# Servidor

O Servidor é a peça central desta solução, tendo como responsabilidades principais gerir a comunicação entre os vários componentes da solução, Aplicação e SPA, gerir os jogadores, sessões de jogo e fluxo de jogo. Disponibiliza uma Web API *RESTful* e uma SignalR[1] Hub para conexões externas, por onde se processa a comunicação com a aplicação e o FrontEnd.

Este componente aborda várias problemáticas, tais como definir o protocolo temporal, a gestão de dados associados às sessões.

A Estrutura do Servidor é composta por 4 peças principais, o **CoreManager** que é a peça central da gestão do jogo e dos jogadores, um SignalR Hub chamado **DrawWarsHub** que trata da comunicação via *Websockets* com os clientes, a Web API *RESTful* que disponibiliza uma serie de métodos invocados tanto durante o fluxo de jogo, como para aceder a recursos guardados na Base de Dados, e um modulo de data que gere o acesso a dados SQL.

A implementação do servidor foi feita sobre a Framework ASP.NET Core, e tira partido do SignalR, que é uma biblioteca que agiliza a comunicação em tempo real através de 4 tipos de transporte, *WebSockets, Long Pooling, Server Sent Events* e *Forever Frame* permitindo o servidor enviar e receber notificações assíncronas para os clientes.

# Arquitectura

**CoreManager** é uma classe estática, que mantem os dados de jogo. Contem uma instância **ConcurrentDictionary** de Strings, que mapeia o código da sala no Id da sessão, e uma instância de **MemoryCache** cujos **CacheItems** são sessões de jogo **GameSession** e está configurada com *SlidingExpiration* em 10 minutos, que significa que caso a instância da sessão não seja utilizada em 10 minutos, esta é removida.

Ao ser invocado a criação de uma sessão, é gerado um código de sala único e aleatório composto por Letras maiúsculas e números através da função GenerateRoomCode().

O **CoreManager** disponibiliza ainda uma serie de métodos que permitem questionar ou alterar dados das sessões e dos jogadores.



Figura

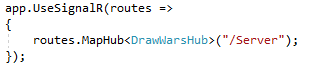
A **GameSession** é então a própria sessão de jogo, contem a identificação da sala (ROOM CODE) e o identificador da sessão, contem o id da conexão ao cliente de *FrontEnd*, e dados referentes ao estado do jogo como o tema atual, se já começou, as rondas, etc.., e uma lista com os jogadores associados à sessão.

É através de uma função desta classe que é feito o acesso aos temas, estes encontram-se guardados em base de dados, como tal é invocada uma *query* que obtém a lista de temas.

Esta classe contém também algumas funções de leitura e edição dos dados do jogo, e outras que respondem a questões acerca do estado do jogo, tais como se o jogo terminou, se já foram submetidos todos os desenhos, e outras que tal.

Os jogadores, classe **Player** contém apenas campos que representam a sua informação pessoal e estado, como Id, Id de conexão, Id do dispositivo, Pontos, Nome entre outros. Contém também uma variável de controlo do tipo int chamada *InteractionCounter* que é incrementada de forma atómica, esta variável contabiliza as interações entre o servidor e a aplicação android, e é utilizada quando existem falhas na comunicação e é necessário saber se foi perdida informação. Essa informação é obtida através da comparação entre esta variável, e uma outra do lado da aplicação, que server o mesmo propósito, e que representa também a quantidade de ações registada de cada lado.

**DrawWarsHub** é uma classe que implementa **Hub** do SignalR.

Na configuração este **Hub** é definido para um determinado *routing*, e é usado nas comunicações WebSocket com os clientes.

Ao conectarem-se com este **Hub** é feito o HandShake entre servidor e cliente, e estes passam a estar conectados via WebSockets, a partir desse ponto ambos estão aptos a enviar notificações entre eles.

É exposta uma interface de métodos assíncronos, com a qual os clientes devem comunicar.



Figura

Numa primeira fase o **DrawWarsHub** receberá pedidos de registo da parte do FrontEnd, sendo invocado o método “*RegisterUIClient”* que trata de criar uma nova sessão através do **CoreManager**, registar o id da comunicação com este cliente como o id do UIClient, id este que será usado como alvo da notificação pelo **Hub**, e é lhe notificada a informação referente a esta nova sessão, através da ação registado com assinatura “*AckUIClient*”.

Já os jogadores, através da aplicação móvel, poderão inscrever-se invocando a ação “*Inlist*” passando o código da sala que se pretendem conectar, e o seu Id de dispositivo. Caso exista uma sessão com esse código, e que ainda permita a inscrição de mais jogadores, o android é notificado com “*AckSession*” e é passado um contexto **Context** que representa a associação do user à sessão, caso a sessão não existe o utilizador é notificado com “*NonExistingSession*”.

De seguida o utilizador pode enviar o seu username, que se ocorrer com sucesso vai despoletar uma notificação ao ClienteUI com a informação do nome do novo jogador, para que possa ser refletido no ecrã.

Eventualmente, após todos os jogadores estarem registados, o **DrawWarsHub** será notificado com “*Ready*”, quando isto acontecer será dado início ao jogo, o campo “*Started”* da sessão passa a *true*, são obtidos os temas e enviados em forma de notificação para os jogadores, e é também notificado o FrontEnd da transição de fase.

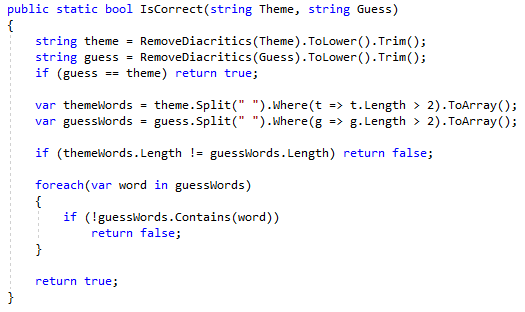
Associada a esta notificação de início de jogo “*DrawThemes*” é passado um valor inteiro que representa o tempo para esta fase.

A contagem do tempo poderia ser feita do lado do servidor, e no final da contagem terminaria inequivocamente a fase, no entanto isso levantaria varias questões, como por exemplo, o que aconteceria se um utilizador submetesse o seu desenho atempadamente, mas a latência o fizesse ultrapassar o tempo limite, sendo assim rejeitado o seu desenho, mesmo tendo cumprido as regras do jogo, ou que devido a latência por parte do frontend, o contador visual de tempo estivesse deslocado do tempo real, e os utilizadores pensassem que tinham tempo restante, mas na verdade o verdadeiro temporizador já tinha terminado. Como tal, e com o objetivo de melhorar a experiência de utilização, optamos por manter a contagem do tempo do lado do frontend, ao terminar este notifica o servidor alertando que a contagem já acabou, e este por consequência, notifica os clientes que o tempo acabou. Esta solução não é livre de problemas, mas garante que o utilizador tem sempre pelo menos o tempo estipulado, no entanto continuaria a poder existir desvio a quando do enviar da resposta da parte cliente, para tal a solução encontrada foi a de não bloquear entregas depois do fim do tempo, mas sim forçar o utilizador a enviar o que tem até ao momento, e esperar passivamente que todas sejam entregues. Assim, apesar de favorecer os jogadores que podem ter tido mais algum tempo para terminar as suas ações, não taxa quem cumpriu as regras e tem problemas de latência, o que tomamos como sendo benéfico para a *User Experience.*

Durante esta fase “*DrawThemes*” do jogo, os jogadores irão desenhar e submeter os seus desenhos. Essa submissão será feita à WebApi, via POST, que será mencionada mais à frente neste relatório.

Quando a submissão é feita com sucesso, o servidor é notificado com “*DrawSubmitted”,* e quando todos os jogadores submetem os seus desenhos, o **DrawWarsHub** notifica o servidor que já todos os jogadores submeteram os seus desenhos e passa-lhe mais uma vez o valor de segundos a esperar para apresentar a transição de fase aos jogadores.

Quando o servidor é notificado do final desse tempo, notifica o FronEnd com o desenho que deve ser apresentado, e os jogadores com a informação de início da fase de adivinha. Durante esta etapa os jogadores enviarão as suas *Guesses*, para tentar adivinhar o título original do desenho.

Foi desenvolvido um algoritmo de forma a facilitar esta adivinha, removendo todas as palavras com comprimento igual ou menor a dois caracteres, removendo espaços nas extremidades da frase, normalizando os caracteres especiais e tornando toda a frase em minúsculas.

À medida que os jogadores vão acertando, o FrontEnd vai sendo notificado do seu sucesso, assim como os utilizadores vão recebendo feedback se acertaram ou erraram na *Guess*. Quando todos os utilizadores acertam, é dado como terminada essa ronda e lançada a notificação “*SeeResults*” para todos os intervenientes.

Enquanto houverem desenhos para mostrar, a fase de adivinha repete-se, quando todos os desenhos foram mostrados, o **CoreManager** é questionado se ainda existem rondas a serem jogadas, que faz com que se repita a fase de enviar os temas, ou que já foram jogadas todas as rondas, e como tal terminou o jogo.

Caso o jogo tenha chegado ao fim, é gravado na base de dados as scores de todos os jogadores intervenientes, associados à sessão, é enviada uma notificação a transmitir a informação de que o jogo terminou.

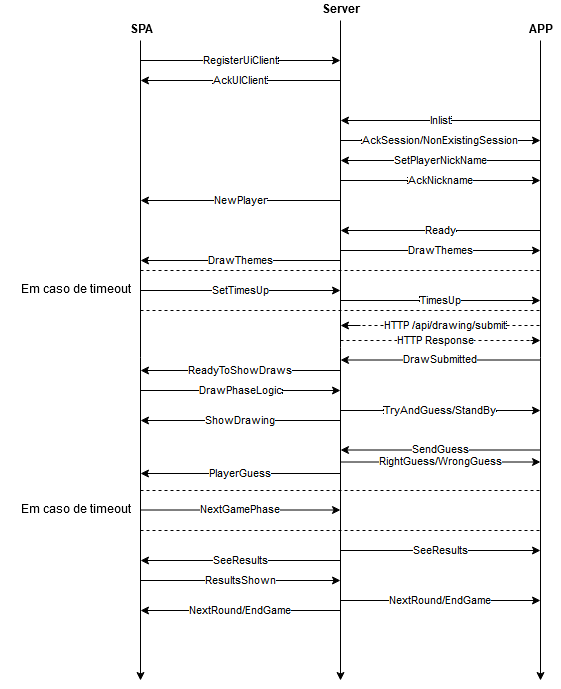


Figura Diagrama temporal assumindo só um jogador e só uma ronda

O Servidor disponibiliza uma serie de endpoints, que servem tanto o propósito do funcionamento do jogo, como alguns métodos para futuras implementações ou módulos.

[NECESSÁRIO AUTENTICAÇAO?]

Para os métodos associados a jogabilidade não é necessária autenticação, no seu formato tradicional. No entanto, os pedidos são acompanhados de um contexto, que indica sessão de jogo e id de utilizador. Caso este não exista, ou não se encontre em jogo, o pedido é ignorado.

É apresentada, de seguida, a tabela com os endpoints atualmente disponíveis.

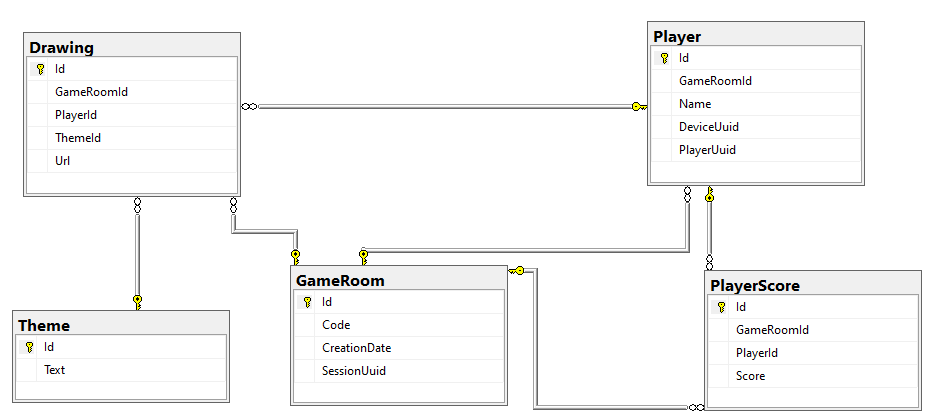
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **URI** | **Resposta** | **Parametros** | **Descrição** |
| POST | /api/drawing/submit | {uri, errorMessage} | SessionId  PlayerId | Caso a sessão exista, e o utilizador exista associado à sessão, o desenho é guardado nos servidores AWS, e o uri é gravado na base de dados, contextualizado com o utilizador e sessão. |
| GET | /api/gameInfo/{Session}/{Player} | {Count} | Session  Player | Retorna o contador de interações do servidor para este utilizador, nesta sessão. |

Figura

Data

Para esta solução foi criada uma Base de Dados relacional do tipo SQL Server, para a persistência de dados. A conexão a esta Base de Dados é feita apenas do lado do servidor web, e tem como propósito guardar informação relativa ao histórico de jogo, e conter os temas a serem desenhados.

O diagrama desta BD é o seguinte,



Figura