UNIVERSIDADE FEDERAL DO ABC



**RELATÓRIO DO PROJETO DA DISCIPLINA PARADIGMAS DE PROGRAMAÇÃO**

André Barbosa 11001814

Lucas Oliveira 11056816  
 Iasmin Pracchias 11018111

Santo André - 2018

[**1. Introdução**](#_u9f9amq7uh9) **2**

[**2. Definição do Problema**](#_bghbs0f64vcs) **2**

[**3. Arquitetura e Implementação**](#_7mmmo5qrs2e4) **2**

[**4. Avaliação**](#_qi4xc8a81l1p) **3**

[4.1 Testes realizados, com ou sem usuários, testes de execução e screenshots](#_lqr3aunsyy1r) 3

[**5. Discussão**](#_sx576j8r5d82) **7**

[**6. Detalhamento das contribuições de cada integrante para o projeto**](#_px2ladi6trld) **7**

[**7. Link para o repositório do projeto**](#_azg6k3lr603o) **7**

# 

# 1. Introdução

Este relatório tem por objetivo apresentar os resultados obtidos ao desenvolver um jogo de Blackjack utilizando a linguagem funcional Racket. É possível encontrar o jogo implementado com diversas linguagens imperativas de programação, mas poucas vezes implementado com uma linguagem funcional, logo utilizando os conhecimentos adquiridos na disciplina Paradigmas de Programação resolveu-se verificar quais seriam as diferenças, vantagens e desvantagens de implementar o jogo utilizando uma linguagem funcional.

O relatório apresenta informações sobre sobre a adaptação do Blackjack feita neste projeto, incluindo a arquitetura e implementação do jogo, a avaliação feita pela equipe do projeto, e ainda uma discussão sobre os principais pontos identificados.

# 2. Definição do Problema

Neste projeto foi proposta a construção de um jogo de **Blackjack,** também conhecido como **vinte e um**. A proposta foi a de utilizar a linguagem funcional Racket para a construção do código, e uma arquitetura cliente/servidor para implementação da dinâmica do jogo.

Para simplificação esta versão utiliza um baralho normal com 52 cartas, e a cada turno um jogador compete contra um dealer. No início do jogo tanto o jogador quanto o dealer começam com 2 cartas na mão, o jogador verifica os valores na sua mão e decide se deseja **comprar mais uma carta** ou **parar**. Se o jogador compra uma carta o valor da mão é reavaliado, caso o valor da nova mão ultrapasse 21, o jogador perde, caso contrário ele tem a opção de continuar comprando ou parar. Se o jogador para é a vez do dealer efetuar a compra, nesta versão o dealer continua comprando cartas enquanto sua mão tiver um valor menor ou igual a 17. O jogo finaliza quando o dealer e o jogador param ou quando o valor da mão de um dos dois ultrapassa 21 pontos, no caso em que os dois decidem parar antes de o valor da mão ultrapassar 21, o valor que chegar mais próximo de 21 vence.

# 3. Arquitetura e Implementação

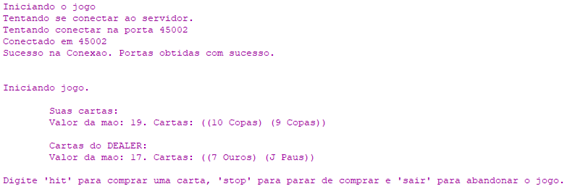
O jogo foi construído utilizando-se uma arquitetura cliente/servidor, onde o jogador interage com o servidor (*servidor-21.rkt*) por meio do cliente (*cliente-21.rkt*), e o servidor armazena uma IA que representa o dealer. A conexão de portas cliente/servidor foi estabelecida utilizando a biblioteca TCP oferecida pela própria linguagem (*conexao-lib.rkt*).   
 A lógica do jogo foi dividida em dois arquivos onde um trata as trocas de mensagens entre cliente e servidor durante o decorrer do jogo (*blackjack-msgs.rkt*) e outro aplica as regras do jogo analisando os tipos e valores de cartas nas mãos do jogador e do dealer (*blackjack-rules.rkt*). O arquivo *baralho.rkt* fornece um baralho de 52 cartas embaralhadas.

# 4. Avaliação

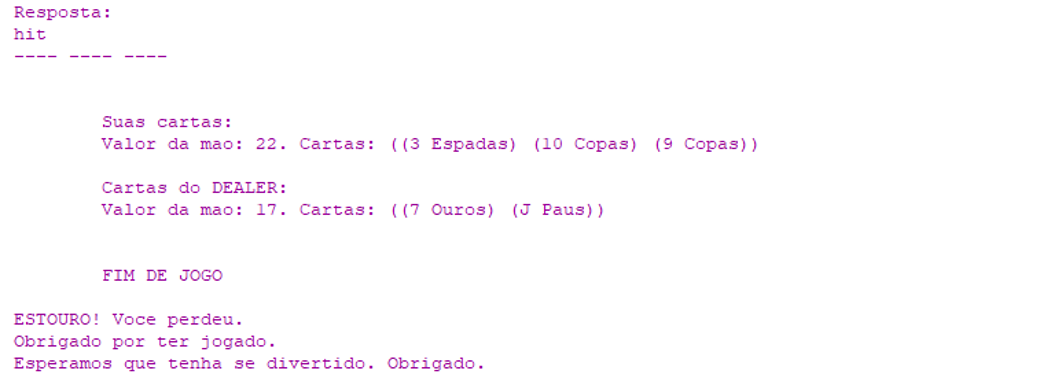
Os testes foram feitos pelos próprios integrantes do grupo utilizando tanto o terminal quanto a ferramenta DrRacket oferecida pelos desenvolvedores da linguagem.

## 4.1 Testes realizados, com ou sem usuários, testes de execução e screenshots

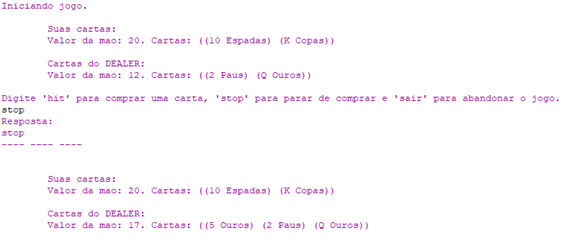
* Com o servidor rodando, ao ser iniciado o cliente mostra na tela informação sobre as cartas em cada mão enviadas pelo servidor, e fornece opção de escolha ao jogador



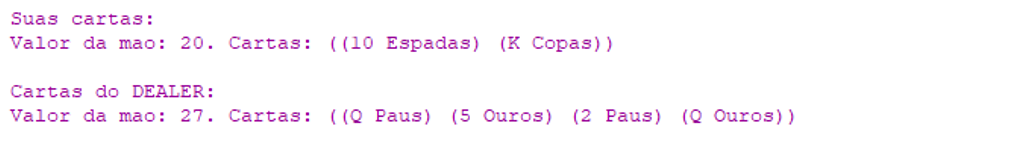
* Caso 1: o jogador escolhe a opção ‘hit’ para comprar uma carta e o valor na mão ultrapassa 21, o jogador perde e o dealer ganha.

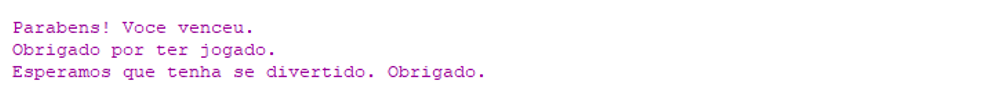
.

* Caso 2: o jogador escolhe a opção ‘stop’, aguarda o dealer, o dealer compra e ultrapassa o valor de 21, e o jogador ganha.

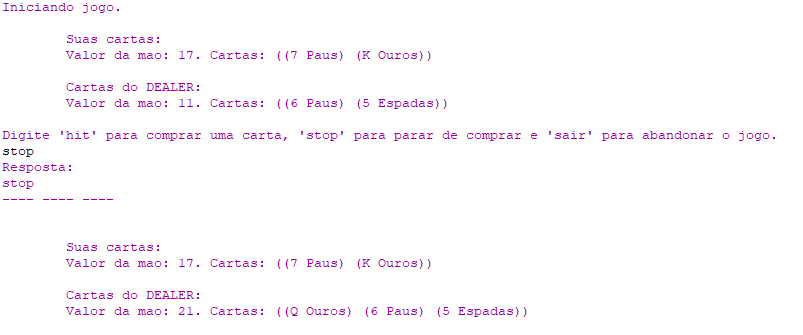








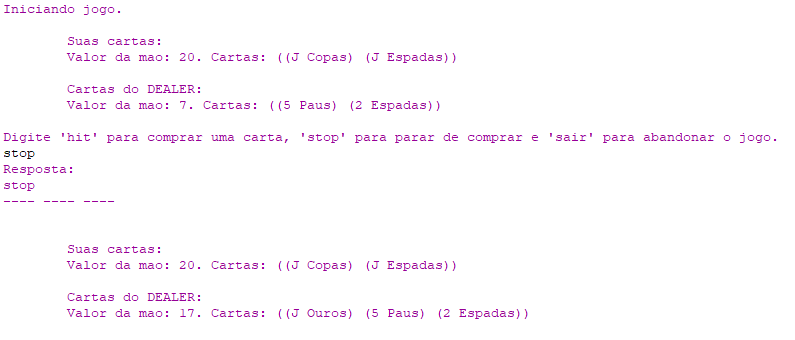
* Caso 3: o dealer ganha, pois chega mais próximo de 21 do que o jogador.



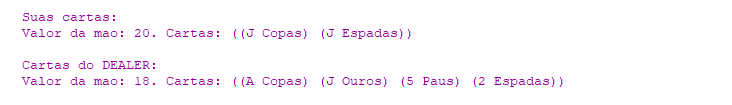


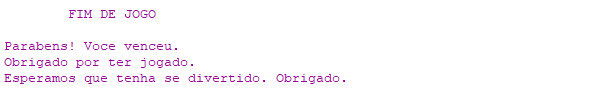


* Caso 4: jogador ganha, pois chega mais próximo de 21 do que o dealer.

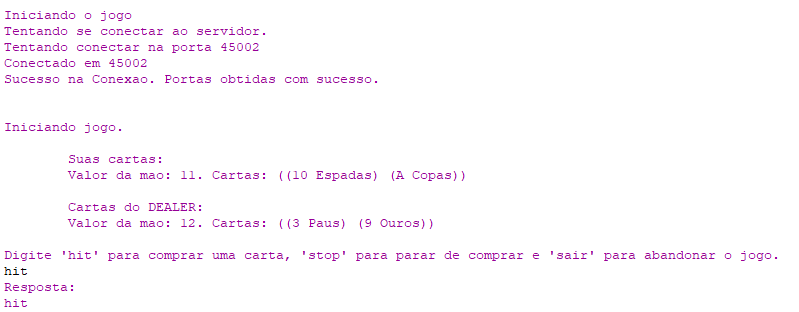


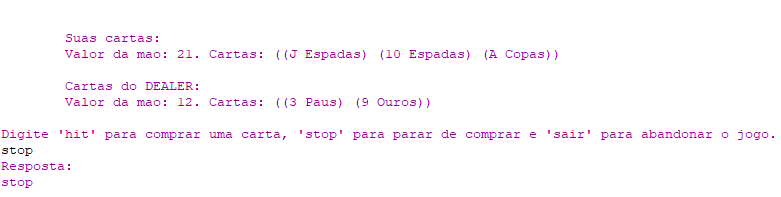


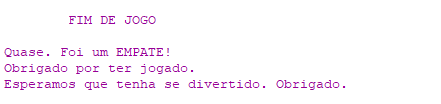




* Caso 5: Empate.







# 5. Discussão

As maiores dificuldades encontradas envolvem a mudança de paradigma de linguagem, como os integrantes não haviam utilizado uma linguagem funcional anteriormente foi necessário um maior esforço para encontrar as estruturas e bibliotecas necessárias para desenvolver o projeto. Houve também uma mudança de tema do projeto durante o quadrimestre por conta do tempo hábil disponível para finalização, assim foi necessário refazer algumas partes do projeto que já estavam encaminhadas.

# 6. Detalhamento das contribuições de cada integrante para o projeto

Todos fizemos contribuições individuais no código e nas entregas parciais do projeto. Fizemos também reuniões online onde uma pessoa compartilhava a tela pela ferramenta *hangouts* e todos os integrantes conseguiam auxiliar com a lógica e opinar.

# 7. Link para o repositório do projeto

https://github.com/VFS/pp201802/tree/master/src/temp