**معماری کلی پروژه**

┌─────────────────────────────────────────────────────────────────────────────┐

│                           CHAT APPLICATION ARCHITECTURE                    │

└─────────────────────────────────────────────────────────────────────────────┘

┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐

│   FRONTEND      │    │    BACKEND      │    │   NATS SERVER   │

│   (Next.js)     │◄──►│   (.NET 8)      │◄──►│   (Message     │

│                 │    │                 │    │    Broker)     │

└─────────────────┘    └─────────────────┘    └─────────────────┘

│                       │                       │

│                       │                       │

▼                       ▼                       ▼

┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐

│   SQLite DB     │    │   gRPC Service  │    │   JetStream     │

│   (Local File)  │    │   (Protocol     │    │   (Persistence) │

│                 │    │    Buffers)     │    │                 │

└─────────────────┘    └─────────────────┘    └─────────────────┘

**جریان داده (Data Flow)**

**1. ارسال پیام (Send Message)**

User Input → Frontend → API Route → Backend → SQLite + NATS → Other Clients

│           │           │          │         │

│           │           │          │         └──► Broadcast to all tabs

│           │           │          └──► Save to database

│           │           └──► HTTP POST to /send

│           └──► /api/send route

└──► Type message

**2. دریافت پیام (Receive Message)**

NATS Message → Backend → SQLite → Frontend → UI Update

     │           │         │         │

     │           │         │         └──► React state update

     │           │         └──► Store in database

     │           └──► NATS subscriber service

     └──► Published by sender

**جزئیات هر سرویس**

**1. NATS Server (Message Broker)**

┌─────────────────────────────────────────────────────────────┐

│                    NATS SERVER                             │

├─────────────────────────────────────────────────────────────┤

│ Port 4222: Client connections (Backend)                    │

│ Port 8222: HTTP monitoring                                 │

│ Port 5080: WebSocket (Browser clients)                     │

├─────────────────────────────────────────────────────────────┤

│ Features:                                                   │

│ • JetStream enabled (persistence)                          │

│ • WebSocket support (browser clients)                      │

│ • Pub/Sub messaging                                        │

│ • Queue groups for load balancing                          │

└─────────────────────────────────────────────────────────────┘

**نقش NATS:**

* **Message Broker**: پیام‌ها را بین سرویس‌ها منتقل می‌کند
* **Real-time Communication**: ارتباط لحظه‌ای بین فرانت و بک‌اند
* **Scalability**: امکان اضافه کردن سرویس‌های بیشتر
* **Reliability**: پیام‌ها در JetStream ذخیره می‌شوند

**2. SQLite Database (Persistence Layer)**

┌─────────────────────────────────────────────────────────────┐

│                    SQLITE DATABASE                         │

├─────────────────────────────────────────────────────────────┤

│ Table: ChatMessages                                        │

│ ├── Id (Primary Key)                                       │

│ ├── RoomId (Chat room identifier)                          │

│ ├── User (Username)                                        │

│ ├── Content (Message text)                                 │

│ ├── Timestamp (When sent)                                  │

│ └── CreatedAt (Database timestamp)                         │

├─────────────────────────────────────────────────────────────┤

│ File: chat.db (local file)                                 │

│ Location: Backend project directory                        │

└─────────────────────────────────────────────────────────────┘

**نقش SQLite:**

* **Chat History**: ذخیره تمام پیام‌های ارسال شده
* **Persistence**: پیام‌ها بعد از restart حفظ می‌شوند
* **Query Support**: امکان جستجو و فیلتر پیام‌ها
* **Lightweight**: بدون نیاز به نصب database server

**3. gRPC Service (Backend API)**

┌─────────────────────────────────────────────────────────────┐

│                    gRPC SERVICE                            │

├─────────────────────────────────────────────────────────────┤

│ Service: ChatHistoryService                                │

│ ├── GetChatHistory(roomId) → List<Message>                │

│ ├── SaveMessage(message) → Success/Error                   │

│ └── StreamMessages(roomId) → Stream<Message>               │

├─────────────────────────────────────────────────────────────┤

│ Protocol: Protocol Buffers (.proto files)                  │

│ Port: 5001 (HTTP) + 5002 (gRPC)                           │

│ Language: C# (.NET 8)                                      │

└─────────────────────────────────────────────────────────────┘

**نقش gRPC:**

* **High Performance**: ارتباط سریع بین سرویس‌ها
* **Type Safety**: تعریف دقیق API با Protocol Buffers
* **Bidirectional Streaming**: امکان stream کردن پیام‌ها
* **Cross-platform**: قابل استفاده در زبان‌های مختلف

**نحوه اتصال سرویس‌ها**

**Connection Flow:**

1. NATS Server starts (Port 4222)

↓

2. Backend connects to NATS

↓

3. Frontend starts (Port 3000)

↓

4. Frontend connects to Backend (Port 5001)

↓

5. Frontend connects to NATS WebSocket (Port 5080)

↓

6. All services are connected and ready

**Backend to NATS Connection:**

*// Program.cs*

var natsUrl = builder.Configuration["Nats:Url"]; *// nats://localhost:4222*

var opts = new NatsOpts { Url = natsUrl };

var connection = new NatsConnection(opts);

*// Subscribe to chat messages*

await connection.SubscribeAsync<ChatMessage>("chat.room.\*", async (message) => {

*// Save to SQLite*

    await \_context.ChatMessages.AddAsync(message);

    await \_context.SaveChangesAsync();

});

**Frontend to Backend Connection:**

*// API calls*

const response = await fetch('http://localhost:5001/history?roomId=general');

const response = await fetch('http://localhost:5001/send', {

    method: 'POST',

    body: JSON.stringify(message)

});

**Frontend to NATS Connection:**

*// SSE for real-time updates*

const eventSource = new EventSource(`/api/events?roomId=${roomId}`);

*// When message sent, broadcast via SSE*

broadcastMessage(roomId, {

    type: 'new\_message',

    message: message

});

**Data Flow Diagram**

┌─────────────┐    ┌─────────────┐    ┌─────────────┐

│   USER 1    │    │   USER 2    │    │   USER 3    │

│  (Tab 1)    │    │  (Tab 2)    │    │  (Tab 3)    │

└─────┬───────┘    └─────┬───────┘    └─────┬───────┘

│                  │                  │

│                  │                  │

▼                  ▼                  ▼

┌─────────────────────────────────────────────────────────┐

│                FRONTEND (Next.js)                       │

│  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐    │

│  │ /api/send   │  │ /api/history│  │ /api/events │    │

│  └─────┬───────┘  └─────┬───────┘  └─────┬───────┘    │

└────────┼─────────────────┼─────────────────┼──────────┘

│                 │                 │

▼                 ▼                 ▼

┌─────────────────────────────────────────────────────────┐

│                BACKEND (.NET 8)                        │

│  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐    │

│  │   /send     │  │  /history   │  │ gRPC Service│    │

│  └─────┬───────┘  └─────┬───────┘  └─────┬───────┘    │

└────────┼─────────────────┼─────────────────┼──────────┘

│                 │                 │

▼                 ▼                 ▼

┌─────────────────────────────────────────────────────────┐

│                DATA LAYER                              │

│  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐    │

│  │    NATS     │  │   SQLite    │  │   SSE       │    │

│  │ (Message    │  │ (Database)  │  │ (Real-time) │    │

│  │  Broker)    │  │             │  │             │    │

│  └─────────────┘  └─────────────┘  └─────────────┘    │

└─────────────────────────────────────────────────────────┘

**مزایای این معماری**

**1. Scalability:**

* امکان اضافه کردن سرویس‌های بیشتر
* Load balancing با NATS queue groups
* Microservices architecture

**2. Performance:**

* gRPC برای ارتباط سریع سرویس‌ها
* NATS برای messaging با latency کم
* SQLite برای ذخیره‌سازی سریع

**3. Reliability:**

* پیام‌ها در JetStream ذخیره می‌شوند
* Auto-reconnect در صورت قطع اتصال
* Fallback mechanisms

**4. Real-time:**

* SSE برای push notifications
* NATS WebSocket برای browser clients
* Instant message delivery