



Universidade Estadual Vale do Acaraú - UVA
Curso: Ciência da Computação
Disciplina: Introdução a Ciências da Computação
Professor: Eder Jacques Porfirio Farias

ICC - AP3 WebPong(BETA v0.1.0)

Aluno(a): Ilder Rafael Damasceno Ponte **Programador**
Aluno(a): Rafael Cordeiro Araujo Lima **Revisor de Código**
Aluno(a): Luis Fernando Paiva de Souza **Testar o jogo e reportar bugs/Apresentação**
Aluno(a): Caio Monte Mendes **Testar o jogo e reportar bugs/Apresentação**

Resumo

O trabalho se baseia na criação de uma versão online multiplayer do jogo Pong, onde dois jogadores em computadores distintos jogam em uma mesma sala. Para criar o jogo se usarão as tecnologias: **ReactJS**, **NodeJS**, **WebSocket**. Sendo a aplicação dividida entre "server" e "front". Os dados serão processados no "server" e o front será responsável por mostrar o jogo na tela. Ademais o jogo deve conter um sistema de chat e salas.

1. Introdução

O programa utiliza o **ReactJS** para conseguir manusear componentes e valores de uma melhor forma de início vamos explicar a função básica de cada script do projeto, sendo que cada um desses é um componente.

- **index.js:** Este é o responsável por ligar a aplicação a página **index.js** presente no diretório `./public`, o que torna a aplicação visível a todos.
- **app.js:** Este é responsável pela união dos componentes **pong.js** e **styleGlobal.js**.
- **styleGlobal.js:** Este é responsável pelo estilo da aplicação como um todo. Para gerenciar o estilo é utilizada a "biblioteca" **styled-components**. [Ele também passa algumas propriedades ("props") para o **pong.js**]
- **pong.js:** Este é responsável por juntar e organizar todos os componentes para o funcionamento da aplicação. Nele também são definidas as propriedades ("props") utilizadas nos componentes, elas são obtidas com o **useContext** que importa o "contexto" do script **GameContext.js**.
- **roomList.js:** Este é responsável por listar as salas disponíveis e os botões relacionados a salas.
- **playerList.js:** Este é responsável por listar os *players* na tela.
- **chat.js:** Este é responsável por gerenciar o chat e mensagens.
- **game.js** Este é responsável por desenhar o jogo na tela, para isso usa a "biblioteca" **react-svg-draw**.
- **gameContext.js:** Este é responsável por gerenciar a comunicação com o **server.js** utilizando o **Socket**.
- **server.js:** Este é responsável por gerenciar as informações gerais da aplicação, como as salas, os players, e o jogo.

2. Execução

Para executar basta seguir os seguintes passos:

- Abra dois terminais(cmd, powershell, bash, etc.), um na pasta `./gameserver` e outro na pasta `./gamefront`
- Depois digite os seguintes comandos

```
npm install  
npm start
```

- Depois basta abrir no endereço `http://localhost:3000`[Na versão final o jogo funcionará em LAN, mas nesse caso basta apenas abrir em duas guias diferentes]

3. Problemas Conhecidos

- **Demora na desconacção:** Quando a pessoa recarrega a pagina varias vezes pode acontecer de o "player" anterior que era a propria pessoa não se disconnect, isso acontece devido a um atraso por "afogamento"(varias chamadas feitas de uma só vez) no **Socket** devido a varias requisições, mas não afeta em nada o desenpenho da aplicação.
- **Falta das barras:** As barras ainda não foram adicionadas no jogo devido a necessidade de se ter que fazer pesquisas para ver como funcionará a colisão, mas nada muito difícil.
- **"Match is null":** Esse erro pode aparecer devido a um bug envolvendo "afogamento"no **Socket**, mas é raro.

"FLUXOGRAMA"e IMAGENS:



