza a maior qualidade do software final depois de estruturado e implementado com todas essas diretrizes.