Grupo:

Gabriel de Pádua RA 200749
Gabriela Fontolan RA 200645
Náthali Yukari Obara Linares RA 200145
Rafael Henrique Penha Prestes RA 200779
Thomas Elinton de Oliveira RA 130531

Projeto

Para quem gosta de jogos multiusuários os5 jogos de Adivinhação e Enigma tem o objetivo de entreter, "rankiar" os vencedores para trazer diversão em bares e restaurantes para vender produtos (comida e bebidas).

Product Backlog

User story

Prioridade	Categoria	Como usuário	Eu quero poder	De modo que
		Admin	avaliar os jogos	mantenha a veracidade das informações e o jogo válido.

Sprint Backlog

BDDs

Canários	Dado	E	Quando	E	Então	E
Avaliando um resultado valido para o jogo	um resultado valido		o jogo está acontecendo	o resultado é anotado	o resultado deve ser aceito	
Avaliando um resultado invalido para o jogo	um resultado invalido de <valor> e do Particicpante <nomeParticipante>		o jogo está acontecendo	o resultado é anotado	o resultado não deve ser aceito	
Avaliação de vários resultados para o Jogo	um Jogo Acontecendo		existe dois resultados	de participates diferentes	O jogo é considerado valido	
Avaliando resultados de um participante	um Jogo Acontecendo		um Participante	anotar dois ou mais resultados deste participante	o segundo resultado deve ser ignorado	

Implementação

TDDs

Cenário	Execução	Resultado
@Test public void avaliandoUmResultadoValidoPar aJogo(){ Jogo jogo = new CriadorDeJogo() .para("Poker") .resultado(new Participante("Gabriel"), 150.0) .constroi();		assertEquals(1, jogo.getResultados() .size()); assertEquals(150.0, jogo.getResultados() .get(0).getMetrica(), 0.00001);

@Test(expected=Exception.class)
public void
avaliandoUmResultadoInvalidoP
araJogo() throws Exception {
Jogo jogo = new CriadorDeJogo()
.para("Poker")
.resultado(new
Participante("Gabriel"), -1)
.constroi();
}		

@Test
public void
deveTerVariosResultados() {
Jogo jogo = new CriadorDeJogo()
.para("Cata moedas")
.resultado(new
Participante("Nelson"), 150.0)
.resultado(new
Participante("Pedro"), 200.0)
.constroi();
		assertEquals(2,
		jogo.getResultados().size());
		assertEquals(150.0,
		jogo.getResultados().get(0).
		getMetrica(), 0.00001);
		assertEquals(200.0,
		jogo.getResultados().get(1).
		getMetrica(), 0.00001);

@Test public void naoDeveAceitarDoisResultadosD oMesmoParticipante(){ Jogo jogo = new Jogo("Ca&#xfffd;a pe&#xfffd;as"); Participante leonardo = new Participante("Leonardo");	jogo.anota(new Resultado(leonardo, 500.0)); //deve ser ignorado jogo.anota(new Resultado(leonardo, 600.0));	assertEquals(1, jogo.getResultados() .size()); assertEquals(500, jogo.getResultados() .get(0).getMetrica(), 0.00001);