

Relatório de Progresso

Draw Wars

Projeto e Seminário

2018/19

LI61N - Grupo 45

**Paulo Pereira**

Rodrigo Cabral | 40661

Manuel Briceño | 39318

Luís Vieira | 39317

# Resumo



Contents

[Resumo 2](#_Toc9079042)

[Introdução 4](#_Toc9079043)

[O Jogo 4](#_Toc9079044)

[Requisitos 4](#_Toc9079045)

[Preparação 4](#_Toc9079046)

[Fase 1 – Desenhar 4](#_Toc9079047)

[Fase 2 – Adivinhar 5](#_Toc9079048)

[Glória 5](#_Toc9079049)

[Formulação do Problema 6](#_Toc9079050)

[Solução Proposta 7](#_Toc9079051)

[Arquitetura 7](#_Toc9079052)

[Servidor ASP.NET Core 7](#_Toc9079053)

[Aplicação Android 7](#_Toc9079054)

[Aplicação Web 7](#_Toc9079055)

[Implementação 7](#_Toc9079056)

[Servidor ASP.NET Core 7](#_Toc9079057)

[Android 8](#_Toc9079058)

[Aplicação Web 9](#_Toc9079059)

[Planeamento 10](#_Toc9079060)

[Até ao momento 10](#_Toc9079061)

[27 maio, Cartaz e Versão Beta 10](#_Toc9079062)

[Até 16 Julho 10](#_Toc9079063)

# Introdução

*DrawWars, um jogo sobre desenhos e palpites, é uma reimplementação do jogo clássico Pictionary, onde os jogadores devem fazer uso dos seus dispositivos Android para participar – todos os desenhos, assim como os vários palpites, são feitos através de uma aplicação android.*

*Este projecto é, em grande parte, um estudo de jogos já existentes no mercado, que permitem a utilização de dispositivos móveis como comandos – um grande exemplo é a gama PlayLink, da Sony, que utiliza um sistema Playstation 4 para apresentar o estado do jogo, sendo os smartphones dos jogadores utilizados como meio de interacção.*

*Para fins de implementação do projecto, a arquitectura escolhida é composta por três componentes – uma aplicação web, um web service, e uma aplicação Android. A aplicação web é uma Single Page Application desenvolvida com a framework Vue.js, o web service foi criado em ASP.Net Core, e a aplicação Android trata-se de uma aplicação nativa, desenvolvida com Kotlin.*

*Dada a necessidade de coordenar, através do servidor, os vários componentes responsáveis pelo jogo, isto é, os vários dispositivos android e a aplicação web, decidimos utilizar a biblioteca SignalR, que permite a criação de grupos, assim como o envio de mensagens para todos os elementos dos ditos grupos, em simultâneo.*

## O Jogo

### Requisitos

Para jogar este jogo, é necessário um web browser capaz de correr a aplicação web, e que todos os jogadores tenham um dispositivo Android, com a aplicação instalada. A aplicação web tem, como objectivo, a apresentação do estado do jogo – tanto os desenhos, como a tabela de resultados, são apresentados apenas na aplicação web. A aplicação, por outro lado, serve apenas para introdução de dados, isto é, é na aplicação que são feitos todos os desenhos, assim como todas as tentativas de adivinha.

### Preparação

Para começar o jogo, deve-se, através da aplicação web, carregar no botão “*Create Room*”. Após carregar no botão, será apresentado o código da sala recém-criada – o *Room Code*. Este room code deverá ser introduzido na aplicação android, para estabelecer uma ligação entre o jogador e o servidor. Depois de introduzido o código, é pedido ao utilizador que definam um nome de utilizador, username, para fácil identificação durante o jogo. Quando todos os jogadores tiverem estabelecido a ligação com o servidor, deve ser premido o botão “Start”.

### Fase 1 – Desenhar

Nesta primeira fase do jogo, é enviado para cada um dos jogadores, através do seu dispositivo android, um tema aleatório que o jogador deverá desenhar, nunca revelando o tema aos demais jogadores. Esta fase tem um tempo limite no qual os desenhos deverão ser submetidos. No caso dos utilizadores não finalizarem o seu desenho dentro do tempo permitido, será submetido o desenho no estado em que se encontrar.

### Fase 2 – Adivinhar

Durante a fase de adivinha, serão apresentados na aplicação web, à vez, os desenhos dos jogadores. Cada um dos desenhos será apresentado durante um tempo limitado. Enquanto um desenho está visível, os jogadores (exluindo o artista do desenho apresentado) deverão tentar adivinhar qual o tema que inspirou o desenho, devendo inserir os seus palpites no respectivo dispositivo android. Quando um jogador adivinha o tema, são-lhe concedidos pontos, e não pode inserir mais palpites. Quando todos os jogadores adivinham o tema, ou termina o tempo, é apresentado, na aplicação web, a tabela de pontuação actualizada.

### Glória

Acabada a fase 2, isto é, terem sido apresentados todos os desenhos, é apresentada, uma última vez, a tabela de classificação. Aqui, é escolhido o vencedor do jogo, sendo ele o jogador que acumulou o maior número de pontos.

# Formulação do Problema

*DrawWars*, como explicado acima, consiste num videojogo onde os jogadores terão como objetivo desenhar temas secretos que lhes serão apresentados, e adivinhar qual o tema que inspirou os desenhos dos outros jogadores.

Posto isto, existem várias componentes a ter em conta, tal como a atribuição de sessão de jogo aos utilizadores, a comunicação bidirecional entre estes e o jogo, a apresentação dos seus desenhos, a avaliação das suas sugestões, e a apresentação de resultados.

Como tal, o projeto está dividido em três componentes - Servidor, Front-End Web e aplicação Android **-** dos quais as especificidades de implementação serão detalhadas nos seguintes capítulos.

# Solução Proposta

## Arquitetura

### Servidor ASP.NET Core

O servidor ébaseado na biblioteca *SignalR*. O *SignalR* fornece componentes para comunicação bidireccional entre os clientes e o servidor, através de notificações em tempo real, utilizando w*eb sockets*. O servidor é responsável por manter o estado do jogo, as conexões com os vários jogadores e a aplicação web, notificar todos os intervenientes de alterações no estado do jogo (acabou o tempo, comecem a desenhar, etc...), e gerir a informação enviada pelas várias aplicações, avançando assim o estado do jogo.

### Aplicação Android

A aplicação *Android*, escrita em *Kotlin*, tira partido da biblioteca *SignalR*para android. Não sendo a aplicação mais que um comando que permite a interacção dos jogadores com o jogo, a maioria das suas *Activities* não dispõe de estado, servido apenas para comunicar com o servidor. A fins de manter a ligação com o servidor aberta, foi implementado um *Bound Service*, que acompanha todo o ciclo de vida da aplicação, sendo todas as actividades *bound* ao dito serviço.

### Aplicação Web

A aplicação Web,implementada com VueJs, utiliza Nodepara gerir os pacotes e, através de SignalR, estabelece a comunicação com o servidor. A aplicação é uma SPA e, sucintamente, um display de informação. O servidor é utilizado maioritariamente para consumo de dados, e à medida que estes são recebidos, o estado, neste caso as páginas, vão alternando.

## Implementação

### Servidor ASP.NET Core

Como peça central, foi implementado um Servidor em ASP.NET Core. A responsabilidade deste componente é a de gerir o jogo, assim como os jogadores.

Em termos de comunicação, e tal com foi mencionado anteriormente, foi utilizada a biblioteca *SignalR*, que permite estabelecer uma ligação com os clientes (*Web* e *Android*) via *web sockets*. Para fazer uso da mencionada biblioteca, foi criada uma classe que estende **Microsoft.AspNetCore.SignalR.Hub**. Esta classe disponibiliza a interface através da qual os clientes podem fazer trocas de dados.

Existe, ainda, uma estrutura de classes que representam a lógica de jogo. Destas, a mais importantes é a **GameSession**, que representa a sessão de jogo. A **GameSession** contem uma lista de **Player**, classe esta que representa um jogador, contendo dados como o *username*, o identificador, e os pontos. Existe ainda uma classe estática que trata de manter as várias sessões de jogo – **CoreManager**.

### Android

A aplicação android serve como comando – nada mais que um meio de interação entre o jogador e o jogo. Tendo todas as interações necessidade de serem comunicadas ao servidor, foi, em conformidade com o mesmo, feito uso da biblioteca *SignalR* para android.

Ao abrir a **MainActivity**, é iniciado o *bound service* **ServerService**, que está encarregue da comunicação com o servidor. O **ServerService**, para iniciar a comunicação com o servidor, faz um *handshake* com o mesmo, enviando um pedido HTTP. A **MainActivity**, por sua vez, fica *bound* ao **ServerService**.

A **MainActivity**, para além de iniciar o **ServerService**, tem a responsabilidade de permitir a inserção do *room code*, deixando o jogador aceder a uma sessão de jogo. Quando o jogador insere um *room code*, é iniciada a **SetupActivity**, que permite ao utilizador a escolha de um nome de utilizador.

De seguida a aplicação progride para a **GameActivity**, nesta activity é apresentada uma mensagem que alerta o jogador que ainda não se juntaram todos os jogadores, e como tal, este deve esperar, ou avisar (através de um botão) que todos os jogadores estão prontos. Quando isto acontece, o serviço é notificado, contendo como parâmetro o tema a desenhar, pelo jogador.

É apresentado uma tela onde deve ser desenhado o tema. Assim que o desenho esteja completo, cada jogador deve submeter o seu desenho para o servidor, clicando no respetivo botão.

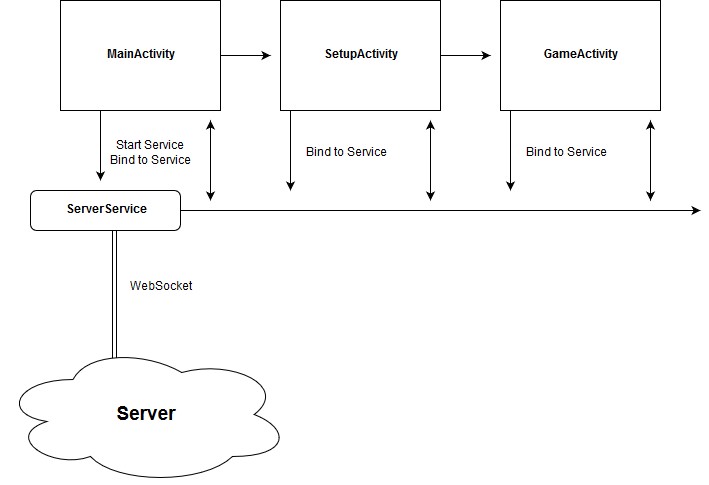


Fig1

### Aplicação Web

### Arquitectura

A aplicação web representa o estado de jogo partilhado. As interações que a aplicação android despoleta para o servidor são notificadas à Single Page Aplication (SPA), e reflectidas no browser. Para tal foi incluída a biblioteca @aspnet/signalr no projeto de forma a manter a comunicação websocket com o servidor.

A SPA é responsável por apresentar em tempo real as alterações de estado, despoletadas através das interações dos jogadores. É também a SPA que temporiza algumas fases de jogo, ou seja, **contém timers que informam o utilizador do tempo restante para termino da ação corrente e quando o tempo expira notifica o servidor do sucedido de forma a transitar de estado.**

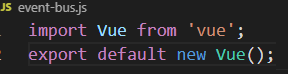
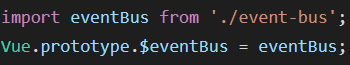
Esta SPA assenta sobre a framework js VueJs. Vue Js tem um sistema que permite renderizar, de uma forma declarativa, os dados na DOM através de uma sintaxe muito simples.



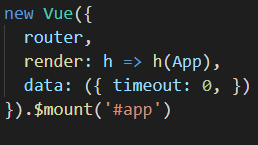
O que acontece aqui é que uma nova instância do Vue é criada, e esta por sua vez, é associada à div com o id “app”. A partir daqui todos os componentes de Vue criados, que representam as páginas da aplicação contendo cada um o seu html, serão injetados dentro desta div. Estes mesmos componentes têm acesso à sua instância, denominada por $root, e poderão aceder a possíveis variáveis globais criadas, neste exemplo “message” é transversal a todos os componentes e acessível através de $root.message.

Tirando partido de esta globalização, e de forma a simplificar o processo de temporização, foi criado um event bus e a variável global timeout.

O event bus é uma outra instância do VueJs, associada à instância principal, que apenas emite o evento “timer-finished” e recebe o evento “clear-timer”. É através destes eventos que os vários componentes da aplicação avançam de estado conforme o tempo de jogo tenha ou não terminado.

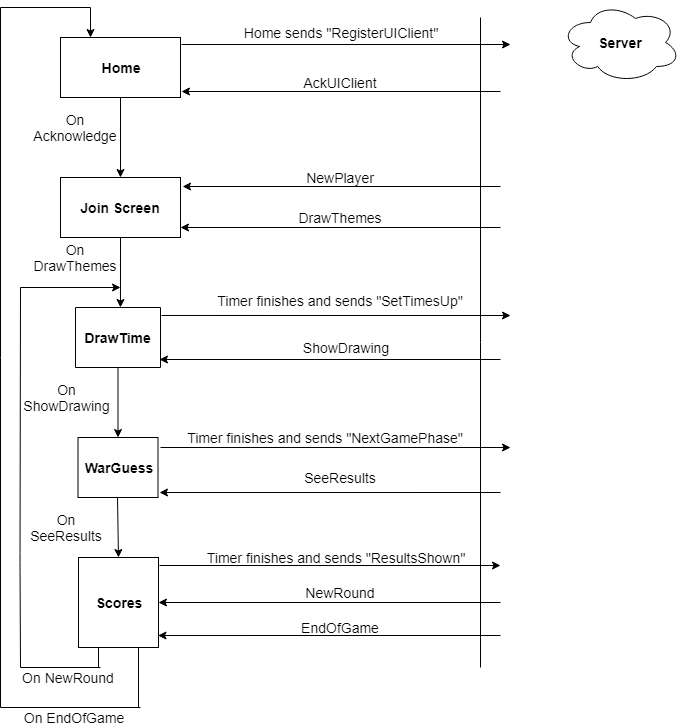
 

Visto que o valor de tempo é enviado nas várias notificações recebidas pelos componentes, foi necessário criar a variável global timeout de forma a ser afetada em diferentes partes da SPA.



De seguida, na Figura 1 e Tabela 1, serão apresentados um diagrama temporal sobre o fluxo da SPA e os vários eventos despoletados tanto pelo servidor como pela Aplicação Web de forma a manter comunicação bidirecional.

Fluxo Aplicacional



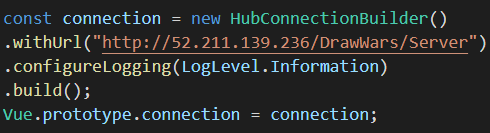
Figura

Tabela de eventos SignalR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Origem** | **Destino** | **Nome** | **Ecrã** | **Descrição** |
| Web App | Servidor | RegisterUiClient | Home | Registo do cliente WEB após o click no botão |
| Servidor | Web App | AckUiClient | Home | Confirmação do registo do cliente WEB |
| Servidor | Web App | NewPlayer | Join | Envio do novo jogador |
| Servidor | Web App | DrawThemes | Join | Notificação para avançar de ecrã |
| Web App | Servidor | SetTimesUp | DrawTime | Caso o tempo de desenho termine |
| Servidor | Web App | ReadyToShowDraws | DrawTime | Caso todos terminem de desenhar |
| Web App | Servidor | DrawPhaseLogic | DrawTime | Notifica o servidor que pode ser enviado o desenho |
| Servidor | Web App | ShowDrawing | DrawTime | Avança de ecrã mostrando o desenho |
| Servidor | Web App | PlayerGuess | WarGuess | Envio da guess do jogador |
| Web App | Servidor | NextGamePhase | WarGuess | Caso o tempo de adivinha termine |
| Servidor | WebApp | SeeResults | WarGuess | Avança de ecrã mostrando a pontuação |
| WebApp | Servidor | ResultsShown | Scores | Tempo de visualização de scores termina |
| Servidor | WebApp | NextRound | Scores | Caso ainda haja rondas de jogo |
| WebApp | Servidor | RoundEndedAck | Scores | Tempo de espera para a próxima ronda termina |
| Servidor | WebApp | DrawThemes | Scores | Reinicia o jogo indo para o ecrã DrawTime |
| Servidor | WebApp | EndOfGame | Scores | Não existem mais rondas voltando para Home |

Tabela

Após a inicialização interna do VueJs é criada a conexão com o servidor via instanciação de um Hub e o handshake é realizado. A conexão é mantida na instância do Vue de forma a todos os componentes terem acesso e conseguirem comunicar com o servidor.

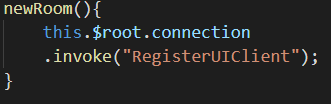
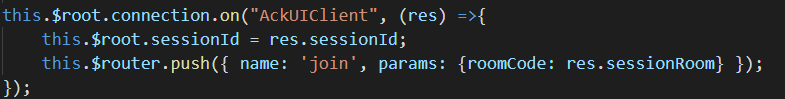


Com o sucesso do handshake é apresentado o ecrã principal, Home.

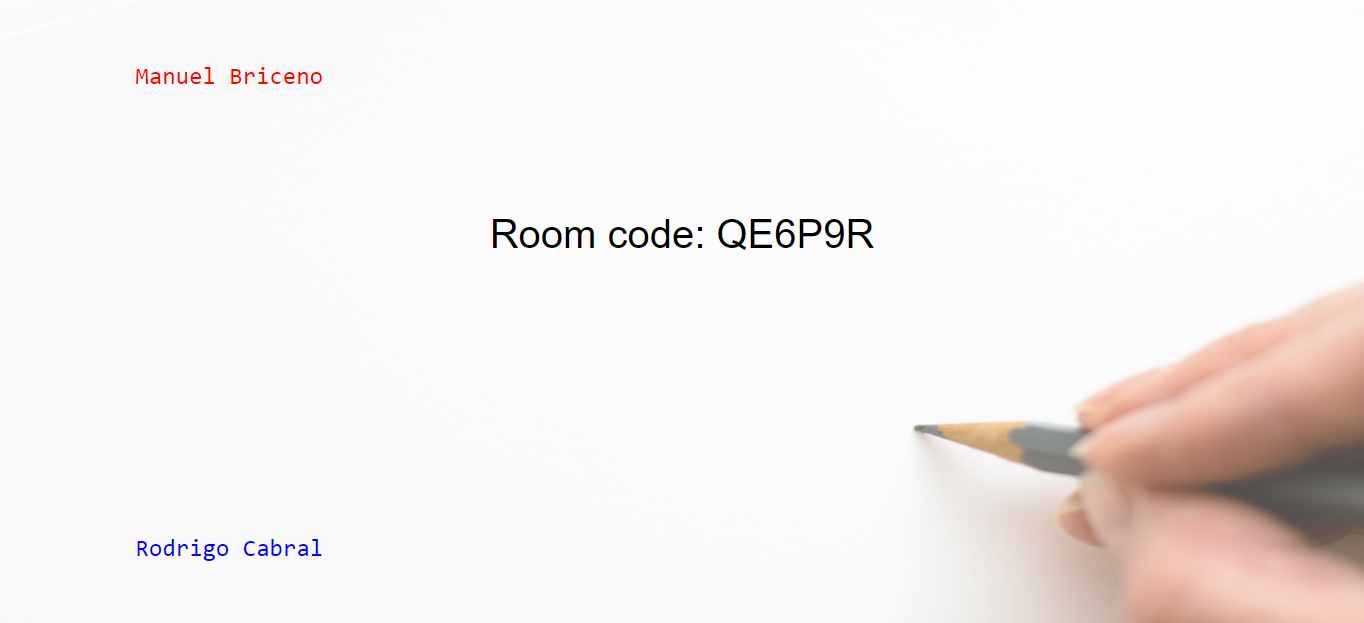


Este ecrã é o ponto de entrada da SPA. No caso de o estado de jogo ser corrompido ou a conexão com o servidor seja quebrada, nomeadamente se o utilizador tentar andar para trás ou refrescar a página, a aplicação é também redirecionada para este ponto. Optamos por esta abordagem visto que podem ser perdidas notificações que tornam o estado aplicacional inconsistente quebrando o fluxo de jogo.

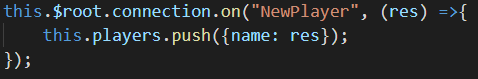
Quando o botão de “New Room” for clicado é lançado o evento “RegisterUIClient” que visa iniciar uma nova sessão de jogo, recebendo do servidor a notificação “AckUiClient” que contém no payload o roomCode. Após receber com sucesso a notificação é feita a navegação para o próximo ecrã, Join, eviando como propriedade o roomCode.

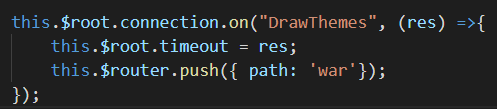
O ecrã de Join apresenta o roomCode e os jogadores.



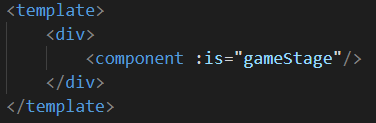
É com este código que os jogadores se ligam à sessão de jogo atual. Quando um jogador insere o código no seu dispositivo android o servidor notifica a SPA com “NewPlayer”, contendo na notificação um novo jogador que irá ser apresentado no ecrã.

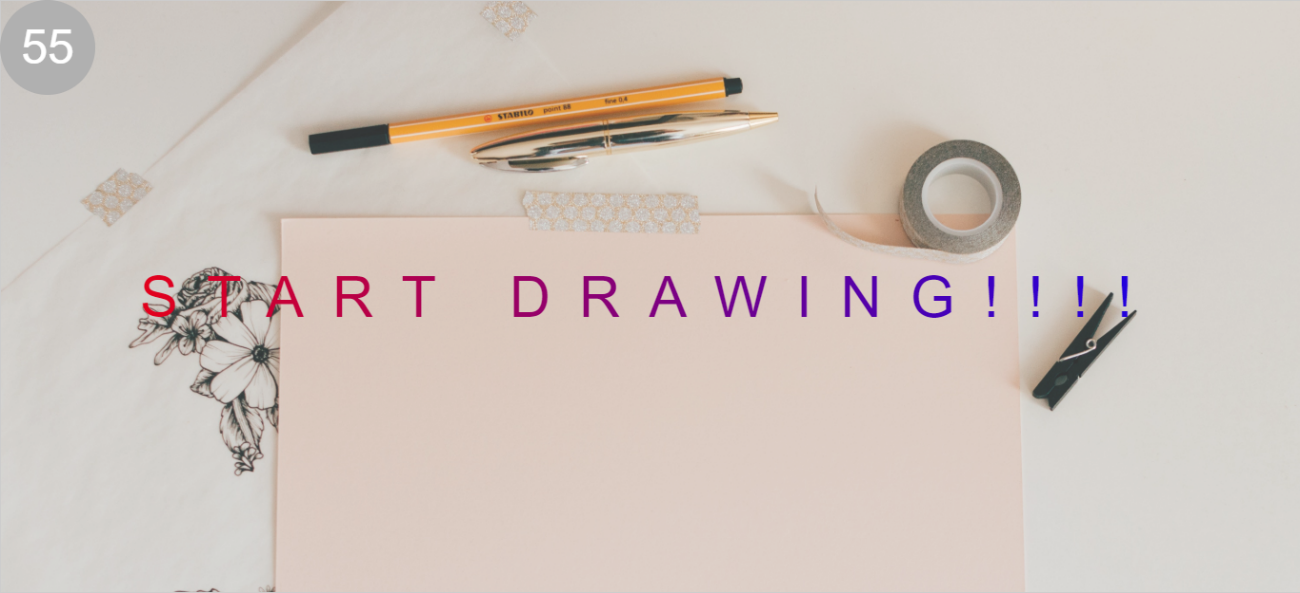


Quando o servidor der início ao jogo a SPA será notificada com “DrawThemes”. Como parâmetro da notificação é passado o valor que irá ser temporizado para a próxima fase de jogo.



A próxima fase de jogo representa um ciclo. O início do ciclo é o início da ronda, ecrã de DrawTime, de seguida é a fase de adivinha, DrawGuess, e por fim a fase de pontuações, Scores. De forma a simplificar o processo de desenvolvimento, e visto que todas estas fases estão interligadas, foi criado um componente base Wars que representa um contentor dos restantes componentes. Neste os componentes são carregados dinamicamente, através da variável gameStage, à medida que o servidor notifica a SPA para a mudança de estado.

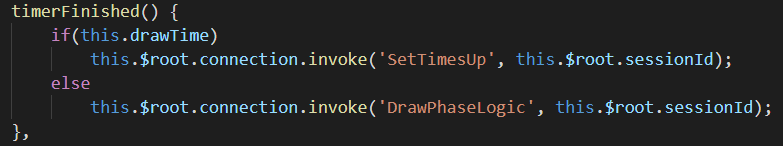


O ecrã de “DrawTime” serve para informar os jogadores que entramos na fase de desenho e podem começar a desenhar nos seus dispositivos android. 

Existem 2 formas de esta fase chegar ao fim. A primeira é o tempo estipulado terminar e a SPA lançar o evento “SetTimesUp”. Quando este evento é lançado o servidor recolhe os desenhos em falta, e notifica com “ReadyToShowDraws”. A segunda é todos os desenhos terem sido submetidos a tempo, e o servidor notifica igualmente com “ReadyToShowDraws”. Como parâmetro da notificação, é enviada a quantidade de tempo que o ecrã de transição de fase deve ser apresentado.

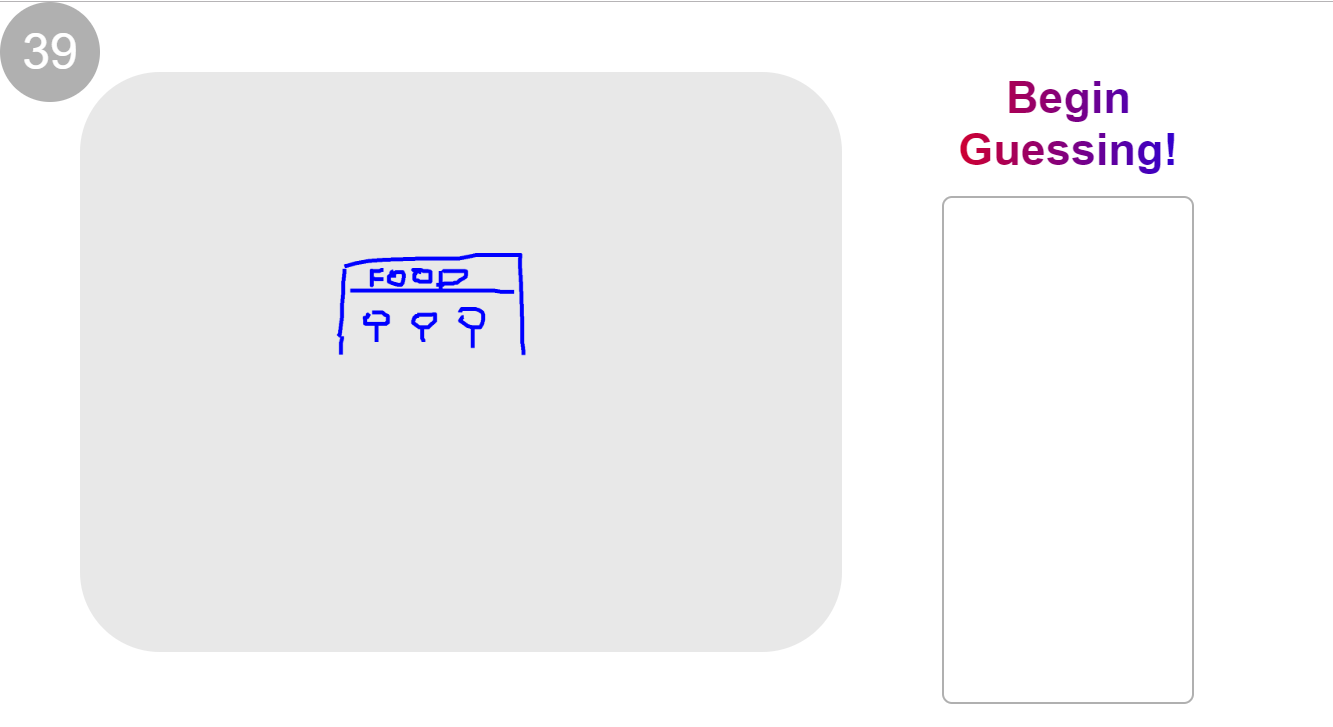


Quando a temporização do ecrã de transição termina, a SPA lança o evento “DrawPhaseLogic” de forma a notificar o servidor que se encontra apta a receber o desenho.



A SPA aguardará passivamente pela notificação “ShowDrawing”, que como parâmetro recebe um url para o desenho a apresentar e um novo valor de tempo para a próxima fase de jogo, e transita para ecrã de WarGuess.

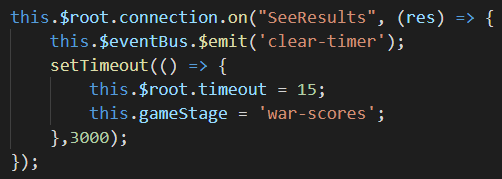
O ecrã de WarGuess é responsável por mostrar o desenho dos jogadores. Contém também uma área onde as adivinhas feitas pelos jogadores são mostradas.



À medida que os palpites vão sendo inseridos no dispositivo android, o servidor vai notificando a SPA com “PlayerGuess”. Esta notificação contém o palpite atual do jogador e se está ou não correto. Caso esteja, em vez do palpite, é apresentado “Correct!” para os restantes não saberem qual a *“guess”* correta.

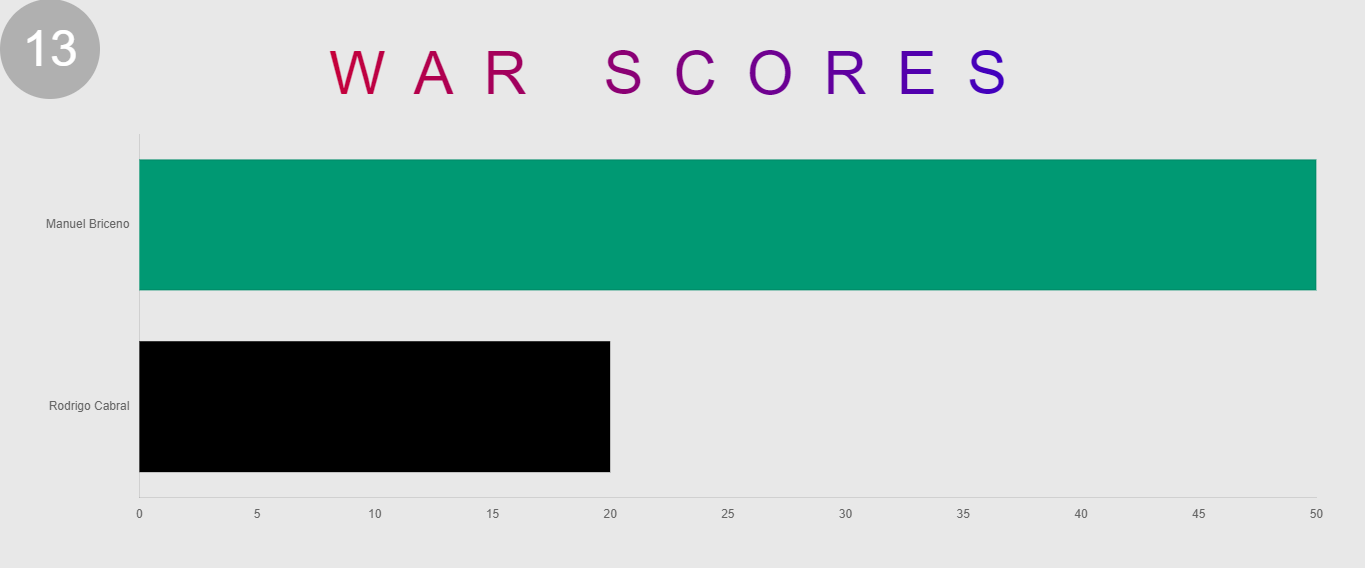
Quando o tempo terminar é enviado “NextGamePhase” pela SPA para informar o servidor do sucedido. Caso o servidor receba “NextGamePhase” ou todos os jogadores acertarem dentro do tempo limite o servidor notifica com “SeeResults” passando como parâmetro as pontuações atuais dos jogadores.

Ao receber “SeeResults” e de forma a que a transição não seja instantânea, ocultando o último palpite submetido, é criado um contador de 3 segundos sendo apresentado no browser um aviso de que não serão aceites mais submissões.

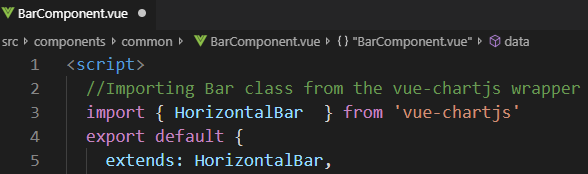


Quando os 3 segundos terminarem, a variável global timeout é afectada com 15(segundos) que será a duração do próximo ecrã “Scores” seguido dessa mesma transição de ecrã.

É no ecrã de “Scores” que as pontuações dos jogadores são mostradas. Para tal utilizamos uma biblioteca externa, *vue-chartjs*, de forma a criar um gráfico de barras.



Esta biblioteca permite de uma forma simplista a criação de vários tipos de gráficos, apenas é necessário importar o tipo de gráfico pretendido, criar um novo componente e extender o gráfico importado passando para a função de render os dados e as configurações do gráfico.

O tempo estipulado para a visualização das pontuações é de 15 segundos. Quando o tempo termina a SPA envia para o servidor “ResultsShown”. O servidor irá responder com 1 de 3 eventos: “ShowDrawing”, “NextRound” e “EndOfGame”.

No caso de os desenhos de todos os jogadores ainda não terem sido apresentados o servidor envia “ShowDrawing” e passa a ser apresentado o ecrã “WarGuess” com o desenho do próximo jogador, seguindo o fluxo explicado anteriormente até ao ecrã “Scores”. Continua assim sucessivamente até todos os desenhos desta ronda terem sido apresentados.

Com o término da apresentação dos desenhos, caso ainda existam rondas de jogo, o servidor envia “NextRound” passando um novo valor de tempo. A SPA ao receber esta notificação apresenta novamente o contador, com o novo valor recebido, e informa os jogadores que uma nova ronda está prestes a começar. Quando o tempo termina é enviado para o servidor o evento “RoundEndedAck”. O servidor ao receber esta notificação envia “DrawThemes” com o devido valor de tempo. Neste caso a SPA reencaminha para o ecrã de “DrawTime” e o ciclo inicial de jogo recomeça.

Quando já não existem mais rondas a serem jogadas, o servidor envia “EndOfGame” e a SPA reencaminha a aplicação web para o ecrã “Home” onde se pode criar uma nova sessão e iniciar um jogo de novo.

# Planeamento

## Até ao momento

Até à data, este projeto encontra-se perto da conclusão da versão mínima apresentável. Faltando apenas o final do ciclo de jogo (envio das adivinhas e a visualização das classificações), e a hospedagem do servidor e da *SPA* no *AWS*.

## 27 maio, Cartaz e Versão Beta

Planeamos até esta data, ter uma versão de DrawWars jogável, ainda que imperfeita. Com toda a logica de jogo concluída, Servidor e *SPA* hospedados.

## Até 16 Julho

Serão feitas melhorias e correções, e a versão final do relatório.