Nomes: Lucas Domingos

Kevin Mikio

**Requisitos**

Requisitos funcionais:

* Logar o usuário
* O usuário pode buscar o seu banco por nome
* O usuário pode visualizar a lista de bancos existentes
* Buscar o saldo do usuário
* Pagar a conta
* Notificar o usuário se há conta em atraso
* O Administrador pode adicionar novos bancos
* O Administrador pode editar

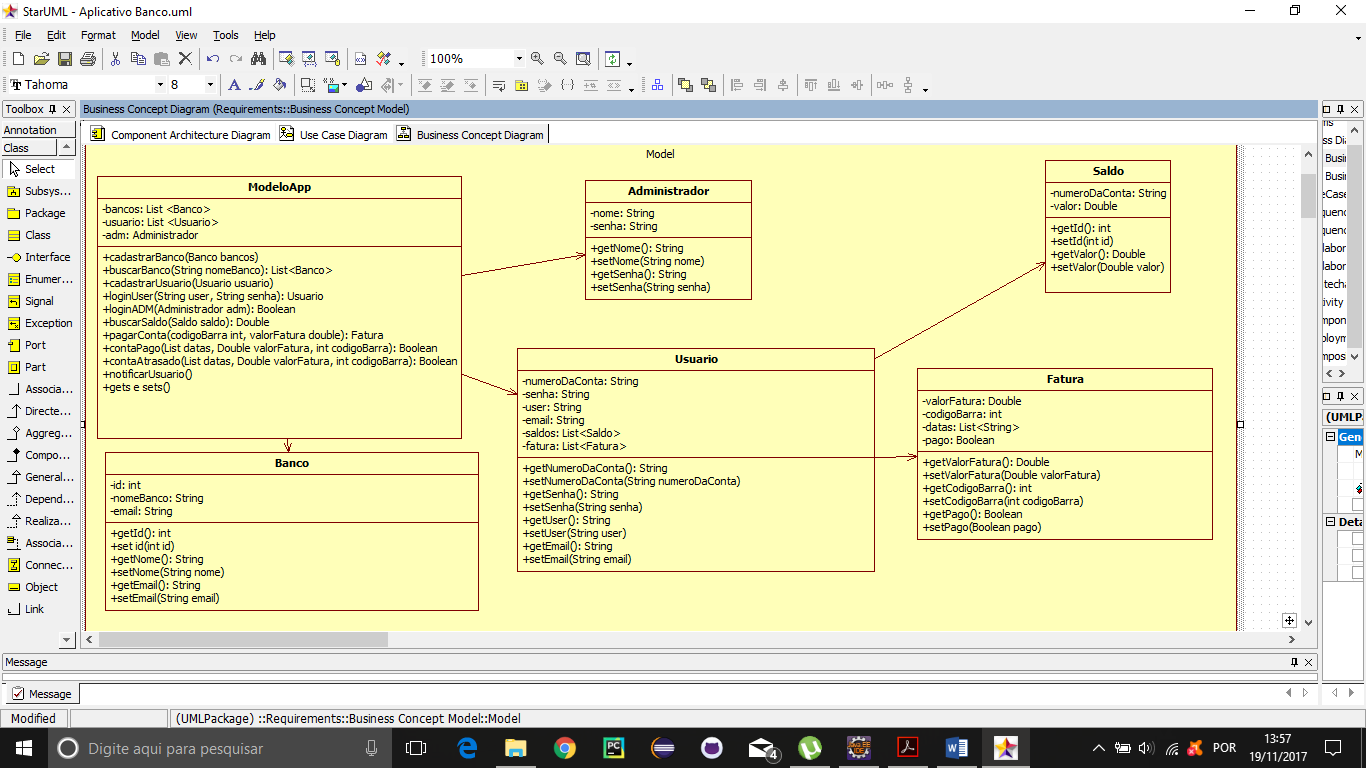
Requisitos não funcionais:

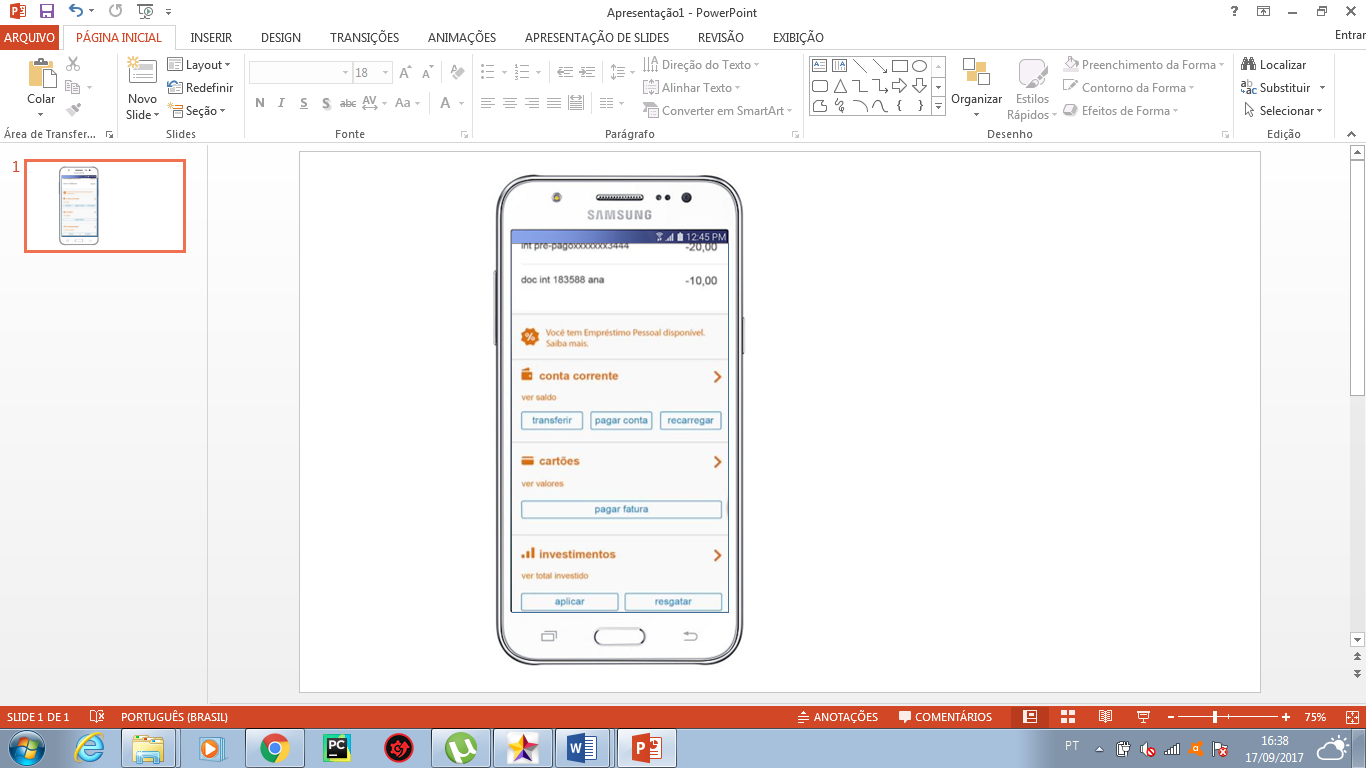
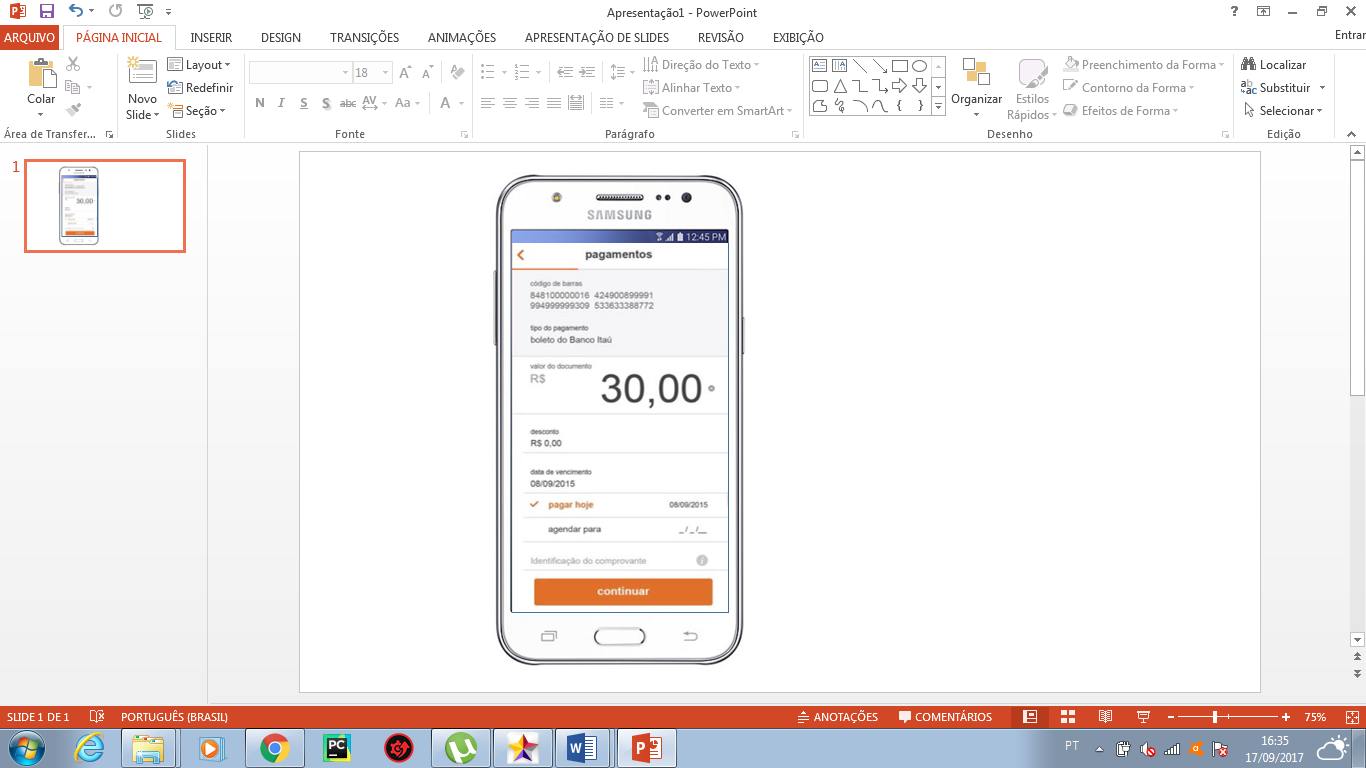
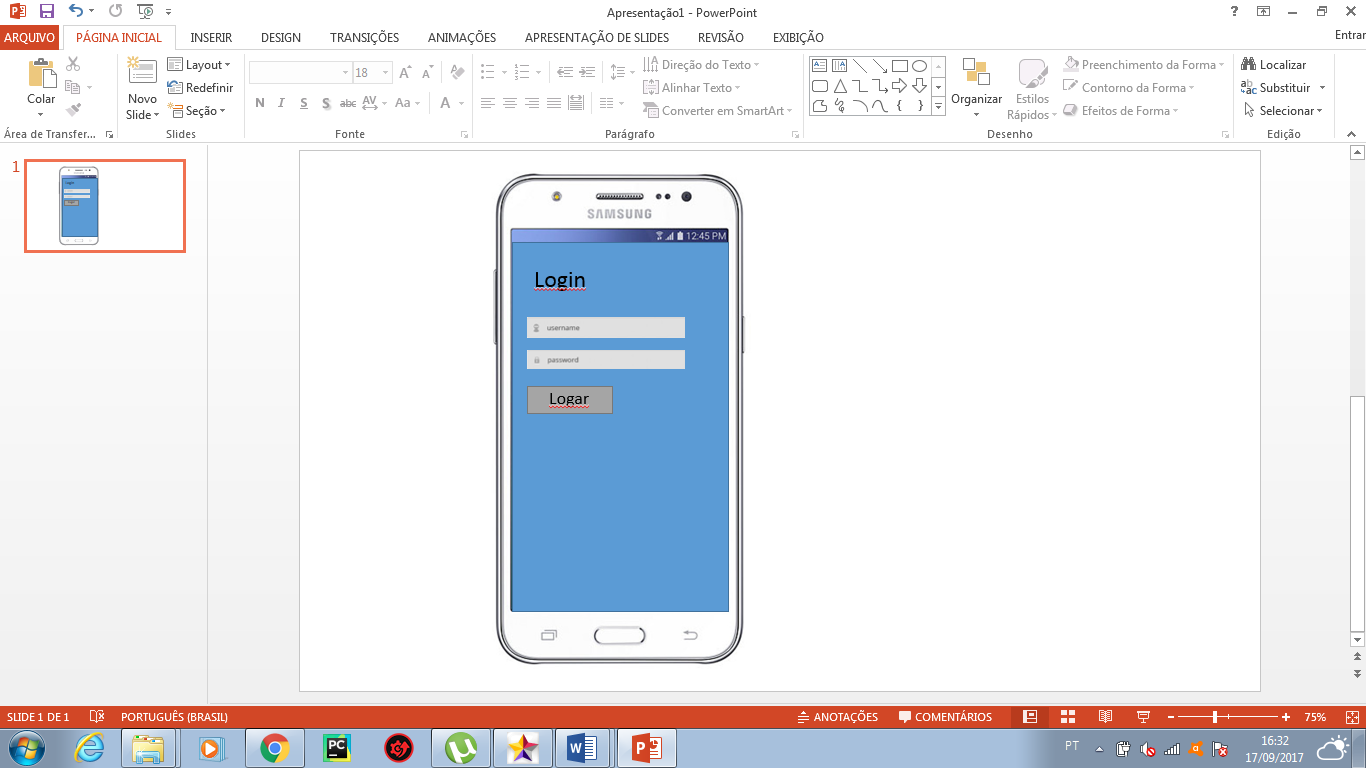
* Segurança (Já que estamos manuseando dados referentes a números de contas e seus saldos)
* Interface de fácil utilização
* Minimalista
* Usabilidade
* Padronização
* Compatibilidade (Em várias plataformas mobile)
* Disponibilidade (nos casos de dúvidas)

**Projeto**

Será um projeto dedicado a leigos da tecnologia, onde o aplicativo desenvolvido será a porta de entrada para aqueles que buscam facilidade relacionados a setores bancários, como saque, pagamentos de contas, controle de gastos, etc.

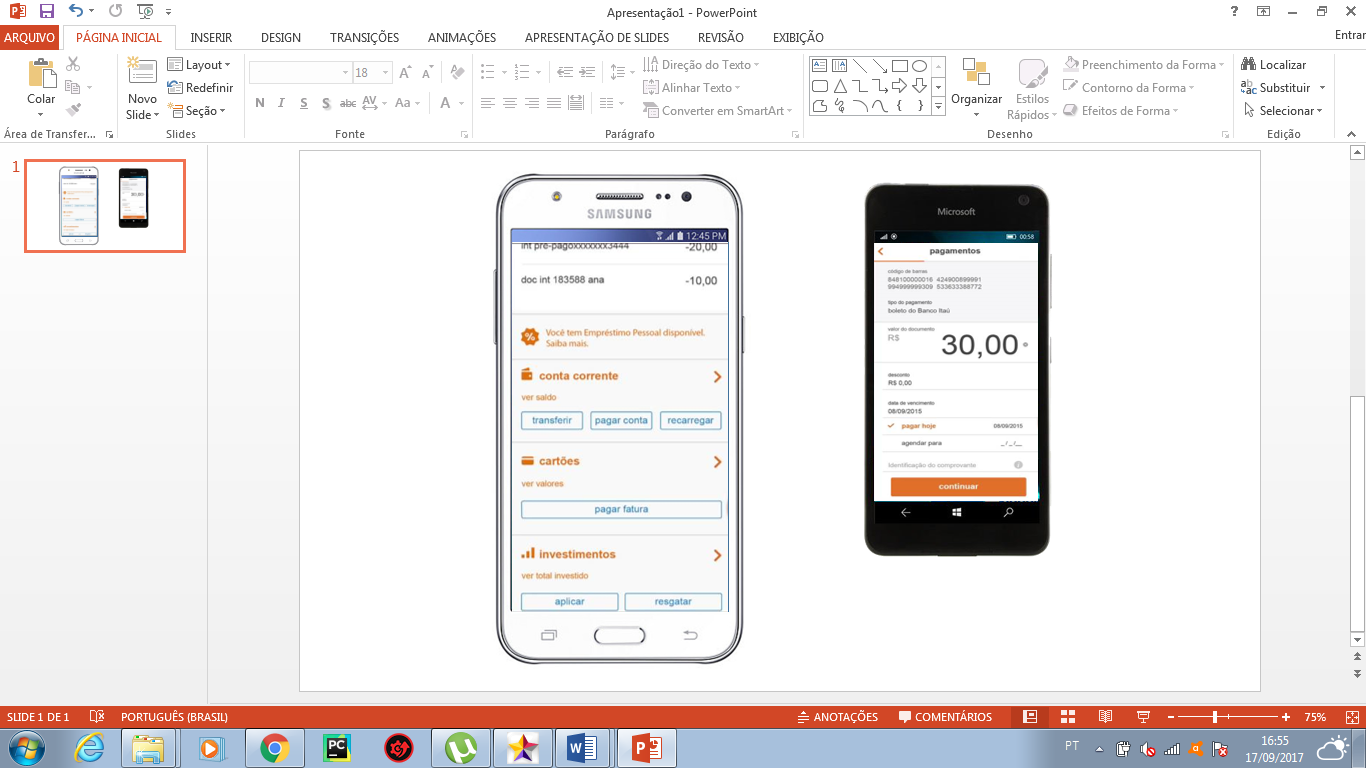
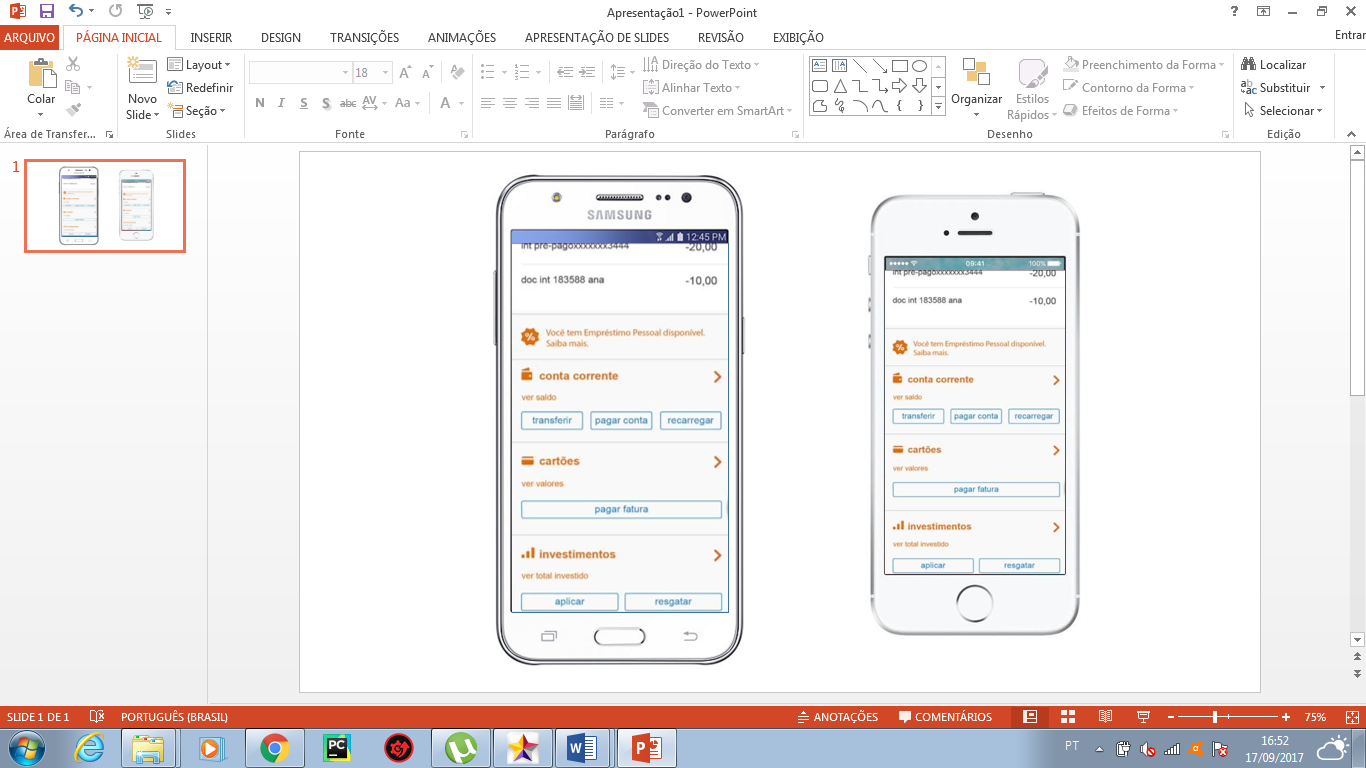
Além de toda facilidade proposto pelo aplicativo, terá como intuito a padronização do mesmo, pois ela contara com uma imensa lista de bancos onde, se quiser trocar simplesmente de um banco para o outro, basta modifica-lo, não perdendo a sua usabilidade.





1. Manter o usuário sempre informado de qual janela o aplicativo de encontra.

8- Estética e Design Minimalista.



4-O usuário não deve se preocupar com as plataformas que irão usar, pois o aplicativo se adapta de acordo com quais quer sistema.

5-Prevenção de erros, para que o usuário tenha o app sempre à disposição.

<https://github.com/kevin799/Java_Eclipse/tree/master/workspace/APPBank>

Modelo.java

**package** APPBank;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.time.LocalDate;

**public** **class** Modelo {

**private** List<Usuario> usuarios = **new** LinkedList<Usuario>();

**private** List<Banco> bancos = **new** LinkedList<Banco>();

**private** List<Admin> admins = **new** LinkedList<Admin>();

**public** **void** cadastrarUsuario(Usuario usuario){

**if** (!usuario.getUser().equals("")||!usuario.getSenha().equals("")||!usuario.getEmail().equals("")||!usuario.getNumeroDaConta().equals("")){

**if**(contasRepetidas(usuario.getEmail(),usuario.getNumeroDaConta())){

usuarios.add(usuario);

}

}

}

**public** **void** cadastrarBanco(Banco banco){

**if**(!(banco.getId()==0) || !banco.getNomeBanco().equals("")|| !banco.getEmail().equals("")){

**if**(bancosRepetidos(banco.getNomeBanco(),banco.getId())){

bancos.add(banco);

}

}

}

**public** **void** cadastrarAdmin(Admin admin){

**if**(adminRepetidas(admin.getNomeAdmin(),admin.getSenhaAdmin())){

admins.add(admin);

}

}

**public** List<Usuario> buscarUsuarioPorUser(String user){

List<Usuario> userEncontrados = **new** LinkedList<Usuario>();

**for**(Usuario usuario: **this**.usuarios){

**if**(usuario.getUser().equals(user)) userEncontrados.add(usuario);

}

**return** userEncontrados;

}

**public** **boolean** adminRepetidas(String nomeAdmin,String senhaAdmin){

**for**(Admin admin:admins){

**if** (admin.getNomeAdmin().equals(nomeAdmin) || admin.getSenhaAdmin().equals(senhaAdmin)) **return** **false**;

}

**return** **true**;

}

**public** **boolean** contasRepetidas(String email,String numeroDaConta){

**for**(Usuario usuario:usuarios){

**if**(usuario.getEmail().equals(email) || usuario.getNumeroDaConta().equals(numeroDaConta)) **return** **false**;

}

**return** **true**;

}

**public** **boolean** bancosRepetidos(String nomeBanco ,**int** id){

**for**(Banco banco:bancos){

**if**(banco.getNomeBanco().equals(nomeBanco) || banco.getId()==id) **return** **false**;

}

**return** **true**;

}

**public** List<Banco> buscarBancoPorNome(String nomeBanco){

List<Banco> nomeBancoEncontrado = **new** LinkedList<Banco>();

**for**(Banco banco: **this**.bancos){

**if**(banco.getNomeBanco().equals(nomeBanco)) nomeBancoEncontrado.add(banco);

}

**return** nomeBancoEncontrado;

}

**public** **boolean** logarUsuario(String user,String senha){

**for**(Usuario usuario:usuarios){

**if**(usuario.getUser().equals(user) && usuario.getSenha().equals(senha)) **return** **true**;

}

**return** **false**;

}

**public** Double buscarSaldo(String numeroDaContaSaldo){

**for** (Usuario usuario:usuarios){

**if**(usuario.getNumeroDaConta().equals(numeroDaContaSaldo)) **return** usuario.getSaldo().getValor();

}

**return** **null**;

}

**public** **boolean** pagarConta(**int** codigoBarra,Double valorFatura,Boolean pago){

**for** (Usuario usuario:usuarios){

**if**(usuario.getConta().getCodigoBarra()==codigoBarra && usuario.getConta().getValorFatura().equals(valorFatura)){

**if**(pago.equals(**false**)) **return** **true**;

}

}

**return** **false**;

}

**public** **boolean** isLate(LocalDate date){

LocalDate today = LocalDate.*now*();

**if**(date.isBefore(today)) **return** **true**;

**return** **false**;

}

**public** **boolean** verificaContAtr(String user,String numeroDaConta,LocalDate data){

LocalDate today = LocalDate.*now*();

**for** (Usuario usuario:usuarios){

**if** (usuario.getNumeroDaConta().equals(numeroDaConta)&& usuario.getUser().equals(user)){

**if** (data.isBefore(today))**return** **true**;

}

}

**return** **false**;

}

**public** List<Usuario> getUsuarios(){

**return** **this**.usuarios;

}

**public** List<Banco> getBancos(){

**return** **this**.bancos;

}

**public** List<Admin> getAdmin(){

**return** **this**.admins;

}

}

Admin.java

**package** APPBank;

**public** **class** Admin {

**private** String nomeAdmin;

**private** String senhaAdmin;

**public** Admin(String nomeAdmin,String senhaAdmin){

**this**.nomeAdmin = nomeAdmin;

**this**.senhaAdmin = senhaAdmin;

}

**public** String getNomeAdmin(){

**return** nomeAdmin;

}

**public** **void** setNomeAdmin(String nomeAdmin){

**this**.nomeAdmin = nomeAdmin;

}

**public** String getSenhaAdmin(){

**return** senhaAdmin;

}

**public** **void** setSenhaAdmin(String senhaAdmin){

**this**.senhaAdmin = senhaAdmin;

}

}

Banco.java

**package** APPBank;

**public** **class** Banco {

**private** **int** id;

**private** String nomeBanco;

**private** String email;

**public** Banco(**int** id, String nomeBanco, String email) {

**this**.id = id;

**this**.nomeBanco = nomeBanco;

**this**.email = email;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getNomeBanco() {

**return** nomeBanco;

}

**public** **void** setNomeBanco(String nomeBanco) {

**this**.nomeBanco = nomeBanco;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

}

Usuario.java

**package** APPBank;

**public** **class** Usuario {

**private** String user;

**private** String senha;

**private** String email;

**private** String numeroDaConta;

**private** Saldo saldo;

**private** Conta conta;

**public** Usuario(String user, String senha, String email,String numeroDaConta, Saldo saldo,Conta conta) {

**this**.user = user;

**this**.senha = senha;

**this**.email = email;

**this**.numeroDaConta = numeroDaConta;

**this**.saldo = saldo;

**this**.conta = conta;

}

**public** String getUser() {

**return** user;

}

**public** **void** setUser(String user) {

**this**.user = user;

}

**public** String getSenha() {

**return** senha;

}

**public** **void** setSenha(String senha) {

**this**.senha = senha;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getNumeroDaConta(){

**return** numeroDaConta;

}

**public** **void** setNumeroDaConta(String numeroDaConta){

**this**.numeroDaConta = numeroDaConta;

}

**public** Saldo getSaldo() {

**return** saldo;

}

**public** **void** setSaldo(Saldo saldo) {

**this**.saldo = saldo;

}

**public** Conta getConta() {

**return** conta;

}

**public** **void** setConta(Conta conta) {

**this**.conta = conta;

}

}

Conta.java

**package** APPBank;

**import** java.time.LocalDate;

**public** **class** Conta {

**public** Double valorFatura;

**public** **int** codigoBarra;

**public** Boolean pago;

**public** LocalDate data;

**public** Conta(Double valorFatura, **int** codigoBarra,Boolean pago, LocalDate data){

**this**.valorFatura = valorFatura;

**this**.codigoBarra = codigoBarra;

**this**.pago = pago;

**this**.data = data;

}

**public** LocalDate getData() {

**return** data;

}

**public** **void** setData(LocalDate data) {

**this**.data = data;

}

**public** Double getValorFatura() {

**return** valorFatura;

}

**public** **void** setValorFatura(Double valorFatura) {

**this**.valorFatura = valorFatura;

}

**public** **int** getCodigoBarra() {

**return** codigoBarra;

}

**public** **void** setCodigoBarra(**int** codigoBarra) {

**this**.codigoBarra = codigoBarra;

}

**public** Boolean getPago() {

**return** pago;

}

**public** **void** setPago(Boolean pago) {

**this**.pago = pago;

}

}

Saldo.java

**package** APPBank;

**public** **class** Saldo {

**private** String numeroDaContaSaldo;

**private** **double** valor;

**public** Saldo(String numeroDaContaSaldo, **double** valor){

**this**.numeroDaContaSaldo = numeroDaContaSaldo;

**this**.valor = valor;

}

**public** String getNumeroDaContaSaldo() {

**return** numeroDaContaSaldo;

}

**public** **void** setNumeroDaContaSaldo(String numeroDaContaSaldo) {

**this**.numeroDaContaSaldo = numeroDaContaSaldo;

}

**public** **double** getValor() {

**return** valor;

}

**public** **void** setValor(**double** valor) {

**this**.valor = valor;

}

}

Teste.java

**package** APPBank;

**import** **static** org.junit.Assert.\*;

**import** java.time.LocalDate;

**import** org.junit.Test;

**import** APPBank.Modelo;

**public** **class** Teste {

**private** **static** **final** Object ***True*** = **null**;

@Test

**public** **void** test() {

Modelo modelo = **new** Modelo();

//Teste de cadastramento de Usuario

modelo.cadastrarUsuario(**new** Usuario("Jose\_Ricardo","12345","josericardo@email.com","K2033J", **new** Saldo("K2033J", 100.0), **new** Conta(25.00,123456,**false**, LocalDate.*of*(2017,11,20))));

modelo.cadastrarUsuario(**new** Usuario("Ana\_Brito","21548","anabrito@email.com","J8011K", **new** Saldo("J8011K", 100.0),**new** Conta(25.00,123456,**false**,LocalDate.*of*(2017,11,30))));

modelo.cadastrarUsuario(**new** Usuario("Jose\_Ricardo","12346","josericardo2017@email.com","K2034J", **new** Saldo("K2034J", 100.0),**new** Conta(25.00,123456,**true**,LocalDate.*of*(2017,12,01))));

modelo.cadastrarUsuario(**new** Usuario("Rodrigo Melo","54321","rodrigomelo@email.com","K2033J", **new** Saldo("K2033J", 500.0),**new** Conta(25.00,123456,**false**,LocalDate.*of*(2017,11,13))));

modelo.cadastrarUsuario(**new** Usuario("","","","",**new** Saldo("",0),**new** Conta(0.0,0,**false**,LocalDate.*of*(17,11,13))));

modelo.cadastrarUsuario(**new** Usuario("Cleber\_Jose","66666","cleberjose2017@email.com","C2034J", **new** Saldo("C2034J", 1000.0),**new** Conta(125.00,9874563,**false**,LocalDate.*of*(2017,11,13))));

modelo.logarUsuario("Jose\_Ricardo", "12345");

modelo.logarUsuario("Lucas\_M", "l123");

*assertEquals*(modelo.getUsuarios().size(),4);

*assertEquals*(modelo.getUsuarios().get(0).getUser(),"Jose\_Ricardo"); //Verifica a posição de um dado

//assertEquals(modelo.getUsuarios().get(0).getConta().getData(),LocalDate.of(11,11,13));

*assertEquals*(modelo.getUsuarios().get(0).getNumeroDaConta(),"K2033J");

*assertEquals*(modelo.getUsuarios().get(0).getSenha(),"12345");

*assertEquals*(modelo.buscarUsuarioPorUser("Jose\_Ricardo").size(),2);

*assertEquals*(modelo.logarUsuario("Jose\_Ricardo", "12345"),**true**);

//assertEquals(modelo.logarUsuario("Lucas\_M", "l123"),true); //Nao loga pois nao rexiste na lista de cadastro

//assertEquals(modelo.logarUsuario("Jose\_Ricardo", "12345"),false);

*assertEquals*(modelo.buscarSaldo("K2033J"), **new** Double(100.0)); //Retorna o valor contido no Saldo

//assertEquals(modelo.buscarSaldo("K2033J"), new Double(-1));

//assertEquals(modelo.getUsuarios().get(3).getUser(),"Rodrigo Melo"); //Nao registra pois o numero da conta é igual a do José ricardo da posição 0

//assertEquals(modelo.buscarUsuarioPorUser("Rodrigo Melo").size(),1);

*assertEquals*(modelo.getUsuarios().get(1).getUser(),"Ana\_Brito");

*assertEquals*(modelo.getUsuarios().get(1).getNumeroDaConta(),"J8011K");

*assertEquals*(modelo.getUsuarios().get(1).getSenha(),"21548");

*assertEquals*(modelo.buscarUsuarioPorUser("Ana\_Brito").size(),1);

*assertEquals*(modelo.getUsuarios().get(1).getConta().getData(),LocalDate.*of*(2017,11,30));

*assertEquals*(modelo.getUsuarios().get(3).getUser(),"Cleber\_Jose");

*assertEquals*(modelo.getUsuarios().get(3).getNumeroDaConta(),"C2034J");

*assertEquals*(modelo.getUsuarios().get(3).getSenha(),"66666");

*assertEquals*(modelo.buscarUsuarioPorUser("Cleber\_Jose").size(),1);

*assertEquals*(modelo.getUsuarios().get(3).getConta().getData(),LocalDate.*of*(2017,11,13));

//Teste de cadastramento de Banco

modelo.cadastrarBanco(**new** Banco(0001,"Bradesco","bancoBradesco@email.com"));

modelo.cadastrarBanco(**new** Banco(0002,"Itau","bancoItau@email.com"));

modelo.cadastrarBanco(**new** Banco(0003,"Santander","bancoSantander@email.com"));

modelo.cadastrarBanco(**new** Banco(0004,"Safra","bancoSafra@email.com"));

modelo.cadastrarBanco(**new** Banco(0005,"Caixa","bancoCaixa@email.com"));

modelo.cadastrarBanco(**new** Banco(0001,"Brasil","bancoBradesco@email.com"));//Nao registra(Campo repetido)

modelo.cadastrarBanco(**new** Banco(0002,"Safra","bancoSafra@email.com"));//Nao registra(Campo repetido)

modelo.cadastrarBanco(**new** Banco(0,"",""));//Nao registra(campo vazio)

*assertEquals*(modelo.getBancos().size(),5);

*assertEquals*(modelo.getBancos().get(0).getNomeBanco(),"Bradesco");

*assertEquals*(modelo.getBancos().get(0).getId(),0001);

*assertEquals*(modelo.getBancos().get(0).getEmail(),"bancoBradesco@email.com");

*assertEquals*(modelo.buscarBancoPorNome("Bradesco").size(),1);

*assertEquals*(modelo.getBancos().get(1).getNomeBanco(),"Itau");

*assertEquals*(modelo.getBancos().get(1).getId(),0002);

*assertEquals*(modelo.getBancos().get(1).getEmail(),"bancoItau@email.com");

*assertEquals*(modelo.buscarBancoPorNome("Itau").size(),1);

//assertEquals(modelo.buscarBancoPorNome("Brasil").size(),2);

//Teste de casdastramento Admin

modelo.cadastrarAdmin(**new** Admin("Antonio\_Almeida","senhaAntonio123"));

modelo.cadastrarAdmin(**new** Admin("Antonio\_Almeida","senhaAntonio123"));//Nao registra(Campo repetido)

modelo.cadastrarAdmin(**new** Admin("Cleber\_Augusto","senhaAugusto123"));

*assertEquals*(modelo.getAdmin().size(),2);

//Uso de função isLate(Retorna valor boolean, comparando a data do pagamento se está atrasado)

//assertEquals(modelo.isLate(modelo.getUsuarios().get(0).getConta().getData()),true);

*assertEquals*(modelo.isLate(LocalDate.*of*(2017,11,20)),**false**);

*assertEquals*(modelo.verificaContAtr(modelo.getUsuarios().get(1).getUser(), modelo.getUsuarios().get(1).getNumeroDaConta(), modelo.getUsuarios().get(1).getConta().getData()),**false**);

*assertEquals*(modelo.verificaContAtr(modelo.getUsuarios().get(2).getUser(), modelo.getUsuarios().get(2).getNumeroDaConta(), modelo.getUsuarios().get(2).getConta().getData()),**false**);

*assertEquals*(modelo.verificaContAtr(modelo.getUsuarios().get(3).getUser(), modelo.getUsuarios().get(3).getNumeroDaConta(), modelo.getUsuarios().get(3).getConta().getData()),**true**);

//Pagar conta

*assertEquals*(modelo.pagarConta(modelo.getUsuarios().get(0).getConta().getCodigoBarra(), modelo.getUsuarios().get(0).getConta().getValorFatura(),modelo.getUsuarios().get(0).getConta().getPago()),**true**);

//assertEquals(modelo.pagarConta(modelo.getUsuarios().get(1).getConta().getCodigoBarra(), modelo.getUsuarios().get(1).getConta().getValorFatura(),modelo.getUsuarios().get(1).getConta().getPago()),false);

*assertEquals*(modelo.pagarConta(modelo.getUsuarios().get(2).getConta().getCodigoBarra(), modelo.getUsuarios().get(2).getConta().getValorFatura(),modelo.getUsuarios().get(2).getConta().getPago()),**false**);

}

}