Internia Teknoloji — News Platform Kılavuz (API + Frontend)

Bu doküman **Internia Teknoloji** için geliştirilen Haber Platformu'nun iki ana bileşenini açıklamak adına hazırlanmıştır.

1. **API (Backend)** — Nasıl çalışır, mimari, güvenlik, konfigürasyon ve endpoint'ler.
2. **Frontend (React / Vite)** — Uygulama akışı, oturum yönetimi, kullanılan endpoint'ler ve entegrasyon notları

---

1) BACKEND APIs

1.1 Amaç ve Genel Bakış

News API, haber içeriklerinin güvenli şekilde yayınlanması için geliştirilmiş bir **.NET 8 Minimal API** uygulamasıdır. Veriler **MongoDB**'de tutulur. Kimlik doğrulama **JWT** ile yapılır; token'lar **HttpOnly Cookie** olarak set edilir. Yönetim (create/update/delete) işlemleri yalnızca **Admin** rolündeki kullanıcıyla yapılır; **okuma** (GET) herkese açıktır.

1.2 Mimarinin Kısa Özeti

```
Client (Browser/Frontend)
  └─(fetch + credentials: 'include')→  News API (.NET 8)
        ├─ AuthEndpoints  → Login/Logout (JWT üretimi, HttpOnly cookie)
        └─ NewsEndpoints  → Haber CRUD + Seed
                              └─ Mongo Repository → MongoDB
```

- **Program.cs**: CORS, Authentication/Authorization, Swagger, endpoint grupları
- **AuthEndpoints.cs**: `/auth/login`, `/auth/logout` (JWT üretim/silme)
- **NewsEndpoints.cs**: `/api/news` CRUD ve `/api/news/seed`
- **JWT Doğrulama**: Cookie'deki token middleware tarafından doğrulanır

1.3 Güvenlik Modeli

- **HttpOnly Cookie**: Token JS ile okunamaz → XSS riskini azaltır.

- **CORS + Credentials**: Frontend domain'i izinli origin olarak tanımlanır; istekler `credentials: 'include'` ile yapılır.
- **HTTPS (Prod)**: Cookie `Secure=true` ile set edilir; `SameSite` senaryoya uygun yapılandırılır.
- **Rol Bazlı Yetki**: Admin gereken uçlar `[Authorize(Roles=\"Admin\")]` ile korunur.

1.4 Konfigürasyon

`appsettings.json` (örnek):

```
{
  "Mongo": {
    "ConnectionString": "mongodb://localhost:27017", -> MONGO ATLAS BAĞLA
    "Database": "newsdb"
  },
  "Jwt": {
    "Issuer": "news-api",
    "Audience": "news-app",
    "Key": "jwt key", -> JWT SECRET ANAHTAR
    "ExpiresMinutes": 60
  }
}
```

1.5 Uç Noktalar (Endpoint'ler)

Metod	Yol	Açıklama	Yetki
GET	`/api/news?limit=`	Haberleri listele (varsayılan 50)	Public
POST	`/api/news`	Haber oluştur	Admin
PUT	`/api/news/{id}`	Haber güncelle	Admin
DELETE	`/api/news/{id}`	Haberi sil	Admin
DELETE	`/api/news`	Tüm haberleri sil	Admin
POST	`/api/news/seed`	Örnek veri ekle	Admin
POST	`/auth/login`	Giriş yap → cookie set edilir	Public
POST	`/auth/logout`	Çıkış yap → cookie temizlenir	Public

Örnek `POST /api/news` gövdesi:

```
{
  "title": "Başlık",
  "description": "Açıklama",
  "publishedAt": "2025-08-11T09:00:00Z"
}
```

1.6 Örnek Etkileşim Senaryosu (cURL)

```
# 1) Login olup cookie yakala
curl -c cookies.txt -X POST http://localhost:5000/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"12345"}'

# 2) Admin endpointlere cookie ile eriş
curl -b cookies.txt -X POST http://localhost:5000/api/news/seed
curl -b cookies.txt -X POST http://localhost:5000/api/news \
  -H "Content-Type: application/json" \
  -d '{"title":"Deneme","description":"Açıklama","publishedAt":"2025-08-1

# 3) Public listeleme (cookie gerekmez)
curl http://localhost:5000/api/news?limit=5

# 4) Çıkış
curl -b cookies.txt -X POST http://localhost:5000/auth/logout
```

---

2) Frontend (React / Vite)

Frontend, **React** ve **Vite** teknolojileri kullanılarak geliştirilmiş olup, tamamen **Single Page Application (SPA)** mimarisi ile yapılandırılmıştır. Bu sayede uygulama, istemci tarafında tek bir HTML sayfası üzerinde çalışarak, sayfa yenilemesine gerek kalmadan dinamik içerik güncellemeleri yapar ve kullanıcı deneyimini kesintisiz hale getirir.

2.1 Router'sız Etkileşim

Uygulama internal bir `route` state'i ile sayfaları değiştirir: `home` , `login` , `news` , `panel` .

- **Nav**: Panel sekmesi yalnızca `isAuthed` iken görünür.
- **Home**: Son 3 haberi çeker (`/api/news?limit=3`), banner ve kısayollar gösterir.
- **NewsPublic**: Public haber listesi (`/api/news`), giriş gerekmez.

- **Login**: `/auth/login` başarılı olursa UI'da `isAuthed=true` yapılır (cookie JS'te okunmaz, oturum sinyali UI state'tir).
- **Panel**: Admin işlemleri (seed, ekle, güncelle, sil). 401 alınırsa UI **logout** görünümü verilir.

2.4 Oturum Akışı (Login/Logout)

- **Login** (Form → `/auth/login`): 200 gelirse cookie tarayıcıya set edilir → UI `isAuthed=true` → kullanıcı **Panel**'e yönlendirilir.
- **Admin İşlemi**: Panel içindeki çağrılar (POST/PUT/DELETE) cookie ile otomatik yetkilendirilir. 401 yakalanırsa `onUnauthorized()` tetiklenir (UI logout'a döner).
- **Logout** (`/auth/logout`): Cookie temizlenir, UI `isAuthed=false`, Panel görünmez.

2.5 Panel İşlevleri ve Backend Eşleşmesi

- **Liste**: `GET /api/news`
- **Ekle**: `POST /api/news` — body: `{ title, description, publishedAt }`
- **Güncelle**: `PUT /api/news/{id}` — body: `{ title, description }`
- **Sil (tek)**: `DELETE /api/news/{id}`
- **Sil (tümü)**: `DELETE /api/news`
- **Seed**: `POST /api/news/seed`

2.6 UI Davranışları

- **Panel Sekmesi**: Sadece `isAuthed=true` iken gösterilir.
- **401 Yönetimi**: `Error.message 401` ile başlıyorsa toast + login'e yönlendirme.
- **Toast**: Kısa süreli bilgilendirme bileşeni (`useToast`).

2.7 Frontend Geliştirme / Çalıştırma

```
# Bağımlılıkları kur
npm i

# Geliştirme sunucusu 7777. Portta koşuyor
npm run dev

# Üretim derlemesi
npm run build
npm run preview
```
