FYP part 2

o uso de **Spring WebSocket** pode **reduzir o tráfego no servidor**, dependendo do cenário e da funcionalidade. Isso acontece porque, ao contrário de soluções baseadas em **polling**, o WebSocket estabelece uma conexão persistente entre cliente e servidor, permitindo a troca de dados sem a necessidade de múltiplas requisições HTTP. Vamos entender melhor:

**Como WebSockets Reduzem o Tráfego**

1. **Eliminação do Polling**:
   * **Polling:** O cliente faz requisições HTTP em intervalos regulares para verificar se há novos dados disponíveis. Cada requisição consome largura de banda e recursos no servidor, mesmo quando não há dados para enviar.
     + Exemplo: Se você configurar polling a cada 5 segundos para 1000 usuários, o servidor processará **12.000 requisições por minuto**, mesmo que a maioria delas não tenha novos dados.
   * **WebSockets:** Com uma conexão persistente, o servidor só envia dados quando necessário, eliminando requisições redundantes.
2. **Menor Sobrecarga no Protocolo**:
   * Cada requisição HTTP (mesmo no polling) envolve:
     + Cabeçalhos de requisição/resposta.
     + Negociação de conexão (TCP/TLS).
   * Com WebSockets, a negociação ocorre apenas uma vez no início da conexão. Após isso, apenas os dados reais (payload) são transmitidos, reduzindo a sobrecarga.
3. **Transmissão Direcionada**:
   * WebSockets permitem que o servidor envie mensagens diretamente a clientes específicos ou a grupos (broadcast). Isso é mais eficiente do que retornar dados para todos os clientes a cada requisição, como ocorre no polling.

**Quando WebSockets Realmente Reduzem Tráfego?**

1. **Notificações em Tempo Real:**
   * Em vez de o cliente perguntar constantemente "Há algo novo?", o servidor envia notificações diretamente.
   * Exemplos no seu projeto:
     + Notificar o usuário ao atingir milestones.
     + Atualizações automáticas em componentes do mapa ou galeria.
2. **Alta Frequência de Atualizações:**
   * Se você precisa atualizar o cliente com frequência (e.g., a cada segundo), o WebSocket é significativamente mais eficiente do que enviar múltiplas requisições HTTP.
3. **Muitos Usuários Simultâneos:**
   * Para sistemas com alta concorrência, WebSockets ajudam a evitar a sobrecarga causada por milhares de requisições HTTP simultâneas.

**Casos em que WebSockets Podem Não Reduzir Tráfego**

1. **Baixa Frequência de Atualizações:**
   * Se as notificações são raras (e.g., uma notificação por dia), a redução de tráfego será mínima, e o custo de manter conexões persistentes pode ser desnecessário.
2. **Simples Requisições Assíncronas:**
   * Se os dados não precisam ser entregues imediatamente, usar uma abordagem baseada em REST com mecanismos como long polling ou e-mails pode ser suficiente.

**Impacto na Escalabilidade**

Embora WebSockets reduzam o tráfego, eles exigem recursos para manter conexões abertas:

* **Uso de Memória e Conexões TCP:** Cada cliente conectado ocupa memória e uma conexão aberta no servidor.
* **Soluções Escaláveis:** Para muitos usuários (1000+), é recomendável:
  + Usar um balanceador de carga para distribuir conexões.
  + Integrar ferramentas como **Redis** para lidar com mensagens entre múltiplas instâncias de servidor.