**== Pontos importantes de implementação ==**

- Vamos tratar o envio de objetos um pouco diferentemente. Não há mais payload como antes e a verificação Server-Side do que deve ser feito ao receber um objeto será feito num "case" grandão comparando os tipos de objetos. É meio feio, mas é aquela história, não é curso de eng. de software etc etc. O que importa é que vamos conseguir implementar algo muito legal no final das contas!

- Assim, os objetos vão sempre ser do tipo "NomeDoQueÉPedido" + "Request" ou "Response". Ex: UserDataRequest, UserDataResponse, CharacterDataRequest, etc.

- O "Nome do que é pedido" pode ser um objeto no banco (UserData, CharacterData, LocalData) ou uma função qualquer (ListUsers, ListLocations, RemoveCharacter... ).

- Os Requests e Responses vão funcionar mais como "RequestData" e "SendData" no sentido literal. Ex:

       \_\_----------- Request ------> **Servidor**<--- Acessa ---> (**DB**)

**Cliente**<--\_\_\_\_ Response \_\_\_\_\_\_\_/

ou

       \_\_----------- Request ------> **Cliente**

**Servidor**<--\_\_\_\_ Response \_\_\_\_/

        \

         \\_\_--> (**DB**)

ou até (mais no sentido SendData)

**Servidor/Client** ------ Response ------> **Servidor/Cliente** (podendo acessar o **DB**depois, se for **Servidor**)

- Não vamos lidar com autenticação formal por token nessa primeira versão, será mais fácil apenas recebermos o objeto de UserDataRequest (que é interpretado como um pedido de autenticação) e checarmos se o email-password passado confere no banco.

- Bolamos uma opção diferente para o envio constante de coordenadas. No caso, o envio constante de coordenadas faz a verificação de se o cliente está numa região específica ser server-side. A opção é que isso seja Client-Side (próxima seção).

**== Checagem de região Client-Side ==**

- Partimos do principio que o cliente se locomove uma distância máxima (100km, arbitrário) em um tempo específico (30 minutos, arbitrário). Isso significa que, em um espaço de 30 minutos, o cliente não vai mais longe que 100km do ponto que estava 30 minutos atrás.

- Assim, o servidor irá, na hora do Login/Autenticação do usuário, mandar uma LocationListResponse pro cliente com todos os locais de batalha inclusos nesse raio de 100km em volta dele (sua coordenada atual passa a ser um parâmetro no UserDataRequest). O cliente vai registrar esses locais como os locais acessíveis por ele.

- Com isso, a checagem de se o cliente entra em uma região passa a ser feita pelo próprio cliente. O cliente só precisa enviar uma coordenada nova a cada 30 minutos, atualizando as regiões visíveis ao mesmo.

**== Mecânica Básica do Jogo e Comunicação ==**

- O jogador anda pela cidade com o seu celular ao longo do dia. Quando ele passa por uma região de batalha, o próprio cliente identifica (pelo o que foi sugerido na seção anterior) e inclui uma batalha possível (que expira em x horas) na lista de batalhas pendentes (além de uma notificação no celular).

- (arbitrário) Ocorrerão três batalhas por dia, uma às 8h, outra às 14h e outra às 20h (pode variar dependendo da região). O momento da batalha em si é apenas um evento Server-Side em que o servidor calcula, com base em todos os FightRequests recebidos até então, o resultado dessa batalha e guarda o mesmo para que os clientes os recebam.

- Os clientes por sua vez recebem esse tipo de informação ao entrarem no jogo (você tem resultados de batalha novos!), junto com os dados de seu personagem ou ao clicarem na Batalha na lista de batalhas pendentes (a definir).

=============================================================================