Akdeniz Üniversitesi CSE Chatbot Projesi - Mimari Tasarım ve Uygulama Rehberi

Projenizi inceledim ve sizin için kapsamlı bir mimari analiz hazırladım. Türkçe topluluğunuz için WhatsApp ve mobil uygulama üzerinden çalışan, birden fazla veri kaynağından (Teams, yemekhane, CSE websitesi) bilgi çeken bir AI chatbot sistemi tasarlamak istiyorsunuz.

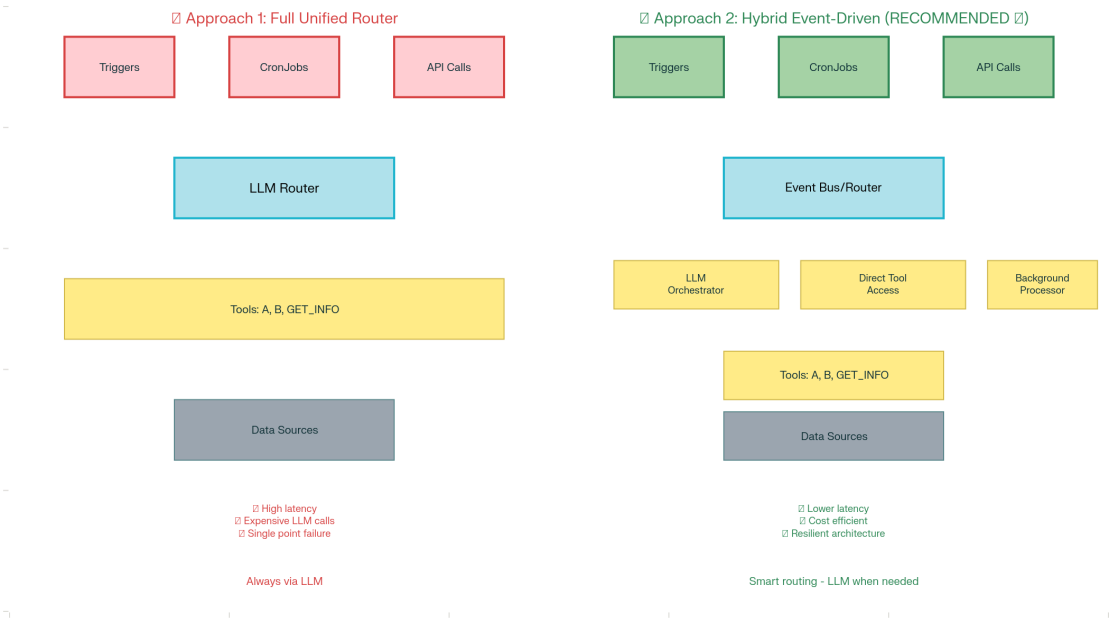
Önerilen Mimari: Hybrid Event-Driven Architecture

Aklınıza gelen iki yaklaşımı analiz ettikten sonra, **Hybrid Event-Driven Architecture** (Hibrit Olay-Güdümlü Mimari) öneriyorum. Bu yaklaşım, her iki yaklaşımınızın güçlü yönlerini birleştirir:

Neden Bu Mimari?

Unified Router'ın basitliği ile **Event-Driven**'ın esnekliğini birleştirir. Şöyle çalışır:

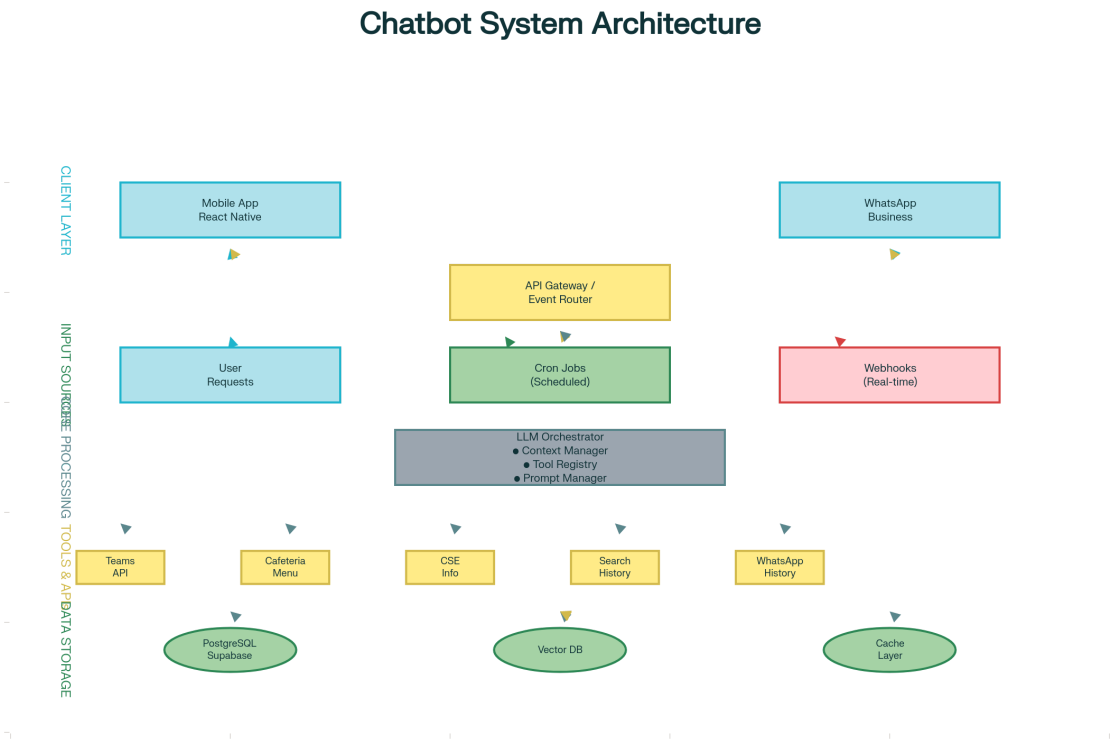
1. **Tüm istekler tek bir Event Router'dan geçer** (trigger, cronjob, API call)
2. Event Router, **akıllı yönlendirme** yapar:
    - Karmaşık sorular → LLM Orchestrator
    - Basit veri çekme → Direkt tool'a
    - Background işler → Queue'ya
3. **LLM her zaman aynı tool sistemini kullanır** - tutarlılık sağlanır
4. **Maliyetli LLM çağrıları optimize edilir** - basit işlemler cache'den



Comparison of Unified Router vs Hybrid Event-Driven architecture, showing the recommended hybrid approach that balances simplicity with intelligent routing

Sistem Mimarisi

İşte projeniz için tasarladığım komple sistem mimarisi:



Comprehensive chatbot system architecture showing event-driven design with multiple data sources, LLM orchestration, and tool calling pattern

Temel Componentler

Sistem 8 ana katmandan oluşur:

1. **Client Layer (Kullanıcı Katmanı)**: Mobil app (React Native/Expo) ve WhatsApp Business API entegrasyonu
2. **API Gateway**: Tüm isteklerin tek giriş noktası, authentication ve rate limiting
3. **Data Source Connectors**: Teams, yemekhane, CSE websitesi için özel connector'lar
4. **LLM Orchestration Layer**: Sistemin beyni - tool seçimi, context yönetimi
5. **Tool System**: Her veri kaynağı için özel fonksiyonlar (GET_TEAMS_MESSAGES, GET_CAFETERIA_MENU vb.)
6. **Knowledge Base**: Vector database ile semantic search ve RAG
7. **Scheduler & Trigger Manager**: Cron job'lar ve webhook listener'lar
8. **Database Layer**: Supabase PostgreSQL + Vector + Redis cache

Senaryolarınızın Çözümleri

Verdiğiniz senaryolar için detaylı veri akışları:

Örnek 1: "Opsys'te en son nereye kadar işlemiştik? @Ikbal"

Akış:

```
Kullanıcı → API Gateway → LLM Orchestrator
→ LLM analiz eder: Teams mesajları lazım
→ GET_TEAMS_MESSAGES(topic="Opsys", user_mention="Ikbal")
→ Vector DB'de semantic search
→ İlgili mesajları bulur
→ LLM yanıtı formatlar: "Shared memory system sanırım. Ikbal'e göre..."
→ Kaynak belirtir: [Teams - CSE3 kanalı, 2 gün önce]
```

Örnek 2: "Bugün yemekte ne var?"

Akış:

```
Kullanıcı → LLM → GET_CAFETERIA_MENU(date=today)
→ Tool önce cache'e bakar (sabah 08:00'da cronjob çalıştı)
→ Cache'de bulur, direkt döner
→ LLM formatlar: "Bugün yemekte: Mercimek çorbası, ..."
```

Örnek 3: Hoca Teams'den mesaj atınca (Real-time)

Akış:

```
Teams Webhook → Event Router → Validate
→ DB'ye kaydet + Vector DB'ye embed et
→ LLM'e gönder: "Bu önemli mi?"
→ LLM: "Evet, final duyurusu"
→ Broadcast servisi → Mobil + WhatsApp kullanıcılarına bildirim
```

Tool Sistemi

Her veri kaynağı için özel tool'lar tasarladım:

Tool Calling Pattern: Modern LLM'ler (GPT-4, Claude) function calling destekler. LLM, kullanıcı sorusuna göre hangi tool'u hangi parametrelerle çağıracağına otomatik karar verir.[1] [2] [3] [4] [5]

Tool Tasarım Prensipleri

1. **Açık ve net tool tanımları**: LLM'in doğru tool'u seçmesi için[5]
2. **Minimal parametreler**: Sadece gerekli parametreler[5]
3. **İyi örnekler**: Her tool için kullanım örnekleri
4. **Hata yönetimi**: Graceful fallback'ler

İmplementasyon Planı

10 haftalık aşamalı bir plan hazırladım:

Kritik nokta: Her fase çalışır bir sistem üretir. İlk 2 haftada basit chatbot çalışır halde olur, sonraki faseler özellik ekler.[6] [7]

Teknik Zorluklar ve Çözümler

Karşılaşabileceğiniz 16 teknik zorluğu ve çözümlerini derledim:

En Kritik 3 Nokta

1. Context Management: Çoklu kanal (WhatsApp + Mobil) arasında tutarlılık[8] [9] [10]

- **Çözüm**: Tek user ID, shared conversation DB, real-time sync

2. LLM Maliyet Optimizasyonu: Her sorguya LLM çağrısı pahalı[6] [11]

- **Çözüm**: Cache mekanizması, basit sorular için ucuz model (GPT-3.5), karmaşık için GPT-4

3. Webhook Güvenilirliği: Teams webhook'ları bazen fail eder[12] [13]

- **Çözüm**: Retry logic + fallback polling her 5-10 dakika

Örnek Kod İmplementasyonu

Sisteminiz için detaylı pseudo-code hazırladım:

Bu kod şunları gösterir:

- Event-driven architecture pattern[14] [15] [16]
- LLM orchestration layer tasarımı[17] [6] [7]
- Tool calling implementasyonu[1] [2] [3]
- Context management sistemi[8] [10]
- Webhook ve cron job handler'ları[12] [13]

Mimari Karşılaştırması

Farklı yaklaşımların detaylı karşılaştırması:

Önerim: Hybrid Event-Driven, sizin use case'iniz için ideal çünkü:[14] [4] [18]

- **Basit başlar**: İlk etapta unified router gibi
- **Sonra scale eder**: Yük arttıkça event-driven'a geçiş kolay
- **Maliyet optimize**: LLM sadece gerektiğinde
- **Yeni kaynak eklemek kolay**: Mikroservis mantığı[15] [19]

Teknoloji Stack Önerisi

Mevcut tecrübenize (Supabase, React Native, Node.js) göre[user profile]:

Backend:

- **Framework**: Express.js veya Fastify (bildiğiniz Node.js)
- **LLM**: OpenAI GPT-4 veya Anthropic Claude
- **Orchestration**: LangChain veya custom (başlangıç için custom daha iyi)[17] [6]

Database:

- **Ana DB**: Supabase PostgreSQL (zaten kullanıyorsunuz)[user profile]
- **Vector DB**: Supabase Vector (pgvector extension)[20] [21]
- **Cache**: Redis (BullMQ için de lazım)

Frontend:

- **Mobile**: Expo/React Native (tecrübeniz var)[user profile]
- **WhatsApp**: WhatsApp Business API[22] [23] [24]

Automation:

- **Cron**: Supabase pg_cron veya node-cron[12] [25]
- **Queue**: BullMQ (Redis tabanlı)[14] [13]
- **Webhooks**: Express endpoints + validation[13] [26] [12]

Data Flow Stratejileri

Trigger vs Cronjob vs API Call - Ne Zaman Hangisi?

Triggers (Webhooks) - Real-time olaylar:[12] [13] [26]

- ✅ Teams mesajları (anında bildirim)
- ✅ Kritik duyurular
- ⚠️ Rate limiting gerekir
- ⚠️ Retry logic şart

Cronjobs - Scheduled görevler:[13] [27] [25]

- ✅ Yemekhane menüsü (günlük 08:00)
- ✅ CSE duyuruları (saatlik kontrol)
- ⚠️ Timezone dikkat (UTC kullan)
- ⚠️ Job overlap önle

API Calls - Kullanıcı istekleri:

- ✅ Chatbot sorguları

- ✅ On-demand veri çekme
- ⚠️ Rate limiting
- ⚠️ Authentication

RAG ve Context Yönetimi

Retrieval-Augmented Generation (RAG) sisteminiz için:[20] [9] [21]

RAG Mimarisi

1. **Indexing**: Tüm mesajları vector DB'ye embed et
2. **Retrieval**: Kullanıcı sorusuna göre ilgili mesajları bul
3. **Generation**: LLM'e context ile birlikte gönder
4. **Response**: Kaynak belirterek yanıtla

Context Window Stratejisi

Sliding Window: Son 10-15 mesajı context olarak tut[8] [10]

- Daha eski mesajlar için vector search
- Token limiti aşmamak için önemli[20] [10]
- Her kullanıcı için ayrı context[8]

Contextual Retrieval: Anthropic'in önerdiği yöntem[20]

- Her mesaja metadata ekle (kim, ne zaman, hangi kanal)
- Semantic search doğruluğunu %49 artırır[20]

Ölçeklenebilirlik ve Performans

Queue-Based Processing

BullMQ ile iş kuyruğu:[14] [13]

```
// Webhook geldiğinde direkt işlemek yerine queue'ya at
await messageQueue.add('process-teams-message', {
  message: webhookData,
  priority: isImportant ? 1 : 5
});

// Worker ayrı process'te çalışır
messageQueue.process(async (job) => {
  await processTeamsMessage(job.data);
});
```

Avantajları:

- Spike'larda sistem çökmez[28] [14]

- Retry otomatik[13] [14]
- Priority queue ile önemli mesajlar önce[29]

Caching Stratejisi

3 Katmanlı Cache:

1. **Memory Cache**: Çok sık sorulanlar (yemekhane menüsü)
2. **Redis Cache**: API sonuçları, LLM yanıtları
3. **DB Cache**: Vector embeddings

TTL Önerileri:

- Yemekhane menüsü: 24 saat
- Teams mesajları: 1 saat
- LLM yanıtları: Benzeri sorular için 1 saat

Güvenlik ve Privacy

GDPR ve Veri Koruma

Kritik noktalar:[4]

1. **User Consent**: WhatsApp mesajlarını indexlemeden önce izin al
2. **Data Retention**: Mesajları maksimum 90 gün sakla
3. **RLS (Row Level Security)**: Supabase RLS ile her kullanıcı sadece kendi verisini görsün[user profile]
4. **Encryption**: Supabase encryption at rest zaten var

Webhook Security

Teams webhook'larını validate et:[12] [13]

```
function validateWebhook(req) {
  const signature = req.headers['x-ms-signature'];
  const secret = process.env.TEAMS_WEBHOOK_SECRET;
  const hash = crypto.createHmac('sha256', secret)
    .update(JSON.stringify(req.body))
    .digest('hex');
  return signature === hash;
}
```

Monitoring ve Debug

LLM Decision Tracking

Her LLM çağrısını logla:[27] [30]

```
await db.llm_logs.insert({
  user_query: query,
  tools_called: toolCalls.map(t => t.name),
  response: answer,
  cost: calculateCost(response),
  latency: responseTime
});
```

Faydaları:

- Neden yanlış yanıt verdi anlaşılır[27]
- Maliyet tracking
- A/B testing için data

Error Monitoring

Sentry veya benzeri tool kullan:[27]

- LLM API errors
- Webhook failures
- Scraping errors
- User-facing errors

Mobil App Özellikleri

Admin Panel (Sizin Use Case'inizden)

Mobil appte özel admin sekmesi[user profile]:

```
Admin Panel:
├── Tool Management
│   ├── Enable/Disable Tools
│   ├── Usage Stats per Tool
│   └── Cost per Tool
├── Data Sources
│   ├── Teams Channels (subscribe/unsubscribe)
│   ├── Notification Settings
│   └── Scraper Status
└── Analytics
    ├── Popular Queries
    ├── User Activity
    └── System Health
```

User Preferences

Her kullanıcı customize edebilir:

- Hangi kanallardan bildirim istiyorum?
- Yemekhane menüsü her gün gelsin mi?
- Hangi Teams kanallarını takip edeyim?

Gelecek İyileştirmeler

Faz 2 Özellikleri (İlk launch sonrası)

1. **Voice Support**: WhatsApp voice message → STT → LLM → TTS
2. **Multi-Intent Handling**: "Email güncelle VE TV servisine kayıt ol" gibi çoklu görevler[31]
3. **Proactive Notifications**: "Yarın final var, hazırlan!" gibi akıllı hatırlatmalar
4. **Multi-Language**: Erasmus öğrenciler için İngilizce destek
5. **Analytics Dashboard**: Admin için detaylı dashboard

Ölçeklenme Senaryoları

100+ kullanıcı: Mevcut mimari yeterli
1000+ kullanıcı: Redis cluster, DB replication
10000+ kullanıcı: Kubernetes, microservices'e geçiş, multiple LLM instances

Maliyet Analizi

Tahmini Aylık Maliyetler (100 kullanıcı için)

LLM API (GPT-4):

- Günlük 10 sorgu/kullanıcı × 100 kullanıcı = 1000 query/day
- Ortalama 500 token/query
- ~$30-50/ay

Supabase (Pro plan):

- Database + Vector DB
- ~$25/ay

WhatsApp Business API:

- Konuşma başına ücret (Türkiye: ~$0.01-0.03)
- Aylık ~$20-30

Diğer (Redis, hosting):

- ~$20/ay

Toplam: ~$95-125/ay (100 kullanıcı için oldukça makul)

Optimizasyon İpuçları

1. **Prompt Caching**: Anthropic'de %90 maliyet düşüşü[20]
2. **Smart Model Selection**: Basit sorular GPT-3.5, karmaşık GPT-4
3. **Cache Aggressive**: Aynı soruyu tekrar LLM'e göndermeme
4. **Batch Processing**: Cron job'ları tek seferde çalıştır

Sonuç ve Tavsiyeler

Kısa Vadede (İlk 2 Hafta)

1. ✅ Supabase projesini kur
2. ✅ Basic Express API + OpenAI entegrasyonu
3. ✅ İlk tool'u yap: GET_CAFETERIA_MENU
4. ✅ Mobil appte basit chat UI
5. ✅ Test et, iterate et

Orta Vadede (2-6 Hafta)

6. ✅ Teams webhook entegrasyonu
7. ✅ Vector DB + RAG sistemi
8. ✅ Tüm tool'ları implement et
9. ✅ Context management
10. ✅ WhatsApp entegrasyonu

Uzun Vadede (6-10 Hafta)

11. ✅ Production deployment
12. ✅ Monitoring ve logging
13. ✅ User testing ve feedback
14. ✅ Admin panel
15. ✅ Documentation

En Önemli 3 Tavsiyem

1. Basit Başla, İterative İlerle
İlk hafta sadece yemekhane menüsü çalışır halde olsun. Her hafta bir özellik ekle. "Big bang" approach yerine incremental.[6] [7]

2. Tool Descriptions'a Zaman Harca
LLM'in başarısı %80 tool descriptions'ın kalitesine bağlı. Test et, refine et, örneklerle zenginleştir.
[5]

[5]

3. Monitoring'i İhmal Etme
İlk günden logla. Neden yanlış yanıt verdi anlamak için şart.[27] [30]

Özet Döküman

Tüm önerileri Türkçe özet olarak derledim:

Başarılar dilerim! Projeniz harika bir fikir ve doğru mimari ile çok işlevsel bir sistem çıkacak. Sorularınız olursa yardımcı olmaktan mutluluk duyarım

⁂

1. https://langchain-ai.github.io/langgraph/concepts/agentic_concepts/
2. https://martinfowler.com/articles/function-call-LLM.html
3. https://blog.dagworks.io/p/agentic-design-pattern-1-tool-calling
4. https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-patterns/tool-based-agents-for-calling-functions.html
5. https://www.braintrust.dev/blog/agent-while-loop
6. https://orq.ai/blog/llm-orchestration
7. https://www.iguazio.com/glossary/llm-orchestration/
8. https://www.chitika.com/strategies-handling-long-chat-rag/
9. https://kairntech.com/blog/articles/rag-conversational-ai-the-complete-guide-to-building-advanced-ai-chatbots/
10. https://towardsdatascience.com/why-context-is-the-new-currency-in-ai-from-rag-to-context-engineering/
11. https://www.scoutos.com/blog/llm-orchestration-key-tactics-and-tools
12. https://docs.openfn.org/documentation/build/triggers
13. https://dev.to/public_cloud/beyond-the-cron-job-advanced-triggering-strategies-for-modern-workflows-1p3f
14. https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-serverless/event-driven-architecture.html
15. https://www.confluent.io/learn/event-driven-architecture/
16. https://microservices.io/patterns/data/event-driven-architecture.html
17. https://www.ibm.com/think/topics/llm-orchestration
18. https://docs.databricks.com/aws/en/generative-ai/guide/agent-system-design-patterns
19. https://www.geeksforgeeks.org/system-design/event-driven-apis-in-microservice-architectures/
20. https://anthropic.com/news/contextual-retrieval
21. https://coralogix.com/ai-blog/how-to-build-a-rag-chatbot-from-scratch-with-minimal-ai-hallucinations/
22. https://www.appypieautomate.ai/integrate/apps/microsoft-teams/integrations/whatsappbusiness
23. https://www.linkmobility.com/blog/how-to-integrate-a-chatbot-into-whatsapp-business-api
24. https://www.socialintents.com/whatsapp-chatbot.jsp

25. https://spinnaker.io/docs/guides/user/pipeline/triggers/
26. https://hookdeck.com/webhooks/guides/when-to-use-webhooks
27. https://www.linkedin.com/posts/breon-reed-813179273_umbrella-topic-building-scaling-a-saas-activity-7358595176375668738-xQ0E
28. https://aws.amazon.com/event-driven-architecture/
29. https://gettalkative.com/info/chat-routing
30. https://latitude-blog.ghost.io/blog/5-patterns-for-scalable-llm-service-integration/
31. https://developers.liveperson.com/conversation-builder-generative-ai-routing-ai-agents-route-consumers-conversationally.html
32. https://www.teknolojioku.com/sosyal-medya/whatsapp-kullanicilari-tepkili-bu-hata-isyan-ettirdi-6630a9e5f187d330b00d94c7
33. https://www.birgun.net/haber/whatsapp-geri-adim-atti-iddiasi-tartisma-yaratan-gizlilik-politikasi-istege-bagli-olacak-356977
34. https://www.ajeetraina.com/inside-the-black-box-how-llm-neural-layers-make-tool-calling-decisions/
35. https://www.systemdesignhandbook.com/guides/chatbot-system-design-interview/
36. https://marutitech.com/ai-powered-event-driven-architecture/
37. https://www.linkedin.com/pulse/system-design-modern-generative-ai-chatbot-akash-srivastava-35hjc
38. https://telnyx.com/resources/architecture-chatbot-development
39. https://aws.amazon.com/blogs/security/hardening-the-rag-chatbot-architecture-powered-by-amazon-bedrock-blueprint-for-secure-design-and-anti-pattern-migration/
40. https://yourgpt.ai/blog/general/train-ai-chatbot-on-my-data
41. https://customgpt.ai/data-integration/
42. https://www.useparagon.com/learn/ai-knowledge-chatbot-chapter-1/
43. https://quidget.ai/blog/ai-automation/12-data-sources-that-will-make-your-ai-chatbot-smarter-beyond-faqs/
44. https://www.leapxpert.com/how-to-use-microsoft-teams-and-whatsapp-for-better-communication-and-collaboration/
45. https://www.searchunify.com/resource-center/sudo-technical-blogs/best-practices-for-using-retrieval-augmented-generation-rag-in-ai-chatbots
46. https://www.reddit.com/r/LangChain/comments/1izxr8w/how_can_i_build_a_chatbot_that_retrieves/
47. https://www.reddit.com/r/automation/comments/1joxri0/easiest_way_to_set_up_a_chatbot_for_whatsapp/
48. https://www.acceldata.io/blog/how-rag-in-ai-is-transforming-conversational-ai
49. https://www.acceldata.io/blog/orchestration-layer-explained-functions-tools-and-best-practices
50. https://botpress.com/blog/sms-chatbot
51. https://www.infobip.com/glossary/llm-orchestration
52. https://wiki.bicomsystems.com/PBXware/PBXware-7-CC-Administration-Manual/Chatbots
53. https://www.2501.ai/research/llm-orchestration-enterprise
54. https://getstream.io/glossary/chatbot/
55. https://solace.com/event-driven-architecture-patterns/
56. https://www.ibm.com/architectures/papers/rag-cookbook/orchestration