1. Visão Geral da Arquitetura

- **Componentes Principais**:
    - **Front-end/Aplicativo**: Ponto de entrada do usuário (ex.: app mobile/web).
    - **API Gateway (JWT)**: Centraliza autenticação (JWT) e roteamento para APIs.
    - **API Usuário**: Gerencia usuários (CRUD e autenticação).
    - **API Jogos**: Gerencia catálogo de jogos (CRUD, busca, biblioteca, compras).
    - **API Pagamento**: Inicia e consulta pagamentos.
    - **Bancos de Dados**: JogosDB (para jogos), UsuárioDB (para usuários, possivelmente compartilhado com pagamentos).
    - **Event Store**: Armazena eventos (append-only) para event sourcing.
    - **Barramento de Eventos**: Propaga eventos para processamentos assíncronos.
    - **Elasticsearch**: Índice para buscas rápidas (ex.: catálogo de jogos).
    - **Função Indexer Assíncrona**: Atualiza Elasticsearch com base em eventos.
    - **Fila de Pagamento**: Fila assíncrona para comandos de pagamento.
    - **Worker de Pagamento Assíncrono com Fila**: Processa pagamentos de forma assíncrona.
    - **DLQ de Pagamento**: Dead Letter Queue para falhas no processamento.
    - **Webhook Pagamento**: Recebe notificações externas (ex.: de PSP).
    - **Função Pagamento Webhook**: Processa webhooks e lida com sucessos/falhas.
    - **Logs TXT**: Observabilidade básica (logs em texto).
- **Cores e Agrupamentos**:
    - Amarelo: Camada de apresentação (Front/API Gateway).
    - Roxo: Subsistema de Jogos.
    - Cinza: Subsistema de Usuário.
    - Verde: Subsistema de Pagamento.
    - Rosa: Fluxo assíncrono de pagamentos (com falhas e sucessos).
    - Laranja: Componentes compartilhados (Event Store, Barramento, etc.).

- **Fluxos Gerais**:
    - O front-end interage exclusivamente via API Gateway.
    - APIs são RESTful (métodos como POST/GET/PUT/Delete indicados).
    - Processos assíncronos usam filas e barramento para desacoplamento.
    - Pagamentos são orquestrados de forma assíncrona com tratamento de falhas.

2. Descrição Detalhada dos Fluxos

Aqui, descrevo os fluxos principais com base nas setas e conexões no diagrama. Dividi por subsistemas para clareza.

2.1 Fluxo de Usuário (Subsistema Cinza)

- **Componentes Envolvidos**: API Gateway → API Usuário → UsuárioDB.
- **Operações**:
    - CRUD Usuário: POST/GET/PUT/Delete para gerenciamento de perfis.
    - Autenticação: POST /login/auth para login (gera JWT).
- **Fluxo Passo a Passo**:

1. Aplicativo/Front envia requisição (ex.: POST /users para cadastro) ao API Gateway.
2. API Gateway valida JWT (se aplicável) e roteia para API Usuário.
3. API Usuário processa (ex.: cria usuário) e persiste em UsuárioDB.
4. Resposta retorna via Gateway ao Front.

- **Eventos**: Publica eventos no Event Store (ex.: UserRegistered).

2.2 Fluxo de Jogos (Subsistema Roxo)

- **Componentes Envolvidos**: API Gateway → API Jogos → JogosDB → Event Store → Barramento de Eventos → Função Indexer Assíncrona → Elasticsearch.
- **Operações**:
    - CRUD Jogos: POST/PUT/GET/Delete para catálogo.
    - Busca Jogos/Biblioteca: GET para consultas (usa Elasticsearch).
    - Compra: POST /jogos/compra para iniciar compra.
- **Fluxo Passo a Passo**:

1. Front envia requisição (ex.: POST /jogos para criar jogo) ao API Gateway.
2. Gateway roteia para API Jogos.
3. API Jogos processa (ex.: grava em JogosDB) e publica evento (ex.: GameCreated) no Event Store.

4. Event Store envia para Barramento de Eventos.

5. Barramento aciona Função Indexer Assíncrona, que atualiza Elasticsearch.

6. Para buscas: API Jogos consulta Elasticsearch diretamente.

7. Para compras: API Jogos publica camada de pagamento (ver fluxo abaixo).

- **Eventos**: GameCreated, GameUpdated; publica camada de pagamento para fila assíncrona.

2.3 Fluxo de Pagamento (Subsistema Verde + Rosa)

- **Componentes Envolvidos**: API Gateway → API Pagamento → UsuárioDB (ou PagamentoDB?) → Event Store → Barramento de Eventos → Fila de Pagamento → Worker de Pagamento Assíncrono com Fila → DLQ de Pagamento (para falhas) → Webhook Pagamento → Função Pagamento Webhook → Logs TXT.
- **Operações**:
    - Iniciar Pagamento: POST /pagamento.
    - Consulta: POST /pagamento/id para status.
    - Processamento Assíncrono: Worker processa fila; webhook lida com respostas externas (ex.: de PSP).
- **Fluxo Passo a Passo (Compra e Pagamento Assíncrono)**:

1. Durante compra (de API Jogos): Publica camada de pagamento no Event Store/Barramento.

2. Barramento envia para Fila de Pagamento.

3. Worker de Pagamento Assíncrono com Fila consome a fila e processa (ex.: chama PSP externo).

4. Se sucesso: Worker publica evento de sucesso no Barramento/Event Store.

5. Se falha: Envia para DLQ de Pagamento; publica evento de falha.

6. PSP externo notifica via Webhook Pagamento.

7. Função Pagamento Webhook processa o webhook, publica evento (sucesso/falha) no Barramento e gera Logs TXT.

8. Barramento atualiza status (ex.: OrderCompleted) e notifica (ex.: via função send-notification, implícita).

- **Tratamento de Falhas**:
    - Falha no pagamento: Vai para DLQ; publica "Falha no pagamento".
    - Sucesso: Publica "Sucesso/Falha no pagamento" (parece haver uma seta para ambos).
- **Eventos**: OrderCreated, PaymentSucceeded/Failed, PurchaseCompleted.

2.4 Fluxos Compartilhados e Observabilidade

- **Event Store e Barramento**: Central para todos os eventos assíncronos (ex.: de usuários, jogos, pagamentos).
- **Elasticsearch**: Usado apenas para buscas rápidas (alimentado por indexer).
- **Logs TXT**: Saída de observabilidade, conectada ao webhook e worker para tracing de falhas/sucessos.
- **Integração Geral**: Todas as APIs persistem em DBs e publicam eventos; Gateway é o único ponto de entrada.