Universidade	Universidade Estadual de Goiás
Disciplina	Qualidade de software
Versão do documento	V01
QA	Augusto Henrique
Professor:	Rafael Viana
Data de criação dos cenários	05/12/2024
Data da inicio dos testes	05/12/2024

Desenvolvimento	Projeto estrutural: Métodos e Classes

1) **Classe** - **Produto**: Possui os atributos cujo tipo está especificado. Os Métodos de acesso **SETS** e **GETS,** foram implementado. Foi criado uma sobrecarga do método construtor e um para instanciar objeto.

```java
package com.GestaoProduto.Model;

public class Produto {
    //=================================>[VARIAVEIS]<============================
    private String codigo;
    private String nome;
    private String descricao;
    private Double preco;
    //=================================>[CONSTRUTOR]<===========================
    public Produto(String CODIGO, String NOME, Double PRECO,String DESCRICAO){
        this.codigo= CODIGO;
        this.nome= NOME;
        this.preco= PRECO;
        this.descricao= DESCRICAO;
    }
    public Produto(){}
    //=================================>[SETS]<=================================
    public void setCodigo(String CODIGO){ this.codigo= CODIGO; }
    public void setNome(String NOME){ this.nome= NOME; }
    public void setDescricao(String DESCRICAO){ this.descricao= DESCRICAO; }
    public void setPreco(Double PRECO){ this.preco= PRECO; }
    //=================================>[GETS]<=================================
    public String getCodigo(){ return codigo; }
    public String getNome(){ return nome; }
    public String getDescricao(){ return descricao; }
    public Double getPreco(){ return preco; }
    @Override
    public String toString() {
        return "[Codigo: "+codigo+" \tNome: "+nome+" \t\tPreço: "+preco+" \tDescrição: "+descricao+" ]\n";
    }
}
```

2) **Classe - User**: Possui os atributos cujo tipo está especificado. Os Métodos de acesso SETS e GETS, foram implementados. Foi criado uma sobrecarga do método construtor e um para instanciar objeto.

```java
package com.GestaoProduto.Model;

public class User {
    //==================================>[VARIAVEIS]<============================
    private int id;
    private String account;
    private String password;
    //==================================>[CONSTRUTOR]<============================
    public User(int ID, String ACCOUNT, String PASSWORD){
        this.id= ID;
        this.account= ACCOUNT;
        this.password= PASSWORD;
    }
    public User(){}
    //==================================>[SETS]<==================================
    public void setID(int ID){ this.id= ID; }
    public void setAccount(String ACCOUNT){ this.account= ACCOUNT; }
    public void setPassword(String PASSWORD){ this.password= PASSWORD; }
    //==================================>[GETS]<==================================
    public int getID(){ return id; }
    public String getAccount(){ return account; }
    public String getPassword(){ return password; }
    @Override
    public String toString() {
        return "[ id: "+id+" account: "+account+" password: "+password+" ]\n";
    }
}
```

3) **Databases** – Foi criando uma classe Database, para armazenar uma lista de objetos do tipo **User** e **Produto**. Por onde é adicionado, consultado, alterado e deletado.

```java
package com.GestaoProduto.Model;

import java.util.ArrayList;

public class DataBase {
    //==================================>[VARIAVEIS]<==============================
    private ArrayList<User> users= new ArrayList<>();
    private ArrayList<Produto> produtos= new ArrayList<>();
    //==================================>[USER]<====================================
    public ArrayList <User> getAllUsers(){ return users; }
    public void setUserDataBase(User OBJ){
    public void UpdateUser(String OBJ){
    public void DeleteUser(String ACCOUNT){
    public void Login(String Account, String Password){
    //==================================>[PRODUTO]<=================================
    public ArrayList <Produto> getAllProdutos(){ return produtos; }
    public void setProdutoDatabase(Produto OBJ){
    public void UpdateProduto(String OBJ){
    public void DeleteProduto(String CODIGO){
}
```

3.1 [**User**]: A baixo especifica o funcionamento dos métodos.

- **getAllUsers**: Retorna todos dados armazenado dentro do arranjo.

- **setUserDataBase**: Tem como receber um objeto no parâmetro do tipo **USER**. Em seguida, atribui-se o tamanho desse arranjo com "users.size()". Percorre-se nesse arranjo se existe outro dado igual, comparando e passando por todas as restrições. Ao passar por todas elas, armazena-se nessa lista de **USER**: this.users.add(**OBJ**); Se algumas delas não passar pela restrição, lançará uma exceção.

- **UpdateUser**: Tem como receber uma string no parâmetro . Pega o tamanho do arranjo. Percorre toda a lista de USER, quando achar a conta para alterar, na posição do arranjo mostrará tela de **inputs** e **outputs**, para leitura de dados. Em seguida grava novamente e sai. Se não encontrar o parâmetro requisitado irá lança exceção;

- **DeleteUser**: Tem como receber uma string no parâmetro . Pega o tamanho do arranjo. Percorre toda a lista de USER, quando achar a conta para deletar a remove do arranjo passando a posição, com: this.users.remove(**i**); Se não encontrar lança exceção;

- **Login**: Recebe dois parâmetros do tipo string, **account** e **password**. Percorre a lista ao mesmo tempo comparando a existência desse registro. Se encontrado, ele entra, caso contrário lança exceção.

3.2 [**Produto**]:

- **getAllProdutos**: Retorna todos dados armazenado dentro do arranjo.

- **setProdutoDatabase**: Tem como receber um objeto no parâmetro do tipo **Produto**. Em seguida, atribui-se o tamanho desse arranjo com "produto.size()". Percorre-se nesse arranjo se existe outro dado igual, comparando e passando por todas as restrições. Ao passar por todas elas, armazena-se nessa lista de **PRODUTO**: this.produto.add(**OBJ**); Se algumas delas não passar pela restrição, lançará uma exceção.

- **UpdateProduto**: Tem como receber uma string no parâmetro . Pega o tamanho do arranjo. Percorre toda a lista de Produto, quando achar a conta para alterar, na posição do arranjo mostrará tela de **inputs** e **outputs**, para leitura de dados. Em seguida grava novamente e sai. Se não encontrar o parâmetro requisitado irá lança exceção;

- **DeleteProduto**: Tem como receber uma string no parâmetro . Pega o tamanho do arranjo. Percorre toda a lista de **PRODUTO**, quando achar a conta para deletar a remove do arranjo passando a posição, com: this.produto.remove(**i**); Se não encontrar lança exceção;

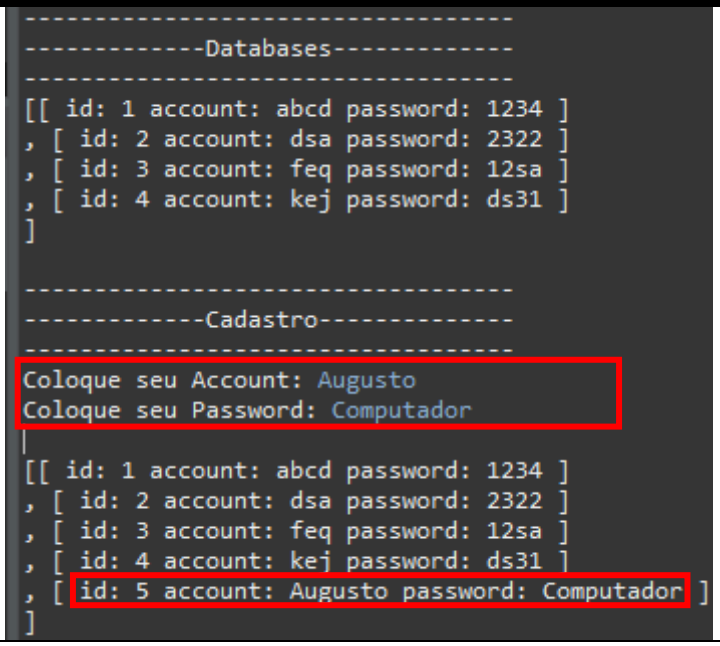
Escopo dos Testes	Gestão de Produtos

ID	Caso de Teste	STATUS
CT1. 01	[**User**] Validar função de cadastro: setUserDataBase.	PASSED
CT1. 02	[**User**] Validar função UpdateUser.	PASSED
CT1. 03	[**User**] Validar função UpdateUser,com parâmetro invalido.	PASSED
CT1. 04	[**User**] Validar função DeleteUser.	PASSED
CT1. 05	[**User**] Validar função DeleteUser, com parâmetro invalido.	PASSED
CT1. 06	[**User**][**GetAll**] Validar consulta database.	PASSED
CT1. 07	[**User**] Validar função de login.	PASSED
CT1. 08	[**User**][**Login**] Validar parâmetro inválido.	PASSED
CT1. 09	[**User**][**Account**] Validar parâmetro já existente na base de dados.	PASSED
CT1. 10	[**User**][**Account**][**Password**] Validar parâmetro igual.	PASSED
CT1. 11	[**User**][**Password**] Validar parâmetro menor igual a 4 caracteres.	PASSED
CT1. 12	[**User**][**Password**] Validar parâmetro, com espaço no INICIO.	PASSED
CT1. 13	[**User**][**Password**] Validar parâmetro, com espaço no FIM.	PASSED
CT2. 01	[**Produto**] Validar função de cadastro: setProdutoDataBase.	PASSED
CT2. 02	[**Produto**] Validar função UpdateProduto.	PASSED
CT2. 03	[**Produto**] Validar função UpdateProduto,com parâmetro invalido.	PASSED
CT2. 04	[**Produto**] Validar função DeleteProduto.	PASSED
CT2. 05	[**Produto**] Validar função DeleteProduto, com parâmetro invalido.	PASSED
CT2. 06	[**Produto**][**GetAll**] Validar consulta database.	PASSED
CT2. 07	[**Produto**] [**Código**] Validar parâmetro já existente na base de dados.	PASSED
CT2. 08	[**Produto**] [**Nome**] Validar parâmetro vazio.	PASSED
CT2. 09	[**Produto**] [**Descrição**] Validar parâmetro superior a 200 caracteres.	PASSED
CT2. 10	[**Produto**] [**Preço**] Validar parâmetro menor igual a 0.	PASSED

CT1. 01	[User] Validar função de cadastro: setUserDataBase.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir elementos válidos para: Account e Password.
When instancio a função “setUserDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá gravar o parâmetro na database do sistema User.

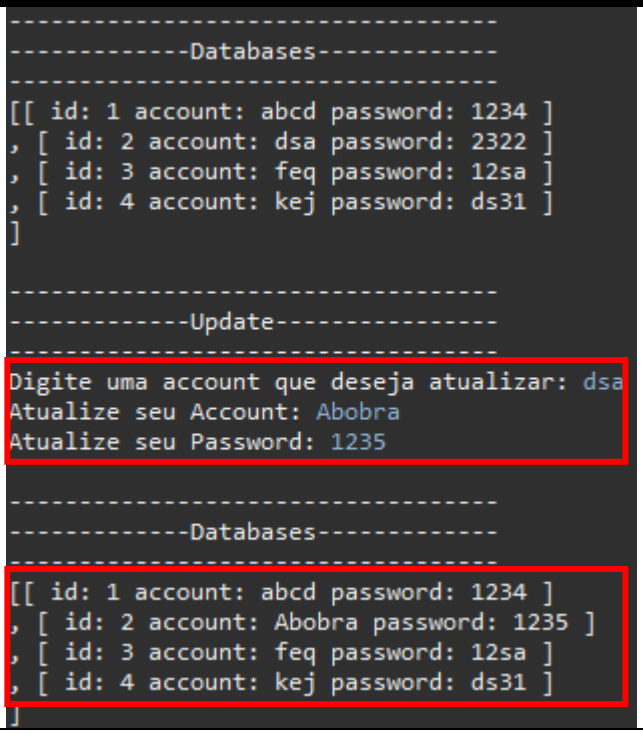
Cadastro



CT1. 02	[User] Validar função UpdateUser.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir elementos válidos para: Account e Password.
When instancio a função “UpdateUser” passando os atributos no parâmetro
And executar a aplicação
Then deverá gravar o parâmetro na database do sistema User.

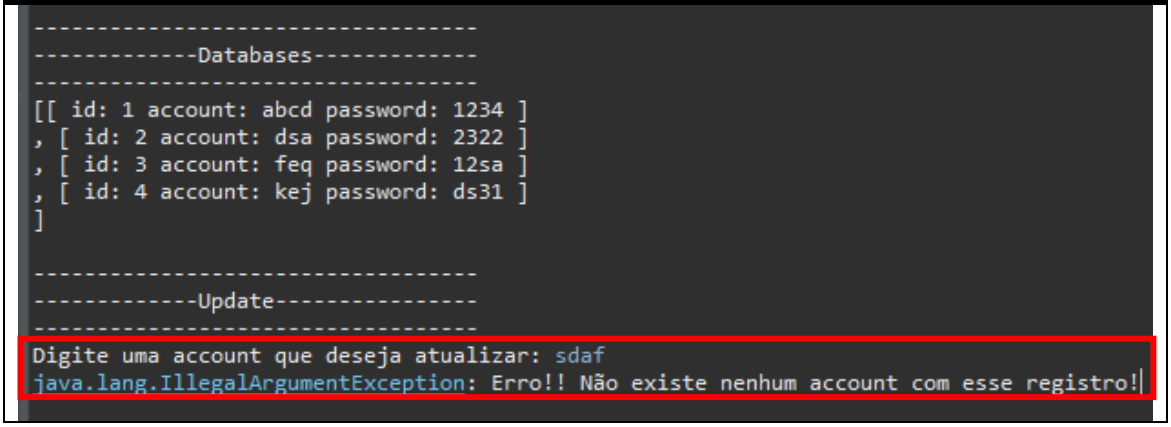
User



CT1. 03	[User] Validar função UpdateUser,com parâmetro invalido.
Versão	V01
Status	PASSED

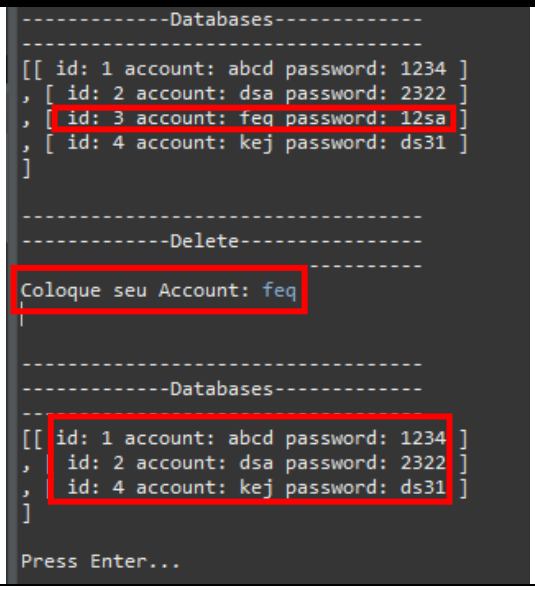
Given ao criar um objeto User e DataBase
And atribuir elementos inválidos para: Account e Password.
When instancio a função “UpdateUser” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: “Erro!! Não existe nenhum account com esse registro!”

User



CT1. 04	[User] Validar função DeleteUser.
Versão	V01
Status	PASSED

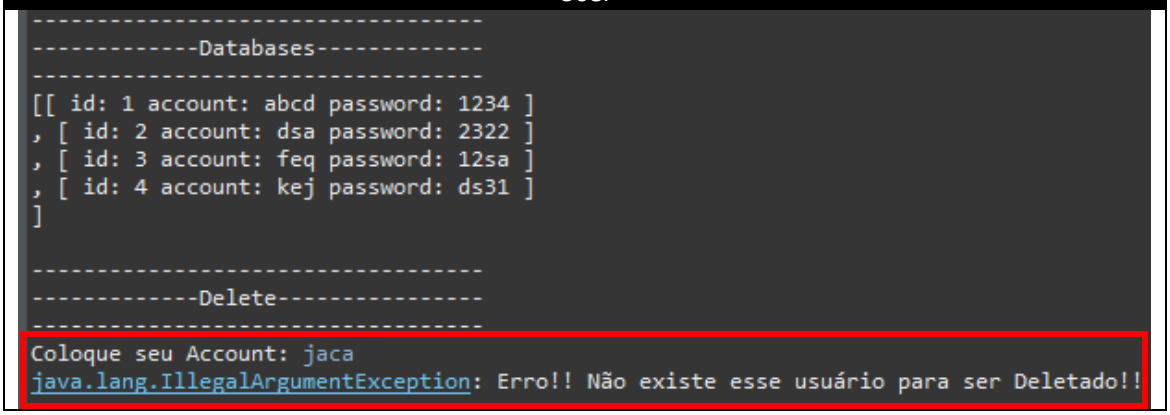
Given ao criar um string Account e um Objeto DataBase
And atribuir elemento válido para: Account.
When chamo a função “DeleteUser” passando o atributo no parâmetro
And executar a aplicação
Then deverá ser deletado o registro na database do sistema User.



CT1. 05	[User] Validar função DeleteUser, com parâmetro invalido.
Versão	V01
Status	PASSED

Given ao criar um string ACCOUNT
And atribuir elemento um inválido para: Account.
When chamo a função “DeleteUser” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! Não existe esse usuário para ser Deletado!!".

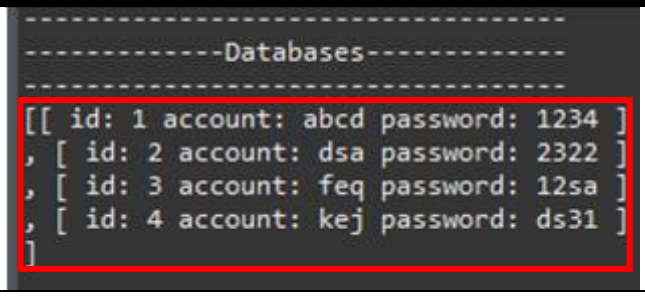
User



CT1. 06	[User][GetAll] Validar consulta database.
Versão	V01
Status	PASSED

Given ao criar um objeto DataBase
When instanciar o método getAllUsers.
And executar a aplicação
Then deverá retornar todos os registros na database do sistema User.

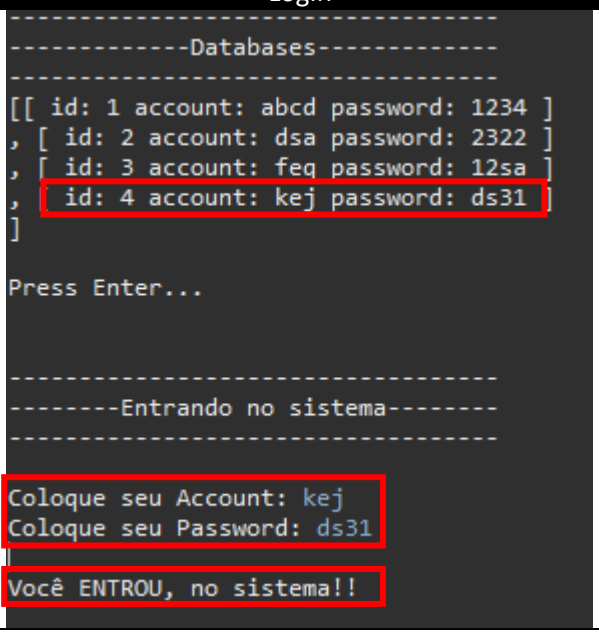
User



CT1. 07	[User] Validar função de login.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir elementos válidos para: Account e Password.
When instancio a função “Login” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar a mensagem: “Você ENTROU, no sistema!!”

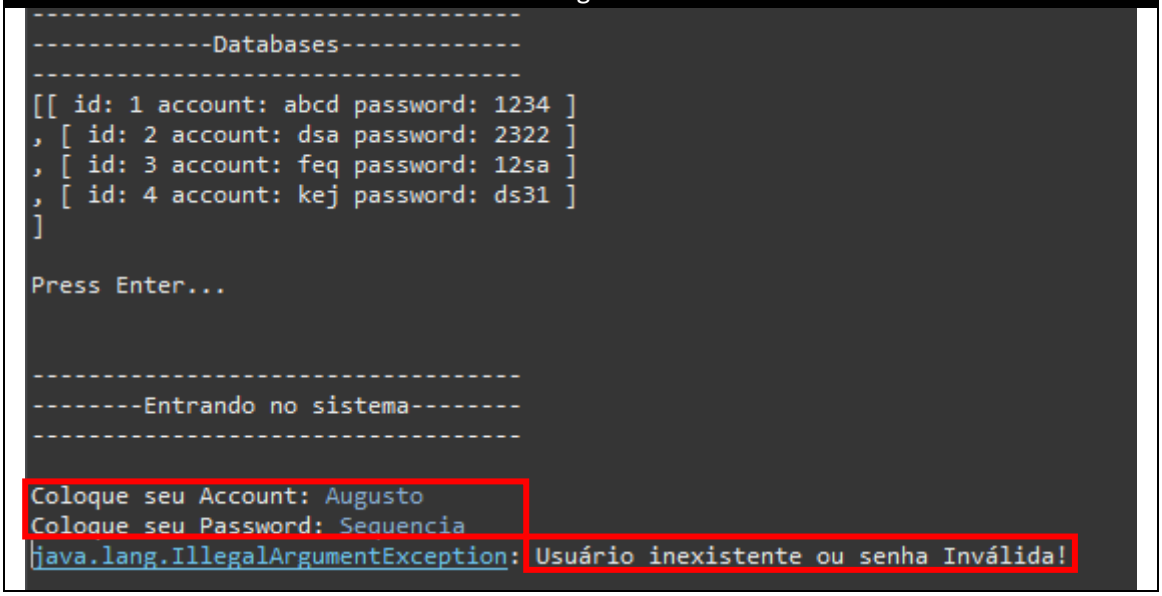
Login



CT1. 08	[User][Login] Validar parâmetro inválido.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir elementos inválidos para: Account e Password.
When instancio a função “Login” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar a mensagem: “Usuário inexistente ou senha Inválida!”

Login



CT1. 09	[User][Account] Validar parâmetro já existente na base de dados.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir um elemento já cadastro para: Account
When instancio a função “setUserDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: “Erro!! Já existe um USER com o nome: [], já cadastrado!!""

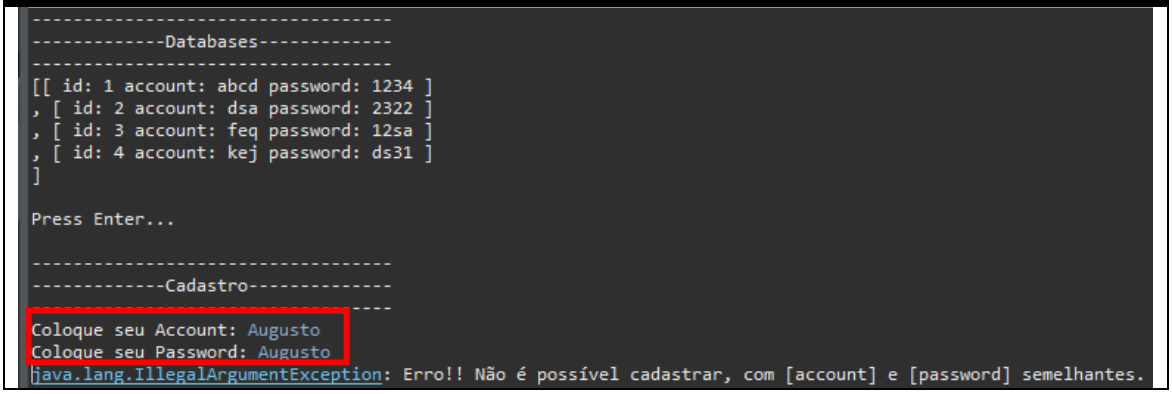
Login

```
-----------------------------------
-------------Databases-------------
-----------------------------------
[[ id: 1 account: abcd password: 1234 ]
, [ id: 2 account: dsa password: 2322 ]
, [ id: 3 account: feq password: 12sa ]
, [ id: 4 account: kej password: ds31 ]
]

Press Enter...

-----------------------------------
-------------Cadastro--------------
-----------------------------------
Coloque seu Account: kej
Coloque seu Password: 1232
java.lang.IllegalArgumentException: Erro!! Já existe um USER com o nome: [kej], já cadastrado!!
```

CT1. 10	[User][Account][Password] Validar parâmetro igual.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir um elemento que seja igual para: Account e Password
When instancio a função “setUserDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! Não é possível cadastrar, com [account] e [password] semelhantes."

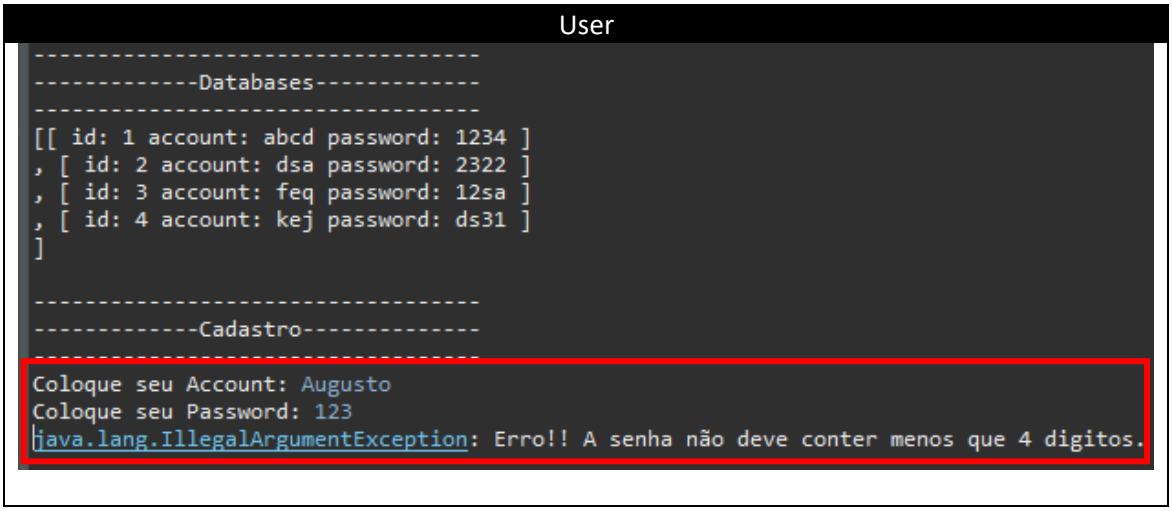
Semelhantes



CT1. 11	[User][Password] Validar parâmetro menor igual a 4 caracteres.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir um elemento que seja menor igual a 4 caracteres para: Password
When instancio a função “setUserDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem "Erro!! A senha não deve conter menos que 4 digitos."

User



CT1. 12	[User][Password] Validar parâmetro, com espaço no INICIO.
Versão	V01
Status	PASSED -> (CORRIGIDO 02/07/2025 – function **.startsWith(" ");**)

Given ao criar um objeto User e DataBase
And atribuir um elemento no inicio que seja um espaço para: Password
When instancio a função “setUserDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! A senha não deve conter espaços no inicio e nem no final!"

```
-----------------------------------
-------------Cadastro--------------
-----------------------------------
Coloque seu Account: Augusto
Coloque seu Password:  test321
java.lang.IllegalArgumentException: Erro!! A senha não deve conter espaços no inicio e nem no final!
```

CT1. 13	[User][Password] Validar parâmetro, com espaço no FIM.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir um elemento no final que seja um espaço para: Password
When instancio a função “setUserDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! A senha não deve conter espaços no inicio e nem no final!"

User

```
-----------------------------------
-------------Databases-------------
-----------------------------------
[[ id: 1 account: abcd password: 1234 ]
, [ id: 2 account: dsa password: 2322 ]
, [ id: 3 account: feq password: 12sa ]
, [ id: 4 account: kej password: ds31 ]
]

-----------------------------------
-------------Cadastro--------------
-----------------------------------
Coloque seu Account: Augusto
Coloque seu Password: Lol2
java.lang.IllegalArgumentException: Erro!! A senha não deve conter espaços no inicio e nem no final!
```

CT2. 01	[Produto] Validar função de cadastro: setProdutoDataBase.
Versão	V01
Status	PASSED

Given ao criar um objeto Produto e DataBase
And atribuir elementos válidos para: Codigo, Nome, Descrição e Preço.
When instancio a função “setProdutoDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá gravar o parâmetro na database do sistema Produto.

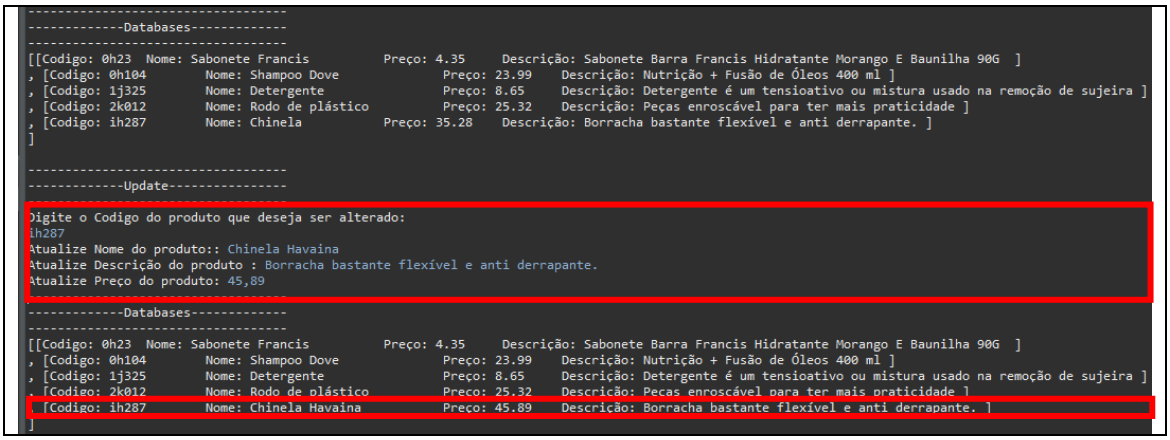
Produto

```
-----------------------------------
------------Cadastro-------------

Digite o Código do produto: ih287
Digite o Nome do produto: Chinela
Digite o Descrição do produto: Borracha bastante flexível e anti derrapante.
Digite o Preço do produto: 35,28

Completed register databases...
-----------------------------------
------------Databases------------
-----------------------------------
[[Codigo: 0h23  Nome: Sabonete Francis          Preço: 4.35     Descrição: Sabonete Barra Francis Hidratante Morango E Baunilha 90G  ]
, [Codigo: 0h104        Nome: Shampoo Dove              Preço: 23.99    Descrição: Nutrição + Fusão de Óleos 400 ml ]
, [Codigo: 1j325        Nome: Detergente                Preço: 8.65     Descrição: Detergente é um tensioativo ou mistura usado na remoção de sujeira ]
, [Codigo: ih287        Nome: Chinela           Preço: 35.28    Descrição: Borracha bastante flexível e anti derrapante. ]
]
```

CT2. 02	[Produto] Validar função UpdateProduto.
Versão	V01
Status	PASSED

Given ao criar um objeto Produto e DataBase
And atribuir elementos válidos para: Codigo
When instancio a função “UpdateProduto” passando os atributos no parâmetro
And executar a aplicação
Then Then deverá gravar o parâmetro na database do sistema Produto.

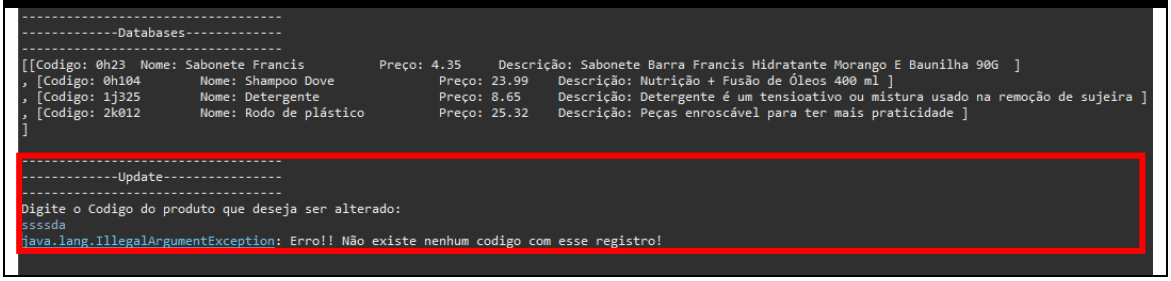
Produto



CT2. 03	[Produto] Validar função UpdateProduto,com parâmetro invalido.
Versão	V01
Status	PASSED

Given ao criar um objeto User e DataBase
And atribuir elementos inválidos para: codigo
When instancio a função “UpdateUser” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: “Erro!! Não existe nenhum código com esse registro!”

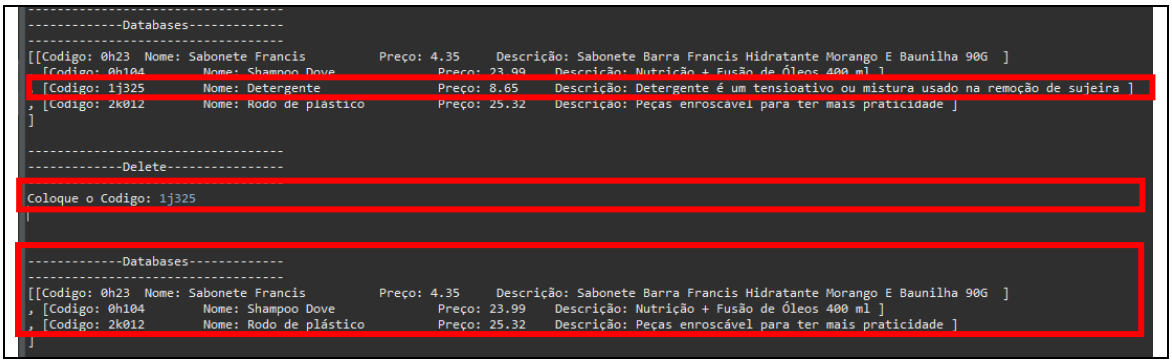
Produto



CT2. 04	[Produto] Validar função DeleteProduto.
Versão	V01
Status	PASSED

Given ao criar um string Codigo e um objeto DataBase
And atribuir elemento válido para: Codigo
When instancio a função “setProdutoDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá deletar o registro na database do sistema Produto.

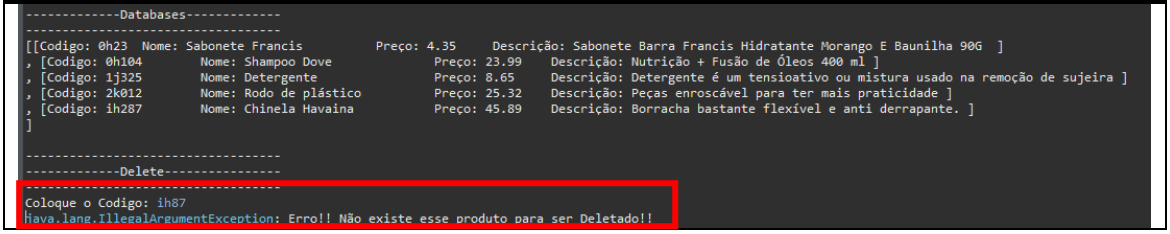
Produto



CT2. 05	[Produto] Validar função DeleteProduto, com parâmetro invalido.
Versão	V01
Status	PASSED

Given ao criar um string Codigo e um objeto DataBase
And atribuir elemento válido para: Codigo
When instancio a função “DeleteProduto” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: “Erro!! Não existe esse produto para ser Deletado!!”

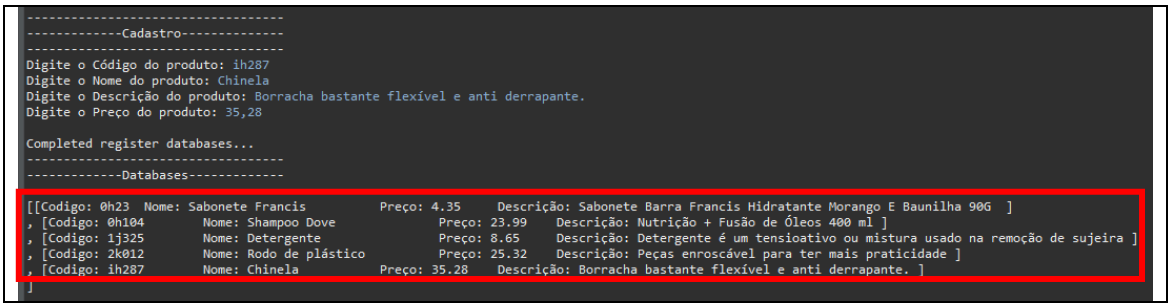
Produto



CT2. 06	[Produto][GetAll] Validar consulta database.
Versão	V01
Status	PASSED

Given ao criar um objeto DataBase
When instanciar o método getAllProdutos.
And executar a aplicação
Then deverá retornar todos os registros na database do sistema Produto.

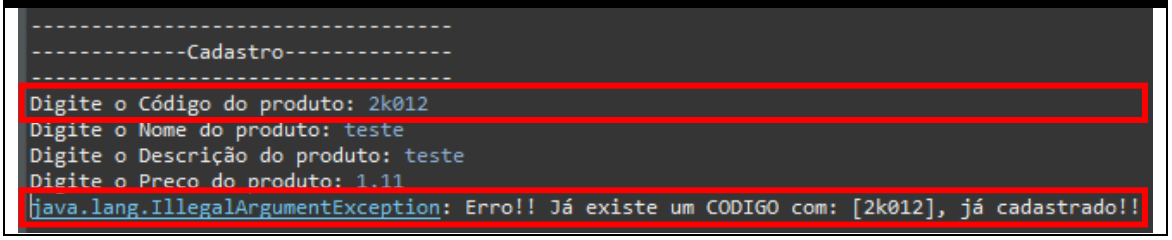
Produto



CT2. 07	[Produto] [Código] Validar parâmetro já existente na base de dados.
Versão	V01
Status	PASSED

Given ao criar um objeto Produto e DataBase
And atribuir um elemento já cadastro para: Codigo
When instancio a função “setProdutoDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! Já existe um CODIGO com: [já cadastrado!!"

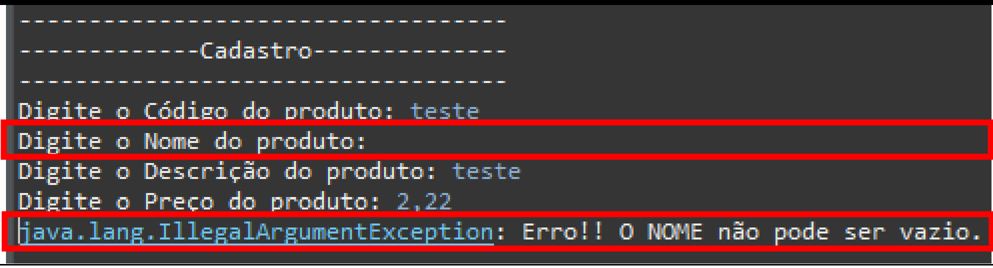
Produto



CT2. 08	[Produto] [Nome] Validar parâmetro vazio.
Versão	V01
Status	PASSED

Given ao criar um objeto Produto e DataBase
And atribuir um elemento vazio para: Nome
When instancio a função “setProdutoDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: Erro!! O NOME não pode ser vazio."

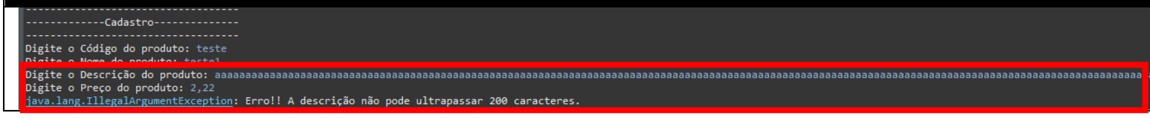
Produto



CT2. 09	[Produto] [Descrição] Validar parâmetro superior a 200 caracteres.
Versão	V01
Status	PASSED

Given ao criar um objeto Produto e DataBase
And atribuir um elemento que seja superior a 200 caracteres para: Descrição
When instancio a função “setProdutoDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! A descrição não pode ultrapassar 200 caracteres."

Produto



CT2. 10	[Produto] [Preço] Validar parâmetro menor igual a 0.
Versão	V01
Status	PASSED

Given ao criar um objeto Produto e DataBase
And atribuir um elemento menor igual a 0 para: Preço
When instancio a função “setProdutoDataBase” passando os atributos no parâmetro
And executar a aplicação
Then deverá retornar mensagem: "Erro!! O PREÇO não pode ser igual e nem menor que 0."

Produto