UniClub Projesi Mimari Özeti

1) Teknoloji Yığını

Frontend: - Düz **HTML + Vanilla JS (fetch API)** - Basit statik sunucu (Python `http.server` veya Live Server)

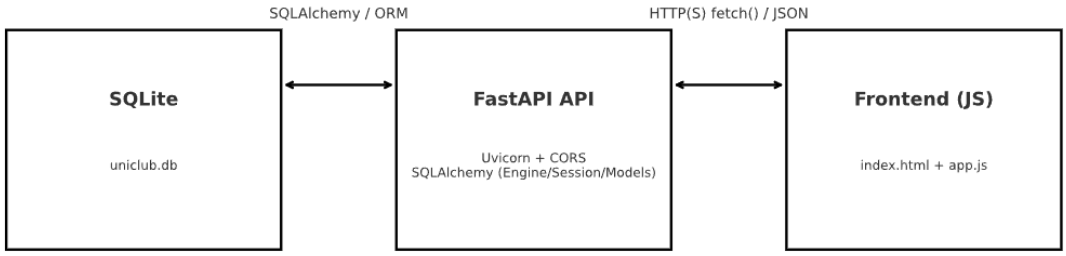
Backend: - **FastAPI** (REST API) - **Uvicorn** (ASGI server) - **SQLAlchemy** (ORM) - **CORS Middleware** (geliştirme için `allow_origins=["*"]`)

Veritabanı: - **SQLite** (dosya: `data_access_layer/database/uniclub.db`)

Katmanlar: - `business_layer`: `SystemMessageService` (in-memory örnek servis) - `data_access_layer`: `db.py` (Engine/Session/Base), `models.py` (Club/Member/Gelistirme) - `webAPI_layer`: `main.py` (FastAPI uygulaması ve endpointler)

---

2) Veri Akış Diyagramı (DB → API → UI)



Akış: 1. Frontend `fetch(API_URL/...)` ile HTTP istek atar. 2. FastAPI, `SessionLocal` ile DB oturumu açar; `models.py` üzerinden sorgu/insert yapar. 3. Yanıt JSON olarak frontend'e döner; DOM'a yazılır (`app.js`).

---

3) Endpointler ve Örnek JSON

`GET /`

200 OK

```
{ "message": "UniClub API çalışıyor 🚀" }
```

`GET /gelistirme`

200 OK

```
{ "description": "Bu site geliştirme aşamasındadır." }
```

(Hiç kayıt yoksa)

```
{ "description": "Henüz içerik yok." }
```

```
GET /clubs
```

200 OK (liste)

```
[
  { "id": 1, "name": "Bilgisayar Mühendisliği Kulübü", "description":
"Teknoloji ve programlama etkinlikleri" },
  { "id": 2, "name": "Fotoğrafçılık Kulübü", "description": "Fotoğraf çekimi
ve sanat etkinlikleri" }
]
```

```
POST /clubs
```

Açıklama: Şu an **query/form parametre** bekler. - Parametreler: `name` (zorunlu), `description` (opsiyonel)

Örnek İstek:

```
POST /clubs?name=Satranç%20Kulübü&description=Turnuvalar
```

200 OK

```
{ "message": "Kulüp eklendi ☑", "club": "Satranç Kulübü" }
```

Alternatif (öneri) – JSON body ile:

```python
from pydantic import BaseModel
class ClubCreate(BaseModel):
    name: str
    description: str | None = None

@app.post("/clubs")
def create_club(payload: ClubCreate, database: Session = Depends(get_db)):
    new_club = models.Club(name=payload.name, description=payload.description
or "")
    database.add(new_club)
    database.commit()
    database.refresh(new_club)
    return {"message": "Kulüp eklendi ☑", "club": new_club.name}
```

JSON Örneği:

```
{ "name": "Satranç Kulübü", "description": "Turnuvalar" }
```

```
POST /clubs/sample-data
```

Örnek kulüpleri ve bir adet geliştirme mesajını ekler. **200 OK**

{ "message": "5 örnek kulüp eklendi ☑", "added_clubs": ["Bilgisayar Mühendisliği Kulübü", "Fotoğrafçılık Kulübü", "Müzik Kulübü", "Spor Kulübü", "Tiyatro Kulübü"] }

---

4) Genişleme / İyileştirme Notları

- **Business Layer** kullanımını genişletme (iş kuralları, validasyonlar)

- **Sprint bazlı geliştirme planı ve dokümantasyon**
  - Her sprint için hedefler
  - Sprint sonunda: değişen endpointler, veri modeli ve UI akışları için kısa belgelendirme
- **API genişlemesi (endpoint ekleme)**
  - POST /members (üye oluştur), GET /members?clubId=… (kulüp üyeleri)
  - PUT /clubs/{id}, DELETE /clubs/{id} (tam CRUD)
  - Filtreleme/sıralama: GET /clubs?q=…&sort=name&page=1&pageSize=20
  - Versiyonlama: /api/v1/… → ileride /api/v2 geçiş stratejisi
- **Arayüz (UI) geliştirmeleri**
  - Kulüp oluşturma/arama/listeleme için formlar, boş durum (empty state) ve hata mesajları
- **Rol/izin yönetimi: admin**, **kulüp yöneticisi**, **üye** gibi rollere göre endpoint koruması
- **Parola güvenliği (şifreleri modern algoritmalarla saklama)**
  - **bcrypt** ile güçlü hash; salt ve uygun cost parametreleri
- **Rol bazlı farklı arayüzler**
  - Admin paneli (kulüp/üye yönetimi), kulüp yöneticisi paneli (etkinlik, başvuru), üye görünümü (liste, başvuru)
- **Hata/İstisna yönetimi ve standart yanıt formatı**
- **Dokümantasyonun sürekli güncellenmesi**
  - OpenAPI/Swagger’den **docs/** altına otomatik statik doküman üretimi
  - Değişikliklerde README ve Mimari Özeti’nin güncellenmesi; “Breaking Changes” bölümü