# Exercício de Fixação Teste de Software usando Black Box Software

O exercício consiste na criação de casos de teste da aplicação “Aposta”. O código apresenta alguns defeitos intencionais e você tem que encontrar criando um conjunto de casos de teste. Deve ser criado **no mínimo dez casos** de testes.

**Requisitos funcionais**

1) O sistema deve permitir o cadastro de um usuário informando CPF , e-mail e password.
    a) Um CPF é válido quando tem 11 caracteres numéricos, caso contrário deve emitir uma excepção indicando “CPF invalido”
    b) Um e mail é válido quando tem pelo menos um caractere seguido por “@”
    c) Uma password é válida quando pelo menos tem um carácter especial (!?.*, etc)

2) O sistema deve inserir 200 moedas na carteira do usuário após o cadastro do usuário.
3) Uma partida é definida pelo nome do campeonato, nome do time visitante e nome do time mandante. Quando a partida é criada, as apostas estão bloqueadas.
4) Uma partida somente pode receber apostas depois de esta ser liberada.
5) Uma aposta somente pode ser feita por um usuário cadastrado e com moedas suficientes para pagar o custo da aposta. O apostador deve definir um número de gols tanto para o time visitante como para o mandante.
6) O custo da aposta é de 50 moedas.
7) O sistema deve permitir uma aposta quando:
    a) A partida estiver liberada para receber apostas;
    b) O usuário tem moedas suficiente para apostar;
    c) A quantidade de gols visitantes e mandantes deve ser maior ou igual que 0;

**Classes e métodos**

| **Classe** | **Método** | **Descrição** |
|---|---|---|
| Usuário | getEmail() | Obtém o e-mail do usuário. |
| | setEmail(String email) | Define um e-mail para o usuário. |
| | getCpf() | Obtém o CPF do usuário. |
| | setCpf(String cpf) | Define o CPF do usuário. |
| | getPassword() | Obtém a senha do usuário. |
| | setPassword(String password) | Define a senha do usuário. |
| | temSaldoSuficiente() | Retorna true se o usuário tem saldo suficiente para realizar uma aposta. False se contrário. |

| | | |
|---|---|---|
| | getMoedas() | Retorna a quantidade de moedas que o usuário tem hoje em carteira. |
| | diminuirMoedas() | Diminui a quantidade de moedas referente ao custo da aposta realizada. |
| Partida | getCampeonato() | Obtém o nome do campeonato. |
| | setCampeonato(String campeonato) | Define o nome do campeonato. |
| | getTimeVisitante() | Obtém o nome do time visitante. |
| | setTimeVisitante(String timeVisitante) | Define o nome do time visitante. |
| | getTimeMandante() | Obtém o nome do time mandante. |
| | setTimeMandante(String timeMandante) | Define o nome do time mandante. |
| | getStatus() | Obtém o status da partida. Valores: *aposta_bloqueada* ou *apostas_abertas*. |
| | liberarApostas() | Libera apostas. |
| | estaDisponivelReceberApostas() | Retorna se uma partida está liberada para receber apostas. |
| | getNumeroApostas() | Obtém o número de apostas realizadas. |
| | enviarAposta(Aposta aposta) | Recebe uma nova aposta. |
| Aposta | getGolsVisitante() | Obtém quantidade de gols visitantes. |
| | setGolsVisitante(int golsVisitante) | Define quantidade de gols visitantes. |
| | getGolsMandate() | Obtém quantidade de gols mandante. |
| | setGolsMandate(int golsMandate) | Define quantidade de gols mandante. |
| | getApostador() | Obtém o objeto apostador que realizou a aposta. |
| | setApostador(Usuario apostador) | Define o objeto apostador que realizou a aposta. |
| | getPartida() | Obtém o objeto da partida. |
| | setPartida(Partida partida) | Define o objeto da partida. |
| | enviar() | Enviar a proposta se a partida está disponível para receber apostas e se o usuário tem saldo suficiente. Retorna *True* se sucesso. |