**1) Divisão da Aplicação (Microsserviços e Desacoplamento)**

**Microsserviço 1 (Função Crítica):**  
**Chat & Voice** — gerencia chat em tempo real, canais de voz entre jogadores, moderação de chat (mensagens, muting, filtros). Muito importante para UX social, mas se cair NÃO pode derrubar o streaming de vídeo/jogo.

**Microsserviço 2 (Função Crítica):**  
**Recomendação & Descoberta** — motor de recomendações de jogos/sessões, listas personalizadas, feed. Melhora retenção/engajamento; falha degrada experiência, mas não impede o streaming.

**Mecanismo de Comunicação (conceito cloud):**  
**Broker de mensagens / Pub-Sub assíncrono** (ex.: managed Kafka, Pub/Sub ou um combo SNS/SQS).  
Por quê: comunicação assíncrona desacopla serviços — se Chat cair, eventos ficam enfileirados/retidos; Recommender pode consumir quando voltar; evita falha em cascata.

**2) Alta Resiliência e Disponibilidade (Geografia)**

**Conceito de Distribuição:**  
**Multi-Região ativa-ativa + Zonas de Disponibilidade (AZs) em cada região**, com **edge/CDN** e pontos de presença (PoPs) para menor latência.

**Justificativa:**

* **Tolerância a falhas regionais:** se uma região ou AZ cair, outras continuam atendendo (failover automático).
* **Latência:** game streaming é sensível à latência → localizar instâncias perto do usuário reduz jitter.
* **Capacidade e escalonamento:** ativa-ativa permite repartir 1M de usuários entre regiões; AZs garantem disponibilidade mesmo com falha local.
* **CDN/Edge:** descarrega assets estáticos e sinais de controle próximos do usuário, reduzindo carga nos servidores de streaming.

**3) Escalabilidade de Tráfego (Entrada)**

**Serviço Necessário:**  
**Global Load Balancer / Edge Load Balancer** (ex.: Global HTTP(S)/TCP/UDP Load Balancer) como primeiro ponto de contato, seguido por **regional load balancers** que encaminham para pool de aplicações/streaming.

**Função (como evita sobrecarga):**

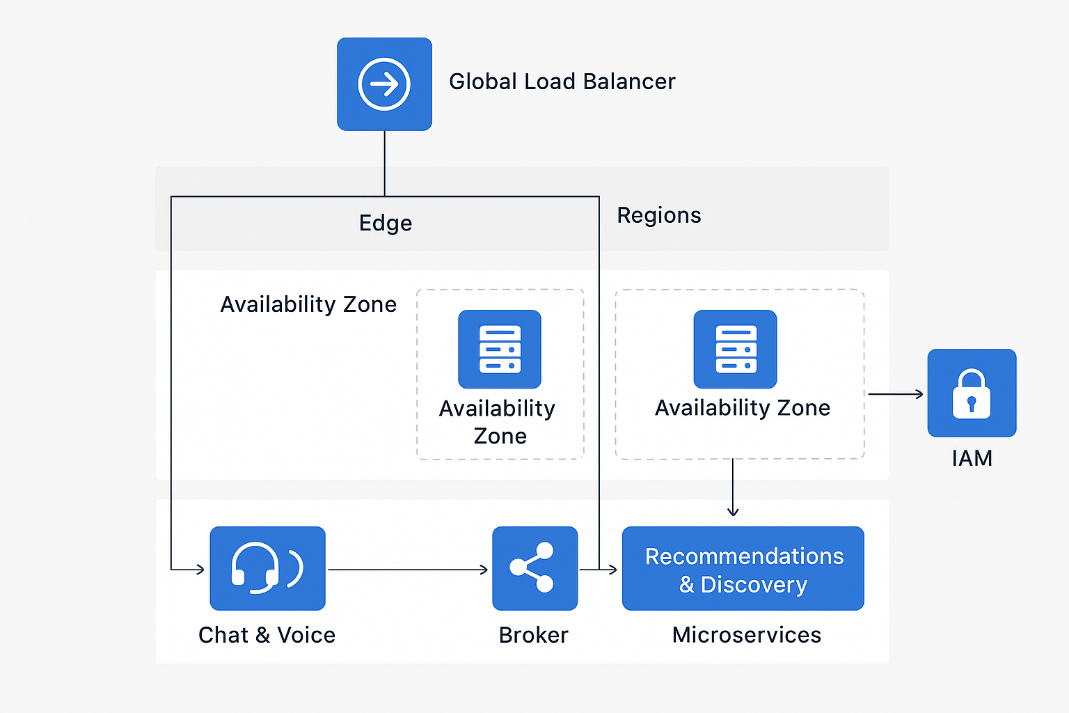
* **Distribuição inteligente** (por proximidade/latência, round-robin, peso) direciona tráfego para instâncias saudáveis.
* **Health checks** detectam instâncias malsucedidas e as retiram do pool.
* **Autoscaling integration:** LB envia tráfego apenas para instâncias ativas; quando a carga sobe, ASGs/instance pools escalam horizontalmente.
* **Rate limiting / WAF no edge:** protege contra rajadas e abusos, evitando saturar backend.
* **Sticky sessions / consistent hashing** (quando necessário para sessões WebRTC/UDP) mantêm afinidade sem sobrecarregar uma única instância.

**4) Segurança por Design (Identity and Access)**

**Pilar de Segurança (Conceito):**  
**Identity & Access Management (IAM) + Mutual TLS (mTLS) dentro de um modelo Zero-Trust** — ou resumindo: **autenticação e autorização forte para serviços (service-to-service auth)**.

**Por que é crucial:**

* **Garante que só o Microsserviço1 autenticado fale com o Microsserviço2** (credenciais curtas, tokens JWT assinados, mTLS).
* **Menor privilégio:** políticas IAM limitam quais operações cada serviço pode executar.
* **Previne movimento lateral e vazamento:** mesmo se uma instância for comprometida, permissões restritas reduzem impacto.
* **Auditoria e rastreabilidade:** logs de autenticação/autorização permitem investigação e conformidade.



**O que é o GameStream?**

É o sistema que transmite os jogos para os jogadores. Tudo acontece pela internet — como assistir um filme em streaming, só que jogando.

**1. Global Load Balancer (Balanceador de Carga Global)**

Pense nele como um **porteiro inteligente**.  
Quando 1 milhão de pessoas entram ao mesmo tempo, ele vê quem está mais perto de qual “filial” (servidor) e envia cada jogador para o local certo.  
Assim, ninguém espera na fila e nenhum servidor fica sobrecarregado.

**2. Regiões e Zonas de Disponibilidade**

Essas “zonas” são **centros de dados em diferentes lugares do mundo**.  
Se um deles cair (por exemplo, falta de energia em São Paulo), outro (como no Rio ou Miami) continua funcionando.  
Isso garante que o GameStream **nunca pare de funcionar**, mesmo com problemas locais.

**3. Microsserviços (pequenas partes do sistema)**

O GameStream é dividido em **partes independentes**.  
Por exemplo:

* **Chat & Voice** → cuida da conversa por texto e voz entre os jogadores.
* **Recomendações & Descoberta** → sugere novos jogos, mostra o que seus amigos estão jogando.

Essas partes são separadas para que, se uma falhar, **o resto continue funcionando normalmente** (você pode continuar jogando mesmo que o chat pare).

**4. Broker (mensageiro entre os serviços)**

É um “carteiro digital” que leva mensagens entre os microsserviços.  
Se um serviço estiver fora do ar, o mensageiro guarda a mensagem e entrega quando ele voltar.  
Isso impede que uma falha pequena cause uma reação em cadeia.

**5. IAM (Controle de Identidade e Acesso)**

Pense nisso como um **cartão de acesso digital**.  
Ele garante que cada parte do sistema só fale com quem realmente deve.  
Por exemplo, o serviço de chat só pode conversar com o de recomendações se estiver autenticado — assim, **nenhum invasor ou serviço falso consegue se passar por outro**.

**Em resumo:**  
O GameStream é planejado como uma cidade moderna:

* vários bairros (microsserviços),
* ruas inteligentes que desviam o trânsito (load balancer),
* geradores de energia espalhados (zonas e regiões),
* e segurança em cada portão (IAM).