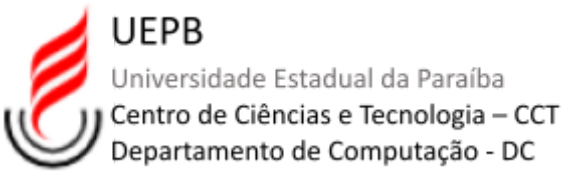


Disciplina: Métodos Avançados de Programação

Professora: Sabrina

Grupo:

Anna Caroline, Mat: 202080226

Daniel Ribeiro, Mat: 212080431

Henrique José, Mat: 202080080

João Pedro, Mat: 192080199

João Vitor, Mat: 211080080

Luana Nóbrega, Mat: 201080478

Relatório: Marketplace

Campina Grande

2023

Relatório Marketplace

MAP - Métodos Avançados de Programação

Grupo: Anna Caroline, Daniel Ribeiro, Henrique José, João Vitor, João Pedro e Luana Nóbrega

---

Índice

Relatório Marketplace

MAP - Métodos Avançados de Programação

Grupo: Anna Caroline, Daniel Ribeiro, Henrique José, João Vitor, João Pedro e Luana Nóbrega

---

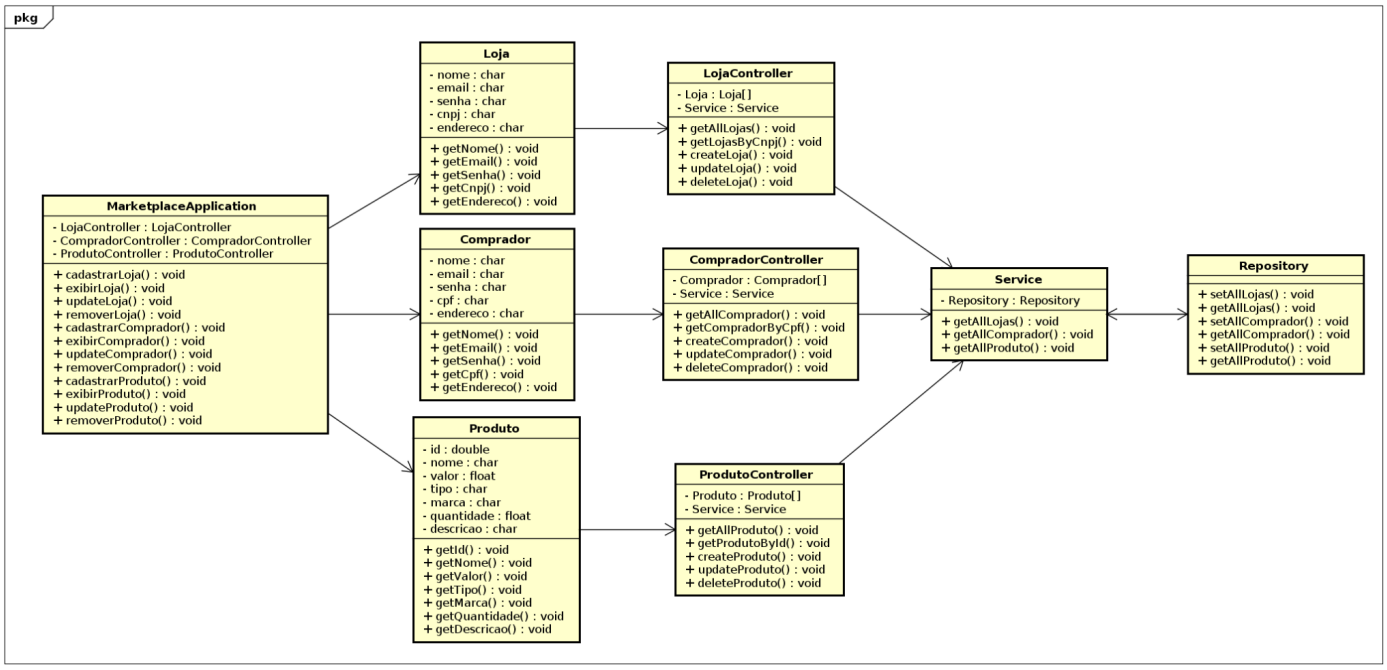
Relatório de entregas

Esse relatório contém informações de acordo com o que foi desenvolvido em cada release, juntamente com observações e alterações, conforme necessário.

1. Release 01 (04/06/23)

Repositório: https://github.com/jv-guimaraes/marketplace/releases/tag/v1.0.0
Vídeo de apresentação: https://drive.google.com/file/d/1-l_X76Rr5ni1xY_GC5L4oSQkHi7w5QAr/view?usp=sharing

1.1. Diagrama de classe



1.2. Lista de padrões utilizados

Padrões:

- Singleton
- Fachada
- Visitor
- Repository

Relatório Marketplace

MAP - Métodos Avançados de Programação

Grupo: Anna Caroline, Daniel Ribeiro, Henrique José, João Vitor, João Pedro e Luana Nóbrega

---

O padrão de projeto singleton é aplicado na classe MarketplaceApplication do sistema de marketplace. Essa classe permite que tenha uma acesso global para as outras instância, sendo esta o ponto de acesso conhecido entre elas. O MarketplaceApplication é uma abordagem que foi aplicada no sistema de marketplace para separar a lógica de acesso aos dados das operações de CRUD (Create, Read, Update, Delete).

Os métodos da classe MarketplaceApplication são responsáveis por chamar os métodos correspondentes nas classes com padrões de fachada, no projeto foram identificados com o sufixo “controller”, sendo estes: LojaController, CompradorController e ProdutoController. Essas fachadas são responsáveis por lidar com a manipulação dos objetos Loja, Comprador e Produto, respectivamente, incluindo a validação, busca, atualização e exclusão dos dados.

Por exemplo, o método cadastrarLoja cria um objeto Loja com os dados fornecidos e chama o método createLoja do LojaController para persistir a loja no sistema. Esse método é responsável por cadastrar uma nova loja no marketplace. O cliente não precisa conhecer os detalhes da implementação complexa envolvida na criação e persistência da loja, pois a fachada fornece uma interface simples e direta para realizar essa operação.

A classe de serviço que é o Service, se utiliza do padrão visitor, o qual permite a separação do algoritmo do objeto que são manipulados, este é responsável pela validação dos dados, busca no banco de dados, atualização e exclusão dos registros correspondentes.

Outro padrão de projeto utilizado é o Repository e tem como objetivo fornecer uma camada de abstração entre o código de negócio e a forma como os dados são persistidos, permitindo uma maior flexibilidade e facilidade na manipulação dos dados.

Para aplicar o padrão Repository, foi criado uma entidade do sistema, Repository. Essas interfaces definem os métodos de acesso aos dados para cada entidade, como busca, criação, atualização e exclusão. As interfaces de repositório descrevem as operações que podem ser realizadas nas entidades, sem especificar a forma como essas operações serão implementadas. Isso permite uma maior separação de responsabilidades, pois a lógica de negócio não precisa se preocupar com os detalhes de como os dados são persistidos.

Essa separação da lógica de acesso aos dados em interfaces de repositório e classes concretas traz benefícios significativos para o sistema, por exemplo permitindo que a camada de acesso aos dados seja modificada sem afetar o restante do código.

Relatório Marketplace

MAP - Métodos Avançados de Programação

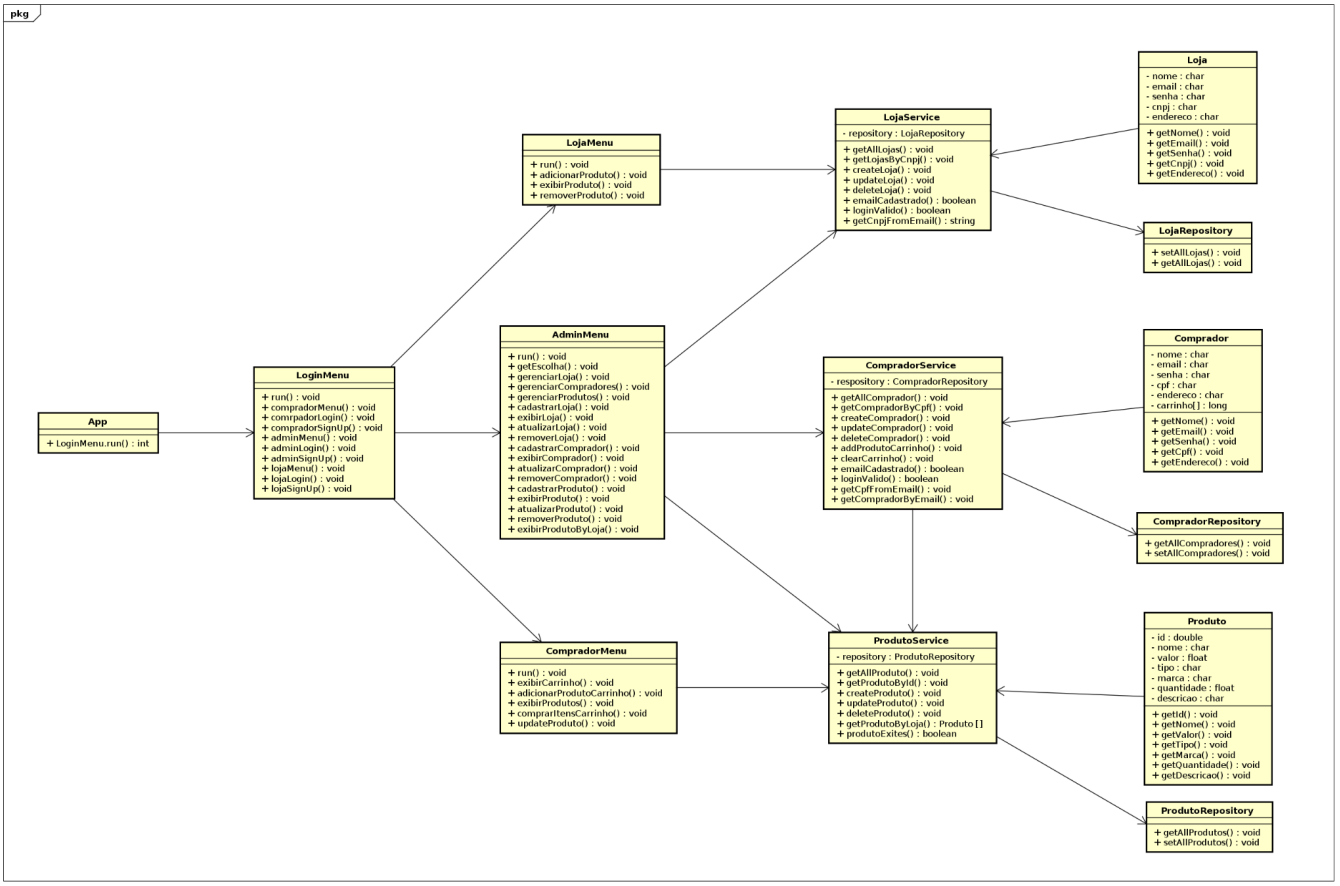
Grupo: Anna Caroline, Daniel Ribeiro, Henrique José, João Vitor, João Pedro e Luana Nóbrega

---

2. Release 02 (18/06/23)

Repositório: https://github.com/jv-guimaraes/marketplace/releases/tag/v1.0.0
Vídeo de apresentação:

2.1. Diagrama de Classe



2.2. Lista de padrões utilizados

- Singleton
- Fachada
- Visitor
- Repository

Na classe LoginMenu, foi utilizado o padrão fachada, o qual será utlizado para direcionar o tipo de login realizado, se foi feito pelo Comprador, Loja ou Admin.

O padrão singleton, por sua vez, encontra-se, agora, no AdminMenu, onde serão realizadas chamadas de gerenciamento e manutenção de Loja, Comprador e Produto, esse está ligada aos três essenciais services.

Os services se mantém como padrões visitors, pois permite a separação do algoritmo do objeto que são manipulados, este é responsável pela validação dos dados de login, busca no banco de dados, atualização e exclusão dos registros correspondentes.

O padrão repository utilizado, ainda se mantém com a função de banco, agora com os novos requisitos, está sendo utilizado como “carrinho de compras” onde é feito o armazenamento dos produtos selecionados e efetuada a compra.

3. Release 03 (29/06/23)

Os requisitos da release 3 consistem em permitir que o comprador avalie a compra e/ou o vendedor, além de possuir uma pontuação que lhe garanta vantagens, como frete grátis.

Para atender ao requisito 7, foram feitas as seguintes alterações:

- Adição de um método chamado adicionarNota na classe ProdutoService. Esse método recebe o ID do produto e a nota dada pelo comprador como parâmetros.
- O método adicionarNota percorre a lista de produtos cadastrados e, ao encontrar o produto com o ID correspondente, chama o método addNotaProduto do objeto produto para adicionar a nota.
- Após a atualização das notas, a lista de produtos é atualizada no repositório chamando o método setAllProdutos.

Além das alterações mencionadas anteriormente, se fez necessário implementar a lógica para apresentar um conceito (ruim, médio, bom, excelente) com base na avaliação dada pelo comprador. Para isso, foram adicionados os seguintes métodos:

- Adição do método getConceitoAvaliacao na classe ProdutoService. Esse método recebe o ID do produto como parâmetro e retorna o conceito da avaliação. O conceito é determinado com base na média das notas atribuídas ao produto. Por exemplo, se a média das notas for menor que 2, o conceito será "ruim". Se a média for maior ou igual a 2 e menor que 3, o conceito será "médio", e assim por diante.
- O método adicionarNota foi atualizado para chamar o método getConceitoAvaliacao após adicionar a nota ao produto. Em seguida, o conceito é armazenado no objeto produto.

Para atender ao requisito 8 continuamos com as funcionalidades da classe CompradorSrvices sendo elas:

- Adição de um método chamado addProdutoCarrinho na classe CompradorService. Esse método recebe o CPF do comprador e o ID do produto a ser adicionado ao carrinho como parâmetros.
- O método addProdutoCarrinho verifica se o produto existe chamando o método produtoExiste do ProdutoService.
- Caso o produto exista, o método percorre a lista de compradores cadastrados e, ao encontrar o comprador com o CPF correspondente, chama o método addProdutoCarrinho do objeto comprador para adicionar o ID do produto ao carrinho.
- Após a atualização do carrinho, a lista de compradores é atualizada no repositório chamando o método setAllCompradores.

Em seguida foram feitas as alterações para tornar possível o requisito 8, sendo elas:

- Adição do método incrementarPontuacao na classe CompradorService. Esse método recebe o CPF do comprador como parâmetro e incrementa a pontuação do comprador em uma unidade.

- O método incrementarPontuacao percorre a lista de compradores cadastrados e, ao encontrar o comprador com o CPF correspondente, chama o método incrementarPontuacao do objeto comprador.
- O método adicionarNota foi atualizado para chamar o método incrementarPontuacao após adicionar a nota ao produto e antes de atualizar a lista de produtos no repositório.

Para facilitar a implementação dos requisitos, a classe Produto possui um método addNotaProduto que permite adicionar uma nota ao produto.

No contexto de ambos os requisitos também foram feitas alterações relevantes nas entidades adicionando os atributos necessários e na classe LojaSrvices, onde foi adicionado o método adicionarNota(String cnpj, int nota). Esse método é responsável por adicionar uma nota à loja correspondente ao CNPJ fornecido.

Essas foram as principais alterações realizadas no código para atender aos requisitos 7 e 8, requisitos finais para a conclusão do projeto. Com essas implementações, o comprador pode avaliar a compra e/ou vendedor, fornecendo uma nota e um comentário. Além disso, o sistema calcula um conceito para a avaliação do produto e incrementa a pontuação do comprador a cada compra e avaliação realizada.