

- * UI framework: Sade JS + HTML template mi, yoksa Preact/Lit gibi hafif bir şey mi tercih edersiniz?
- * CSS: Tailwind mı, minimal CSS framework mü, yoksa sade CSS mi?
- * Build: Vite + @crxjs/vite-plugin kullanıım mı?
- * UDF dönüştürme: Bunun için ayrı öneriniz olacağını söylediniz – bunu bu eklentiye dahil mi edeceğiz yoksa ayrı mı tutacağız?

Değerlendirdi teknoloji seçeneklerini, kullanıcı profiline uygun öneriler hazırladı

Chrome Extension Tech Stack Önerilerim

1. UI Framework: Preact

Seçenek	Artı	Eksi
Vanilla JS + HTML	Sıfır dependency, en küçük bundle	State yönetimi manuel, karmaşık UI'da spaghetti kod
Preact	3KB, React API uyumlu, hooks desteği	Ekstra build step
Lit	Web components standartı, encapsulation	Öğrenme eğrisi farklı, ekosistem daha küçük

Tercihim: Preact – UYAP extension'ında dosya listesi, seçim state'i, filtreleme gibi reaktif UI ihtiyaçları var. Preact bunu 3KB'de çözüyor ve React bilginle doğrudan kullanabilirsin.

2. CSS: Tailwind (JIT mode)

Stack'inde zaten Tailwind var. Extension'larda:

- Content script için CSS isolation önemli → Tailwind'i prefix ile kullan (uyap-)
- Popup/Options için standart Tailwind yeterli
- JIT mode ile sadece kullanılan class'lar bundle'a girer → ~10KB civarı

js

```
// tailwind.config.js
export default {
  prefix: 'uyap-', // content script isolation
  content: ['./src/**/*.{js,jsx,html}'],
}
```

3. Build: Evet, Vite + @crxjs/vite-plugin

Kesinlikle kullan. Avantajları:

- HMR (Hot Module Reload) → popup/content script'te anında değişiklik görürsün
- Manifest v3 desteği otomatik
- TypeScript, JSX, CSS processing tek config'de
- Dev/prod build ayrımı kolay

bash

```
npm create vite@latest uyap-extension -- --template preact-ts
npm i @crxjs/vite-plugin -D
```

4. UDF Dönüşüm: **Ayrı modül, aynı extension**

Önerim şu mimari:

```
src/
  core/
    udf-converter/      # UDF işlemeleri (ayrı modül)
    |   parser.ts
    |   converter.ts
    |   index.ts
  uyap-client/          # UYAP API işlemeleri
```

```
|--- content/          # Content scripts  
|--- popup/           # Popup UI  
└--- background/     # Service worker
```

Neden ayrı modül ama aynı extension?

- UDF dönüştürme geneldeUYAP bağlamında kullanılıyor → kullanıcı tek extension yönetsin
- Modüler yapıda tutarsan ilerde npm paketi olarak da çıkarabilirsin
- Tree-shaking ile kullanılmayan UDF kodu popup bundle'ına girmez

Özet Stack

Katman	Seçim
UI	React + JSX
CSS	Tailwind (prefixed)
Build	Vite + @crxjs/vite-plugin
Language	TypeScript
UDF	src/core/udf-converter/ modülü

UDF Formatı Hakkında

UDF aslında ODF (OpenDocument) tabanlı – içi XML + ZIP. Temel okuma/yazma için:

```
dosya.udf (ZIP)  
|--- content.xml      # Ana içerik  
|--- styles.xml       # Stiller  
|--- meta.xml         # Metadata  
└--- mimetype        # application/vnd.oasis.opendocument.text
```

Benim Önerim

Hibrit yaklaşım:

1. Okuma/görüntüleme → Kendi parser'ımız (JSZip + XML parse) – ücretsiz, hızlı
2. Düzenleme/e-imza gerekiyorsa → e-udf.com editörüne "Editörde Aç" butonu ile yönlendir

typescript

```
// Basit UDF okuma örneği  
import JSZip from 'jszip';
```

```
async function readUDF(file: File) {  
  const zip = await JSZip.loadAsync(file);  
  const content = await zip.file('content.xml')?.async('string');  
  // XML parse et, içeriği göster  
}
```

Netleştirmem Gereken

1. Asıl ihtiyacın ne?

- Sadece UDF'yi PDF/HTML olarak görüntülemek mi?
- UDF'yi düzenlemek mi?
- UDF'yi e-imzalamak mı?

2. e-udf.com hesabın var mı veya kullanıcıların bu servise zaten abone olduğunu varsayıbilir miyiz?

UDF → PDF Dönüşüm Seçenekleri

Seçenek Karşılaştırması

Yöntem	Maliyet	Entegrasyon	Performans
LibreOffice Headless (self-hosted)	Ücretsiz	Backend gereklili	Hızlı
Gotenberg (Docker)	Ücretsiz	Backend gereklili	Hızlı
CloudConvert API	~0.01\$/dönüşüm	Sadece API key	Çok hızlı
e-udf.com	Aylık ücret	Manuel/belirsiz API	-

Önerim: Gotenberg (Self-hosted, ücretsiz)

LibreOffice tabanlı, Docker ile tek komutta çalışır:

bash

```
docker run --rm -p 3000:3000 gotenberg/gotenberg:8
```

API kullanımı çok basit:

typescript

```
// udf-to-pdf.ts
```

```
async function convertUdfToPdf(udfFile: File): Promise<Blob> {
  const formData = new FormData();
  formData.append('files', udfFile, 'document.odt'); // UDF = ODF uyumlu
```

```
const response = await fetch('http://localhost:3000/forms/libreoffice/convert', {
  method: 'POST',
  body: formData,
});
```

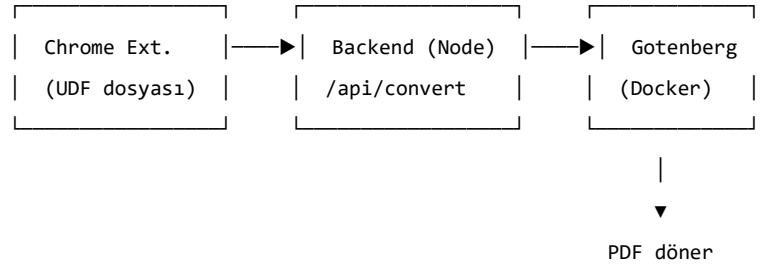
```
return response.blob(); // PDF döner
```

```
}
```

```
...
```

Mimari Öneri

```
...
```



Hızlı Karar

Zaten bir backend'in var mı? (Node/Python/.NET)

- Evet → Gotenberg Docker ekle, 5 dakikada çalışır
- Hayır → Basit bir Node/Express API + Gotenberg kurabiliriz
- Sunucu istemiyorum → CloudConvert API (ücretli ama sunucusuz)

Backend API + Gotenberg Implementasyonu

1. Proje Yapısı

```
uyap-converter-backend/
├── package.json
├── tsconfig.json
├── docker-compose.yml
└── src/
    └── index.ts
```

```
|   └── routes/
|   |   └── convert.ts
|   └── services/
|       └── gotenberg.service.ts
```

2. Docker Compose (Gotenberg + API birlikte)

yaml

```
# docker-compose.yml
```

```
version: '3.8'
```

```
services:
```

```
gotenberg:
```

```
  image: gotenberg/gotenberg:8
```

```
  restart: unless-stopped
```

```
  ports:
```

```
    - "3000:3000"
```

```
  command:
```

```
    - "gotenberg"
```

```
    - "--api-timeout=120s"
```

```
    - "--libreoffice-disable-routes=false"
```

```
api:
```

```
  build: .
```

```
  restart: unless-stopped
```

```
  ports:
```

```
    - "3001:3001"
```

```
  environment:
```

```
    - GOTENBERG_URL=http://gotenberg:3000
```

```
    - PORT=3001
```

```
  depends_on:
```

```
    - gotenberg
```

3. Package.json

json

```
{
```

```
  "name": "uyap-converter-backend",
```

```
  "version": "1.0.0",
```

```
  "type": "module",
```

```
  "scripts": {
```

```
    "dev": "tsx watch src/index.ts",
```

```
    "build": "tsc",
```

```
    "start": "node dist/index.js"
```

```
},
```

```
  "dependencies": {
```

```
    "express": "^4.18.2",
```

```
    "cors": "^2.8.5",
```

```
    "multer": "^1.4.5-lts.1",
```

```
    "form-data": "^4.0.0",
```

```
    "node-fetch": "^3.3.2"
```

```
},
```

```
"devDependencies": {
    "@types/express": "^4.17.21",
    "@types/cors": "^2.8.17",
    "@types/multer": "^1.4.11",
    "@types/node": "^20.10.0",
    "tsx": "^4.6.2",
    "typescript": "^5.3.2"
}
}
```

4. TypeScript Config

```
json
{
    "compilerOptions": {
        "target": "ES2022",
        "module": "NodeNext",
        "moduleResolution": "NodeNext",
        "outDir": "./dist",
        "rootDir": "./src",
        "strict": true,
        "esModuleInterop": true,
        "skipLibCheck": true,
        "declaration": true
    },
    "include": ["src/**/*"],
    "exclude": ["node_modules", "dist"]
}
```

5. Gotenberg Service

```
typescript
// src/services/gotenberg.service.ts
import FormData from 'form-data';
import fetch from 'node-fetch';

const GOTENBERG_URL = process.env.GOTENBERG_URL || 'http://localhost:3000';

export interface ConversionResult {
    success: boolean;
    data?: Buffer;
    error?: string;
    filename?: string;
}

export async function convertUdfToPdf(
    fileBuffer: Buffer,
    originalFilename: string
): Promise<ConversionResult> {
    try {
        const formData = new FormData();
```

```

// UDF = ODF uyumlulu, .odt olarak gönderiyoruz
const convertedFilename = originalFilename.replace(/\.\udf$/i, '.odt');
formData.append('files', fileBuffer, {
  filename: convertedFilename,
  contentType: 'application/vnd.oasis.opendocument.text',
});

const response = await fetch(
  `${GOTENBERG_URL}/forms/libreoffice/convert`,
  {
    method: 'POST',
    body: formData,
    headers: formData.getHeaders(),
  }
);

if (!response.ok) {
  const errorText = await response.text();
  return {
    success: false,
    error: `Gotenberg error: ${response.status} - ${errorText}`,
  };
}

const pdfBuffer = Buffer.from(await response.arrayBuffer());
const pdfFilename = originalFilename.replace(/\.\udf$/i, '.pdf');

return {
  success: true,
  data: pdfBuffer,
  filename: pdfFilename,
};
} catch (error) {
  return {
    success: false,
    error: error instanceof Error ? error.message : 'Unknown error',
  };
}
}

export async function checkGotenbergHealth(): Promise<boolean> {
  try {
    const response = await fetch(`${GOTENBERG_URL}/health`);
    return response.ok;
  } catch {
    return false;
  }
}

```

6. Convert Route

```
typescript
// src/routes/convert.ts

import { Router, Request, Response } from 'express';
import multer from 'multer';
import { convertUdfToPdf, checkGotenbergHealth } from '../services/gotenberg.service.js';

const router = Router();

// Memory storage - dosyaları RAM'de tut (küçük dosyalar için ideal)
const upload = multer({
  storage: multer.memoryStorage(),
  limits: {
    fileSize: 50 * 1024 * 1024, // 50MB max
  },
  fileFilter: (_req, file, cb) => {
    if (file.originalname.toLowerCase().endsWith('.udf')) {
      cb(null, true);
    } else {
      cb(new Error('Sadece .udf dosyaları kabul edilir'));
    }
  },
});

// POST /api/convert - Tekil dosya dönüştürme
router.post('/', upload.single('file'), async (req: Request, res: Response) => {
  try {
    if (!req.file) {
      res.status(400).json({ error: 'Dosya yüklenmedi' });
      return;
    }

    const result = await convertUdfToPdf(req.file.buffer, req.file.originalname);

    if (!result.success || !result.data) {
      res.status(500).json({ error: result.error || 'Dönüştürme başarısız' });
      return;
    }

    res.setHeader('Content-Type', 'application/pdf');
    res.setHeader(
      'Content-Disposition',
      `attachment; filename="${encodeURIComponent(result.filename || 'document.pdf')}"`);
    res.send(result.data);
  } catch (error) {
    res.status(500).json({
      error: error instanceof Error ? error.message : 'Sunucu hatası',
    });
  }
});
```

```

// POST /api/convert/batch - Toplu dönüştürme
router.post('/batch', upload.array('files', 20), async (req: Request, res: Response) => {
  try {
    const files = req.files as Express.Multer.File[];

    if (!files || files.length === 0) {
      res.status(400).json({ error: 'Dosya yüklenmedi' });
      return;
    }

    const results = await Promise.all(
      files.map(async (file) => {
        const result = await convertUdfToPdf(file.buffer, file.originalname);
        return {
          originalName: file.originalname,
          success: result.success,
          pdfBase64: result.success && result.data
            ? result.data.toString('base64')
            : null,
          pdfFilename: result.filename,
          error: result.error,
        };
      });
    );
  });

  res.json({ results });
} catch (error) {
  res.status(500).json({
    error: error instanceof Error ? error.message : 'Sunucu hatası',
  });
}
});

// GET /api/convert/health - Servis durumu
router.get('/health', async (_req: Request, res: Response) => {
  const gotenbergOk = await checkGutenbergHealth();

  res.json({
    status: gotenbergOk ? 'healthy' : 'degraded',
    services: {
      api: true,
      gotenberg: gotenbergOk,
    },
  });
});

export default router;

```

7. Ana Uygulama

```

typescript
// src/index.ts

import express from 'express';
import cors from 'cors';
import convertRouter from './routes/convert.js';

const app = express();
const PORT = process.env.PORT || 3001;

// CORS - Extension'dan gelen istekleri kabul et
app.use(cors({
  origin: [
    'chrome-extension:///*',
    'http://localhost:/*',
  ],
  methods: ['GET', 'POST'],
  allowedHeaders: ['Content-Type'],
}));

app.use(express.json());

// Routes
app.use('/api/convert', convertRouter);

// Root endpoint
app.get('/', (_req, res) => {
  res.json({
    name: 'UYAP UDF Converter API',
    version: '1.0.0',
    endpoints: {
      convert: 'POST /api/convert',
      batch: 'POST /api/convert/batch',
      health: 'GET /api/convert/health',
    },
  });
});

app.listen(PORT, () => {
  console.log(`🚀 API sunucusu çalışıyor: http://localhost:${PORT}`);
  console.log(`📄 Gotenberg URL: ${process.env.GOTENBERG_URL || 'http://localhost:3000'}`);
});

```

8. Dockerfile (API için)

```

dockerfile
FROM node:20-alpine

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

```

```
COPY dist ./dist  
  
EXPOSE 3001  
  
CMD ["node", "dist/index.js"]
```

Çalıştırma

```
bash  
# 1. Bağımlılıkları yükle  
npm install  
  
# 2. Sadece Gotenberg'i başlat (gelistirme için)  
docker run -d -p 3000:3000 gotenberg/gotenberg:8  
  
# 3. API'yi çalıştır  
npm run dev  
  
# VEYA tümünü Docker ile  
docker-compose up -d
```

Chrome Extension'dan Kullanım

```
typescript  
// extension/src/services/converter.ts  
const API_URL = 'http://localhost:3001';  
  
export async function convertUdfToPdf(file: File): Promise<Blob> {  
    const formData = new FormData();  
    formData.append('file', file);  
  
    const response = await fetch(` ${API_URL}/api/convert` , {  
        method: 'POST',  
        body: formData,  
    });  
  
    if (!response.ok) {  
        const error = await response.json();  
        throw new Error(error.error || 'Dönüşürme başarısız');  
    }  
  
    return response.blob();  
}  
  
export async function downloadPdf(blob: Blob, filename: string): void {  
    const url = URL.createObjectURL(blob);  
    const a = document.createElement('a');  
    a.href = url;  
    a.download = filename;  
    a.click();  
    URL.revokeObjectURL(url);  
}
```

Akış 2: UDF → PDF Dönüşümü (opsiyonel)

Kullanıcı UDF dosyalarını seçer → Gotenberg'e gönder → PDF olarak indir

Avantajları:

- Backend olmadan da extension çalışır (sadece indirme)
- Dönüşüm isteğe bağlı
- Gotenberg kurulmamışsa extension yine işlevsel

Gotenberg veya genel mimari hakkında ne merak ediyorsun? Sorularını al, netlestirelim.

PROJECT_REPORT.md

Mevcut Mimari Analizi

Zaten var olanlar:

- file-processor-service.ts → ZIP/UDF/TIFF/JPG işleme (UDF desteği var)
- process-controller.ts → "İşle" sekmesi kontrolcüsü
- 4 sekmeli UI: İndir | AI Analiz | İşle | Ayarlar

İndirme akışı:

UYAP → download_document_brd.uyap → PDF olarak indirilir

Sorularım

1. UYAP'tan inen dosyalar direkt PDF mi? Raporda {evrakAdi}.pdf yazıyor ama UDF işleme servisi de var. Bazı evraklar UDF olarak mı geliyor?
2. Mevcut UDF işleme ne yapıyor? file-processor-service.ts'teki UDF desteği sadece ZIP içinden çıkarma mı, yoksa içeriği de işliyor mu?
3. "İşle" sekmesinde şu an neler var? ZIP açma ve OCR dışında başka özellik var mı?

Önerim: UDF→PDF Dönüşümü Entegrasyonu

"İşle" Sekmesi

```
|--- ZIP Aç (mevcut)  
|--- OCR (mevcut)  
└--- UDF → PDF Dönüşür (yeni) ← Gotenberg backend ile
```

Mevcut file-processor-service.ts'e Gotenberg entegrasyonu eklenebilir. Kullanıcı UDF dosyalarını sürükle-bırak veya dosya seçici ile yükler, backend'e gönderilir, PDF döner.

Faz 1: Robust İndirme Sistemi

UYAP Tree-View

```
📁 Dava 2024/123  
|   ├── 📁 Dilekçeler  
|   |   └── dava-dilekce.pdf  
|   └── cevap.udf  
|   └── 📁 Bilirkişi Raporları  
|       └── rapor.pdf  
|       └── ek-gorsel.tiff  
└── 📁 Duruşma Tutanakları  
    └── tutanak-01.udf
```

Kullanıcı Dizini

```
📁 C:/Davalar/2024-123/  
|   ├── 📁 Dilekçeler/  
|   |   └── dava-dilekce.pdf  
|   └── cevap.udf  
|   └── 📁 Bilirkişi Raporları/  
|       └── rapor.pdf  
|       └── ek-gorsel.tiff  
└── 📁 Duruşma Tutanakları/  
    └── tutanak-01.udf
```

Özellikler:

- File System Access API ile dizin seçimi
- Kesintiye dayanıklılık (indirme durumu kaydı)
- Retry mekanizması (3 deneme, exponential backoff)

- Progress tracking (dosya bazlı + toplam)
- Resume desteği (yarım kalan indirmeden devam)

Faz 2: Post-Process Dönüşüm

İndirilen Klasör: C:/Davalalar/2024-123																			
Tespit Edilen Dosyalar:																			
<table border="1"><tr><td><input checked="" type="checkbox"/> 12 UDF dosyası</td><td>[Dönüştür ▾]</td></tr><tr><td></td><td> └ Gotenberg (yerel)</td></tr><tr><td></td><td> └ e-udf.com (online)</td></tr><tr><td></td><td> └ LibreOffice (manuel)</td></tr><tr><td colspan="2"><input checked="" type="checkbox"/> 5 TIFF dosyası</td></tr><tr><td></td><td> [Dönüştür ▾]</td></tr><tr><td></td><td> └ Tesseract OCR → PDF</td></tr><tr><td></td><td> └ Sharp → PNG/JPEG</td></tr><tr><td></td><td> └ ImageMagick (gelişmiş)</td></tr></table>		<input checked="" type="checkbox"/> 12 UDF dosyası	[Dönüştür ▾]		└ Gotenberg (yerel)		└ e-udf.com (online)		└ LibreOffice (manuel)	<input checked="" type="checkbox"/> 5 TIFF dosyası			[Dönüştür ▾]		└ Tesseract OCR → PDF		└ Sharp → PNG/JPEG		└ ImageMagick (gelişmiş)
<input checked="" type="checkbox"/> 12 UDF dosyası	[Dönüştür ▾]																		
	└ Gotenberg (yerel)																		
	└ e-udf.com (online)																		
	└ LibreOffice (manuel)																		
<input checked="" type="checkbox"/> 5 TIFF dosyası																			
	[Dönüştür ▾]																		
	└ Tesseract OCR → PDF																		
	└ Sharp → PNG/JPEG																		
	└ ImageMagick (gelişmiş)																		
<input type="button" value="[Seçilenleri Dönüştür]"/> <input type="button" value="[Tümünü Dönüştür]"/>																			

TIFF Dönüşüm Seçenekleri

Yöntem	Çıktı	Avantaj	Dezavantaj
Tesseract.js	PDF (aranabilir)	OCR + PDF tek seferde, client-side	Yavaş, büyük dosyalarda sorunlu
Sharp	PNG/JPEG	Hızlı, kaliteli	Sadece görüntü, PDF değil
Gotenberg	PDF	Backend zaten var, TIFF destekler	Backend gereklili
ImageMagick	PDF/çoklu format	En güçlü	Kurulum gereklili

Önerim: Gotenberg hem UDF hem TIFF'i desteklediği için tek backend ile ikisini de çözebiliriz. Client-side alternatif olarak Tesseract.js kalabilir.

Onay Soruları

- İndirme durumu nerede saklanacak?
 - chrome.storage.local (basit, 5MB limit)
 - IndexedDB (büyük veri, karmaşık)
 - JSON dosyası (indirme klasöründe)
- Orijinal dosyalar korunsun mu? Dönüşüm sonrası:
 - rapor.tiff → rapor.tiff + rapor.pdf (yan yana)
 - rapor.tiff → rapor.pdf (üzerine yaz - önerilmez)
- Tree-view yapısını UYAP'tan nasıl alıyorsun? DOM parsing mi, yoksa bir API response mi?

PROJECT_REPORT.md

Mükemmel! Projeyi tam anladım. İşte kritik bulgular:

Mevcut Durum Analizi

Zaten Var Olan Özellikler

Bileşen	Durum	Eksik
Tree-view parse	<input checked="" type="checkbox"/> getKlasorYolu() var	Kullanılmıyor!
File System API	<input checked="" type="checkbox"/> file-processor-service.ts'de	İndirmede yok
UDF metin çıkarma	<input checked="" type="checkbox"/> extractUdfText()	PDF'e dönüştürmüyor
OCR	<input checked="" type="checkbox"/> Tesseract.js	<input checked="" type="checkbox"/> Çalışıyor

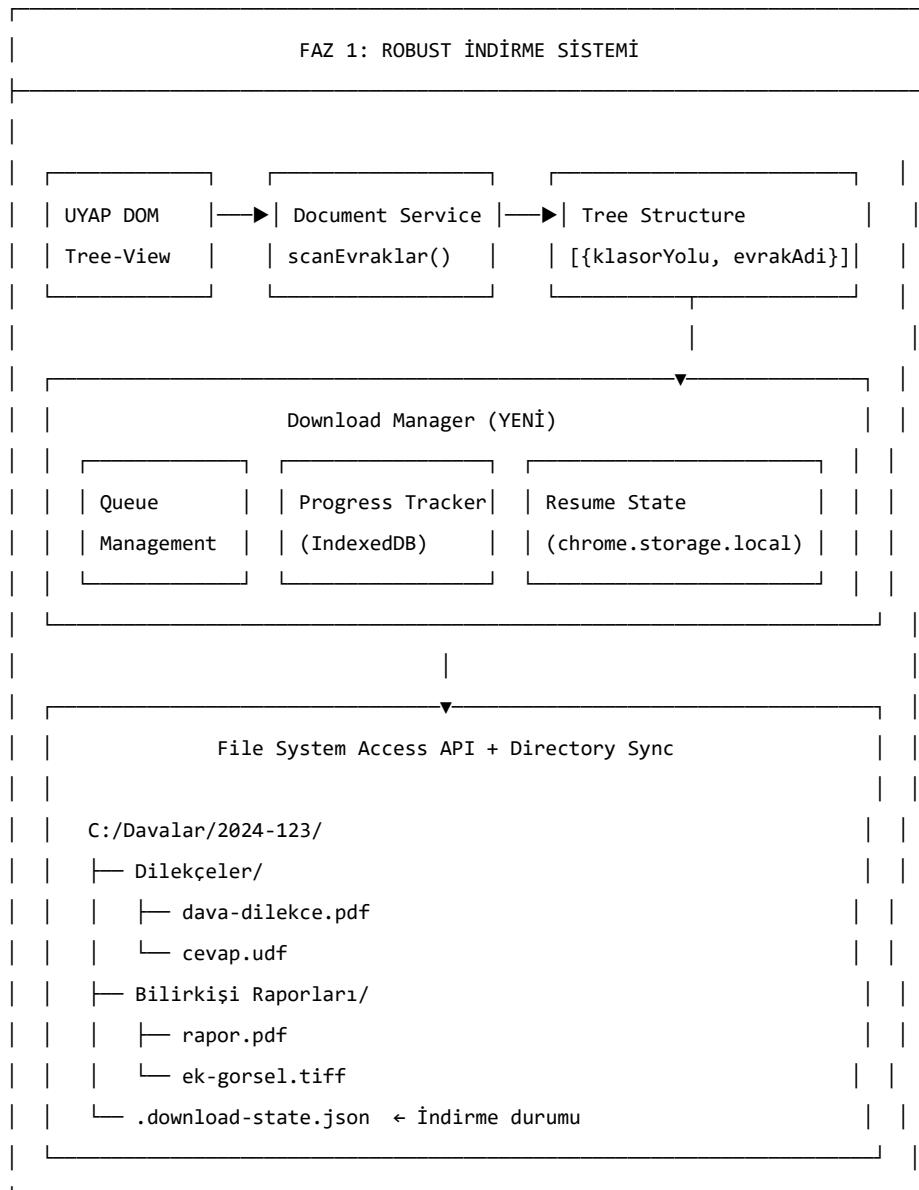
💡 Ana Problem

typescript

```
// download-service.ts - Satır 68
const fileName = `${baseName}.pdf`;

// Klasör yolu kullanılmıyor! Her şey düz yapıda indiriliyor
await this.downloadFile(base64, `${this.outputFolder}/${this.kisiAdi}/${fileName}`);
```
Evrak'ta `klasorYolu` var ama indirmede kullanılmıyor.
```

---  
## 🏠 Yeni Mimari Planı



## FAZ 2: POST-PROCESS DÖNÜŞTÜRME

İndirilen klasörü tara → Dosya tiplerini grupta

|                                                |                                                    |
|------------------------------------------------|----------------------------------------------------|
| 12 UDF Dosyası                                 | Dönüşürme Seçenekleri:                             |
| <input checked="" type="checkbox"/> Tümünü seç | <input type="radio"/> Gotenberg (yerel Dokümanlar) |
| └ cevap.udf                                    | <input type="radio"/> e-udf.com (online API)       |
| └ karar.udf                                    | <input type="radio"/> LibreOffice CLI (manuel)     |
| └ ...                                          |                                                    |

|                                                |  |                                                    |
|------------------------------------------------|--|----------------------------------------------------|
| 5 TIFF Dosyası                                 |  | Dönüştürme Seçenekleri:                            |
| <input checked="" type="checkbox"/> Tümünü seç |  | <input type="radio"/> Tesseract OCR → PDF (client) |
| └── ek1.tiff                                   |  | <input type="radio"/> Gotenberg → PDF (server)     |
| └── ek2.tiff                                   |  | <input type="radio"/> Sharp → PNG/JPEG (client)    |

[Seçilenleri Dönüştür] Çıktı: Aynı klasörde .pdf olarak

## 📁 Yeni/Değiştirilecek Dosyalar

```
src/
 └── services/
 | ├── download-service.ts ← REFACTOR (File System API)
 | ├── download-manager.ts ← YENİ (Queue + Resume)
 | ├── file-processor-service.ts ← EXTEND (Gotenberg entegrasyonu)
 | └── conversion-service.ts ← YENİ (UDF/TIFF dönüştürme)
 └── types/
 └── download.ts ← YENİ (Download state types)
 └── controllers/
 └── download-controller.ts ← REFACTOR
```

## Onay Soruları

1. İndirme state'si nerede tutulsun?

- IndexedDB (büyük veri, karmaşık sorgular)  ÖNERİM
  - chrome.storage.local (5MB limit, basit)
  - .download-state.json dosyası (indirme klasöründe)

2. Orijinal dosya korunsun mu? (dönüşürme sonrası)

- rapor.udf → rapor.udf + rapor.pdf (yan yana)  ÖNERİM

- Üzerine yazma (önerilmez)
3. Backend zorunlu mı? Gotenberg olmadan çalışabilsin mi?
- Evet, sadece client-side seçenekler yeterli
  - Backend varsa ekstra seçenek olsun  ÖNERİM

Onaylarsan download-manager.ts ile başlayalım.

#### ASP.NET Core Web API - Dönüşürme Backend

.NET 8 ile çok daha temiz olur. İşte tam yapı:

UyapConverter/

```
|—— UyapConverter.Api/
| |—— Controllers/
| | |—— ConvertController.cs
| |—— Services/
| | |—— IGotenbergService.cs
| | |—— GotenbergService.cs
| | |—— ConversionResult.cs
| |—— Program.cs
| |—— appsettings.json
| |—— Dockerfile
|—— docker-compose.yml
└—— UyapConverter.sln
```

---

#### 1. Solution ve Proje Oluşturma

bash

```
dotnet new sln -n UyapConverter
dotnet new webapi -n UyapConverter.Api -controllers
dotnet sln add UyapConverter.Api
cd UyapConverter.Api
dotnet add package Microsoft.AspNetCore.OpenApi
```

---

#### 2. appsettings.json

```
json
{
 "Logging": {
 "LogLevel": {
 "Default": "Information",
 "Microsoft.AspNetCore": "Warning"
 }
 },
 "AllowedHosts": "*",
 "Gotenberg": {
 "BaseUrl": "http://localhost:3000",
 "TimeoutSeconds": 120
 },
 "Cors": {
 "AllowedOrigins": [
 "chrome-extension://*"
]
 }
}
```

---

### 3. Services/ConversionResult.cs

```
csharp

namespace UyapConverter.Api.Services;

public sealed record ConversionResult
{
 public bool Success { get; init; }
 public byte[]? PdfData { get; init; }
 public string? FileName { get; init; }
 public string? Error { get; init; }
 public long ProcessingTimeMs { get; init; }

 public static ConversionResult Ok(byte[] data, string fileName, long ms) =>
 new() { Success = true, PdfData = data, FileName = fileName, ProcessingTimeMs = ms };

 public static ConversionResult Fail(string error) =>
 new() { Success = false, Error = error };
}

public sealed record BatchConversionResult
{
 public int TotalFiles { get; init; }
 public int SuccessCount { get; init; }
 public int FailedCount { get; init; }
 public List<ConversionItemResult> Results { get; init; } = [];
}

public sealed record ConversionItemResult
{
 public string OriginalName { get; init; } = string.Empty;
 public string? OutputName { get; init; }
 public bool Success { get; init; }
 public string? Error { get; init; }
 public byte[]? Data { get; init; }
}
```

---

### 4. Services/IGotenbergService.cs

```
csharp

namespace UyapConverter.Api.Services;

public interface IGotenbergService
{
 Task<bool> HealthCheckAsync(CancellationToken ct = default);
 Task<ConversionResult> ConvertUdfToPdfAsync(Stream fileStream, string fileName, CancellationToken ct = default);
 Task<ConversionResult> ConvertTiffToPdfAsync(Stream fileStream, string fileName, CancellationToken ct = default);
 Task<BatchConversionResult> ConvertBatchAsync(IFormFileCollection files, CancellationToken ct = default);
}
```

---

### 5. Services/GotenbergService.cs

```
csharp

using System.Diagnostics;
using System.Net.Http.Headers;

namespace UyapConverter.Api.Services;

public sealed class GotenbergService : IGotenbergService
{
 private readonly HttpClient _httpClient;
 private readonly ILogger<GotenbergService> _logger;

 private const string LibreOfficeEndpoint = "/forms/libreoffice/convert";
 private const string HealthEndpoint = "/health";

 public GotenbergService(HttpClient httpClient, ILogger<GotenbergService> logger)
 {
 _httpClient = httpClient;
 _logger = logger;
 }

 public async Task<bool> HealthCheckAsync(CancellationToken ct = default)
 {
 try
 {
 var response = await _httpClient.GetAsync(HealthEndpoint, ct);
 return response.IsSuccessStatusCode;
 }
 catch (Exception ex)
 {
 _logger.LogWarning(ex, "Gotenberg health check failed");
 return false;
 }
 }

 public async Task<ConversionResult> ConvertUdfToPdfAsync(
 Stream fileStream,
 string fileName,
 CancellationToken ct = default)
 {
 return await ConvertWithLibreOfficeAsync(fileStream, fileName, ct);
 }

 public async Task<ConversionResult> ConvertTiffToPdfAsync(
 Stream fileStream,
 string fileName,
 CancellationToken ct = default)
 {
 return await ConvertWithLibreOfficeAsync(fileStream, fileName, ct);
 }
}
```

```

public async Task<BatchConversionResult> ConvertBatchAsync(
 IFormFileCollection files,
 CancellationToken ct = default)
{
 var results = new List<ConversionItemResult>();
 var successCount = 0;
 var failedCount = 0;

 foreach (var file in files)
 {
 await using var stream = file.OpenReadStream();
 var result = await ConvertWithLibreOfficeAsync(stream, file.FileName, ct);

 if (result.Success)
 {
 successCount++;
 results.Add(new ConversionItemResult
 {
 OriginalName = file.FileName,
 OutputName = result.FileName,
 Success = true,
 Data = result.PdfData
 });
 }
 else
 {
 failedCount++;
 results.Add(new ConversionItemResult
 {
 OriginalName = file.FileName,
 Success = false,
 Error = result.Error
 });
 }
 }

 return new BatchConversionResult
 {
 TotalFiles = files.Count,
 SuccessCount = successCount,
 FailedCount = failedCount,
 Results = results
 };
}

private async Task<ConversionResult> ConvertWithLibreOfficeAsync(
 Stream fileStream,
 string fileName,
 CancellationToken ct)
{

```

```

var sw = Stopwatch.StartNew();

try
{
 using var content = new MultipartFormDataContent();

 var fileContent = new StreamContent(fileStream);
 fileContent.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");
 content.Add(fileContent, "files", fileName);

 // Gotenberg LibreOffice endpoint
 var response = await _httpClient.PostAsync(LibreOfficeEndpoint, content, ct);

 if (!response.IsSuccessStatusCode)
 {
 var errorBody = await response.Content.ReadAsStringAsync(ct);
 _logger.LogError("Gotenberg conversion failed: {Status} - {Body}",
 response.StatusCode, errorBody);
 return ConversionResult.Fail($"Conversion failed: {response.StatusCode}");
 }

 var pdfData = await response.Content.ReadAsByteArrayAsync(ct);
 var outputName = Path.ChangeExtension(fileName, ".pdf");

 sw.Stop();
 _logger.LogInformation("Converted {FileName} in {Ms}ms", fileName, sw.ElapsedMilliseconds);

 return ConversionResult.Ok(pdfData, outputName, sw.ElapsedMilliseconds);
}

catch (TaskCanceledException)
{
 return ConversionResult.Fail("Conversion timeout");
}

catch (Exception ex)
{
 _logger.LogError(ex, "Conversion error for {FileName}", fileName);
 return ConversionResult.Fail(ex.Message);
}
}
}

```

## 6. Controllers/ConvertController.cs

```

csharp
using Microsoft.AspNetCore.Mvc;
using UyapConverter.Api.Services;

namespace UyapConverter.Api.Controllers;

[ApiController]
[Route("api/[controller]")]

```

```

public class ConvertController : ControllerBase
{
 private readonly IGutenbergService _gotenbergService;
 private readonly ILogger<ConvertController> _logger;

 private static readonly HashSet<string> AllowedExtensions = new(StringComparer.OrdinalIgnoreCase)
 {
 ".udf", ".tiff", ".tif"
 };

 private const long MaxFileSize = 50 * 1024 * 1024; // 50MB

 public ConvertController(IGutenbergService gotenbergService, ILogger<ConvertController> logger)
 {
 _gotenbergService = gotenbergService;
 _logger = logger;
 }

 ///<summary>
 ///<Health check endpoint
 ///</summary>
 [HttpGet("health")]
 public async Task<IActionResult> HealthCheck(CancellationToken ct)
 {
 var gotenbergHealthy = await _gotenbergService.HealthCheckAsync(ct);

 return Ok(new
 {
 Status = gotenbergHealthy ? "healthy" : "degraded",
 Gotenberg = gotenbergHealthy,
 Timestamp = DateTime.UtcNow
 });
 }

 ///<summary>
 ///<Tek dosya dönüştürme (UDF veya TIFF → PDF)
 ///</summary>
 [HttpPost]
 [RequestSizeLimit(MaxFileSize)]
 public async Task<IActionResult> ConvertSingle(IFormFile file, CancellationToken ct)
 {
 if (file is null || file.Length == 0)
 {
 return BadRequest(new { Error = "Dosya gerekli" });
 }

 var extension = Path.GetExtension(file.FileName);
 if (!AllowedExtensions.Contains(extension))
 {
 return BadRequest(new { Error = $"Desteklenmeyen dosya tipi: {extension}" });
 }
 }
}

```

```

}

_logger.LogInformation("Converting {FileName} ({Size} bytes)", file.FileName, file.Length);

await using var stream = file.OpenReadStream();

var result = extension.ToLowerInvariant() switch
{
 ".udf" => await _gotenbergService.ConvertUdfToPdfAsync(stream, file.FileName, ct),
 ".tiff" or ".tif" => await _gotenbergService.ConvertTiffToPdfAsync(stream, file.FileName, ct),
 _ => ConversionResult.Fail("Unsupported file type")
};

if (!result.Success)
{
 return UnprocessableEntity(new { result.Error });
}

return File(result.PdfData!, "application/pdf", result.FileName);
}

/// <summary>
/// Toplu dönüştürme
/// </summary>
[HttpPost("batch")]
[RequestSizeLimit(MaxFileSize * 10)] // 500MB toplam
public async Task<IActionResult> ConvertBatch(IFormFileCollection files, CancellationToken ct)
{
 if (files is null || files.Count == 0)
 {
 return BadRequest(new { Error = "En az bir dosya gerekli" });
 }

 // Validasyon
 var invalidFiles = files
 .Where(f => !AllowedExtensions.Contains(Path.GetExtension(f.FileName)))
 .Select(f => f.FileName)
 .ToList();

 if (invalidFiles.Count > 0)
 {
 return BadRequest(new { Error = "Desteklenmeyen dosyalar", Files = invalidFiles });
 }

 _logger.LogInformation("Batch converting {Count} files", files.Count);

 var result = await _gotenbergService.ConvertBatchAsync(files, ct);

 // Base64 olarak döndür (çoklu dosya için)
 var response = new

```

```

 {
 result.TotalFiles,
 result.SuccessCount,
 result.FailedCount,
 Files = result.Results.Select(r => new
 {
 r.OriginalName,
 r.OutputName,
 r.Success,
 r.Error,
 Data = r.Success ? Convert.ToBase64String(r.Data!) : null
 })
 };

 return Ok(response);
}

/// <summary>
/// Desteklenen formatlar
/// </summary>
[HttpGet("formats")]
public IActionResult GetSupportedFormats()
{
 return Ok(new
 {
 Input = new[] { ".udf", ".tiff", ".tif" },
 Output = new[] { ".pdf" },
 MaxFileSizeMB = MaxFileSize / (1024 * 1024)
 });
}
}

```

---

## 7. Program.cs

```

csharp
using UyapConverter.Api.Services;

var builder = WebApplication.CreateBuilder(args);

// Controllers
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
 c.SwaggerDoc("v1", new() { Title = "UYAP Converter API", Version = "v1" });
});

// Gotenberg HttpClient
builder.Services.AddHttpClient<IGotenbergService, GotenbergService>(client =>
{
 var baseUrl = builder.Configuration["Gotenberg:BaseUrl"] ?? "http://localhost:3000";

```

```

var timeout = builder.Configuration.GetValue<int>("Gotenberg:TimeoutSeconds", 120);

client.BaseAddress = new Uri(baseUrl);
client.Timeout = TimeSpan.FromSeconds(timeout);
});

// CORS - Chrome Extension için
builder.Services.AddCors(options =>
{
 options.AddPolicy("ChromeExtension", policy =>
 {
 policy
 .SetIsOriginAllowed(origin =>
 origin.StartsWith("chrome-extension://") ||
 origin.StartsWith("http://localhost"))
 .AllowAnyMethod()
 .AllowAnyHeader();
 });
});

var app = builder.Build();

// Swagger (Development)
if (app.Environment.IsDevelopment())
{
 app.UseSwagger();
 app.UseSwaggerUI();
}

app.UseCors("ChromeExtension");
app.MapControllers();

app.Run();

```

## 8. Dockerfile

```

dockerfile
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
WORKDIR /app
EXPOSE 8080

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["UyapConverter.Api.csproj", "."]
RUN dotnet restore
COPY .. .
RUN dotnet build -c Release -o /app/build

FROM build AS publish
RUN dotnet publish -c Release -o /app/publish /p:UseAppHost=false

```

```
FROM base AS final
WORKDIR /app
COPY --from=build /app/publish .
ENTRYPOINT ["dotnet", "UyapConverter.Api.dll"]
```

---

## 9. docker-compose.yml

```
yaml
version: '3.8'

services:
```

```
api:
 build:
 context: ./UyapConverter.Api
 dockerfile: Dockerfile
 ports:
 - "5000:8080"
 environment:
 - ASPNETCORE_ENVIRONMENT=Production
 - Gotenberg__BaseUrl=http://gotenberg:3000
depends_on:
```

```
 gotenberg:
 condition: service_healthy
```

```
networks:
 - uyap-network
```

```
gotenberg:
 image: gotenberg/gotenberg:8
 ports:
```

```
 - "3000:3000"
 healthcheck:
 test: ["CMD", "curl", "-f", "http://localhost:3000/health"]
 interval: 10s
 timeout: 5s
 retries: 3
 networks:
 - uyap-network
```

```
networks:
```

```
 uyap-network:
 driver: bridge
```

---

## 🚀 Çalıştırma

```
bash
Development (Docker olmadan)
cd UyapConverter.Api
dotnet run
```

```
Production (Docker ile)
docker-compose up -d
```

```
Test
curl -X POST http://localhost:5000/api/convert \
-F "file=@evrak.udf"
```

---

#### Chrome Extension Entegrasyonu

```
typescript
// conversion-service.ts
const API_BASE = 'http://localhost:5000/api/convert';

export async function convertUdfToPdf(file: File): Promise<Blob> {
 const formData = new FormData();
 formData.append('file', file);

 const response = await fetch(API_BASE, {
 method: 'POST',
 body: formData
 });

 if (!response.ok) {
 const error = await response.json();
 throw new Error(error.Error || 'Dönüştürme hatası');
 }

 return response.blob();
}
```

---

Node.js yerine .NET tercih mantıklı mı?

Modal-Dialog Modal-LG Canlı DOM Yapısı

Sayfa: [vatandas.uyap.gov.tr](http://vatandas.uyap.gov.tr) – Dosya Sorgulama → "Dosya Görüntüle" butonu ile açılan modal

Modal Yapısı:

```
.modal-dialog.modal-lg
└── .modal-header
 └── h4: "2021/612 - Isparta 2. Aile Mahkemesi - Hukuk Dava Dosyası"
└── .modal-body
 └── Tab yapısı (Dosya Bilgileri | Taraf Bilgileri | Evrak | Safahat | ...)
 └── #dosya_evrak_bilgileri_tab (Evrak sekmesi)
 └── #dosya_evrak_bilgileri_result
 └── .row (sayfalama: "Toplam 1 sayfadan 1. sayfa" + Önceki/Sonraki)
 └── <hr>
 └── ul#browser.filetree.treeview-gray.treeview
 └── li (kök klasör)
 └── span.folder → "Isparta 2. Aile Mahkemesi 2021/612"
 └── ul
 └── li > span.folder → "Dosyaya Eklenen Son 20 Evrak"
 └── li > span.file[evrak_id] → dosyalar
 └── li > span.folder → "Tüm Evraklar"
 └── li > span.folder → "2021/612 (Hukuk Dava Dosyası)"
 └── li > span.folder → "Tensip Zaptı (1)" vb. (46 klasör)
 └── li > span.file[evrak_id] → dosyalar
 └── iframe#onizleiframe (Ön izleme için)
```

```

└── .modal-footer#modal-footer
 └── button.btn.default → "Kapat"

Dosya (span.file) Attribute'ları: * class="file" * evrak_id="{evrakId}" – indirme için kritik * ce="V" – tüm dosyalarda
sabit * data-original-title – HTML tooltip (Birim Evrak No, Onay Tarihi, Gönderen Yer, Dosya No bilgisi) * data-placement,
data-html, title jQuery Event'leri (span.file üzerinde): focusin, focusout, mouseover, mouseout, dblclick, contextmenu,
click Context Menu (Sağ Tık):

table (position: absolute, z-index yüksek)
└── td
 └── div.context-menu.context-menu-theme-vista
 ├── div.context-menu-item → "Görüntüle"
 ├── div.context-menu-separator
 └── div.context-menu-item → "Kaydet"

Dosya Sayıları: 240 dosya, 46 klasör UDF Viewer (İkinci Tab) URL Pattern:
viewer.jsp?mimeType=Udf&evrakId={ID}&dosyaId={ID}&yargiTuru=1 Yapısı:

.xml-viewer
└── .portlet.box.green
 ├── .portlet-title
 | ├── .caption → "Evrak Görüntüleme"
 | └── .tools → div[onclick="Viewer.download()"] → ".Udf Uzantılı Evrakı İndir"
 └── .portlet-body
 └── #evrakViewContent (UDF HTML olarak render edilmiş: 63 paragraf, tablolar)

```

Önemli bulgu: UDF dosyaları UYAP tarafından zaten HTML olarak render ediliyor. Yani viewer.jsp endpoint'i UDF'i parse edip HTML'e çeviriyor. Bu, Gotenberg'e alternatif olarak doğrudan bu HTML çıktısından PDF üretme imkanı verebilir.

Analyzed DOM yapısını ve PDF oluşturma stratejisini yeniden değerlendirdi

Bu bulgular altın değerinde! Özellikle son bulgu oyun değiştirici:

### Kritik Keşif: UYAP Zaten UDF → HTML Yapıyor!

viewer.jsp?mimeType=Udf&evrakId={ID}&dosyaId={ID}&yargiTuru=1

└── #evrakViewContent → Hazır HTML çıktı!

Bu demek ki:

-  Gotenberg'e gerek yok
-  LibreOffice'e gerek yok
-  UYAP'ın kendi render'ını kullan → window.print() veya Puppeteer ile PDF

### Yeni Strateji Karşılaştırması

| Yöntem          | Avantaj                                   | Dezavantaj             |
|-----------------|-------------------------------------------|------------------------|
| UYAP HTML → PDF | Sıfır bağımlılık, her zaman güncel format | UYAP oturumu gereklili |
| Gotenberg       | Offline çalışır                           | Docker, ekstra backend |
| Raw UDF İndir   | En hızlı                                  | Dönüştürme ayrı adım   |

Öneri: Hibrit yaklaşım - önce raw indir, sonra UYAP viewer'dan PDF opsyonunu sun.

### Güncellenmiş DOM Parser

Paylaştığın yapıya göre document-service.ts güncellenmeli:

typescript

// src/services/document-service.ts - Güncellenmiş Parser

```

export interface EvrakMetadata {
 evrakId: string;
 birimEvrakNo?: string;
}

```

```

onayTarihi?: string;
gonderilenYer?: string;
dosyaNo?: string;
}

export interface TreeNode {
 type: 'folder' | 'file';
 name: string;
 evrakId?: string;
 metadata?: EvrakMetadata;
 children?: TreeNode[];
 path: string; // Tam klasör yolu
}

/**
 * UYAP Tree-View DOM'unu parse eder
 * Hedef: ul#browser.filetree.treeview-gray.treeview
 */
export function parseUyapTreeView(): TreeNode | null {
 const treeRoot = document.querySelector('ul#browser.filetree.treeview');

 if (!treeRoot) {
 console.warn('[UYAP] Tree-view bulunamadı');
 return null;
 }

 return parseTreeNode(treeRoot.querySelector(':scope > li'), '');
}

function parseTreeNode(li: Element | null, parentPath: string): TreeNode | null {
 if (!li) return null;

 const span = li.querySelector(':scope > span.folder, :scope > span.file');
 if (!span) return null;

 const isFolder = span.classList.contains('folder');
 const name = span.textContent?.trim() || '';
 const currentPath = parentPath ? `${parentPath}/${name}` : name;

 if (isFolder) {
 // Klasör node'u
 const childUl = li.querySelector(':scope > ul');
 const children: TreeNode[] = [];

 if (childUl) {
 childUl.querySelectorAll(':scope > li').forEach(childLi => {
 const childNode = parseTreeNode(childLi, currentPath);
 if (childNode) children.push(childNode);
 });
 }
 }
}

```

```

return {
 type: 'folder',
 name,
 path: currentPath,
 children
};

} else {
 // Dosya node'u
 const evrakId = span.getAttribute('evrak_id') || '';
 const metadata = parseTooltipMetadata(span);

 return {
 type: 'file',
 name,
 evrakId,
 metadata,
 path: currentPath
 };
}

}

/***
 * data-original-title tooltip'inden metadata çıkarır
 * Format: "Birim Evrak No: X
Onay Tarihi: Y
..."
 */
function parseTooltipMetadata(span: Element): EvrakMetadata {
 const tooltip = span.getAttribute('data-original-title') || '';
 const evrakId = span.getAttribute('evrak_id') || '';

 const metadata: EvrakMetadata = { evrakId };

 // HTML tooltip'i parse et
 const lines = tooltip.split(/<br\s*/\?>/i);

 for (const line of lines) {
 const [key, ...valueParts] = line.split(':');
 const value = valueParts.join(':').trim();

 const keyLower = key.toLowerCase().trim();

 if (keyLower.includes('birim evrak')) {
 metadata.birimEvrakNo = value;
 } else if (keyLower.includes('onay tarihi')) {
 metadata.onayTarihi = value;
 } else if (keyLower.includes('gonderen') || keyLower.includes('gonderen')) {
 metadata.gonderilenYer = value;
 } else if (keyLower.includes('dosya no')) {
 metadata.dosyaNo = value;
 }
 }
}

```

```

}

return metadata;
}

/***
 * Tree'den tüm dosyaları düz liste olarak çıkarır
 */
export function flattenTree(node: TreeNode): Array<{ file: TreeNode; folderPath: string }> {
const result: Array<{ file: TreeNode; folderPath: string }> = [];

function traverse(n: TreeNode, path: string) {
if (n.type === 'file') {
result.push({ file: n, folderPath: path });
} else if (n.children) {
for (const child of n.children) {
traverse(child, n.path);
}
}
}

traverse(node, '');
return result;
}

/***
 * Dosya ID ve diğer bilgileri modal header'dan çıkarır
 */
export function parseDosyaInfo(): { dosyaNo: string; mahkeme: string; dosyaTuru: string } | null {
const header = document.querySelector('.modal-dialog.modal-lg .modal-header h4');
if (!header) return null;

// Format: "2021/612 - Isparta 2. Aile Mahkemesi - Hukuk Dava Dosyası"
const text = header.textContent?.trim() || '';
const parts = text.split(' - ');

if (parts.length >= 3) {
return {
dosyaNo: parts[0].trim(),
mahkeme: parts[1].trim(),
dosyaTuru: parts[2].trim()
};
}

return null;
}

```

---

#### UYAP Viewer PDF Dönüştürücü

typescript

// src/services/uyap-pdf-service.ts

```

export interface ViewerPdfOptions {
 evrakId: string;
 dosyaId: string;
 yargiTuru?: string;
 paperSize?: 'A4' | 'Letter';
}

/**
 * UYAP viewer.jsp'den HTML alıp PDF'e çevirir
 * Browser'in print-to-PDF özelliğini kullanır
 */
export async function convertViewerToPdf(options: ViewerPdfOptions): Promise<Blob> {
 const { evrakId, dosyaId, yargiTuru = '1' } = options;

 // 1. Viewer URL'i oluştur
 const viewerUrl =
`/main/jsp/avukat/evrak_viewer.jsp?mimeType=Udf&evrakId=${evrakId}&dosyaId=${dosyaId}&yargiTuru=${yargiTuru}`;

 // 2. Hidden iframe'de aç
 const iframe = document.createElement('iframe');
 iframe.style.cssText = 'position:absolute;left:-9999px;width:800px;height:600px;';
 document.body.appendChild(iframe);

 try {
 // 3. Viewer'i yükle
 await new Promise<void>((resolve, reject) => {
 iframe.onload = () => resolve();
 iframe.onerror = () => reject(new Error('Viewer yüklenemedi'));
 iframe.src = viewerUrl;
 });
 }

 // 4. İçeriğin yüklenmesini bekle
 await waitForContent(iframe);

 // 5. HTML içeriğini al
 const content = iframe.contentDocument?.querySelector('#evrakViewContent');
 if (!content) {
 throw new Error('Evrak içeriği bulunamadı');
 }

 // 6. Print-ready HTML oluştur
 const printHtml = createPrintableHtml(content.innerHTML);

 // 7. PDF'e çevir (browser print API)
 return await generatePdfFromHtml(printHtml);

} finally {
 document.body.removeChild(iframe);
}

```

```

}

function createPrintableHtml(content: string): string {
 return `
 <!DOCTYPE html>
 <html>
 <head>
 <meta charset="UTF-8">
 <style>
 @page { size: A4; margin: 2cm; }
 body {
 font-family: 'Times New Roman', serif;
 font-size: 12pt;
 line-height: 1.5;
 }
 table {
 width: 100%;
 border-collapse: collapse;
 margin: 1em 0;
 }
 td, th {
 border: 1px solid #000;
 padding: 8px;
 }
 p { margin: 0.5em 0; }
 </style>
 </head>
 <body>${content}</body>
 </html>
 `;
}

async function waitForContent(iframe: HTMLIFrameElement, timeout = 10000): Promise<void> {
 const start = Date.now();

 while (Date.now() - start < timeout) {
 const content = iframe.contentDocument?.querySelector('#evrakViewContent');
 if (content && content.innerHTML.trim().length > 0) {
 return;
 }
 await new Promise(r => setTimeout(r, 100));
 }

 throw new Error('Viewer içerik yükleme timeout');
}

async function generatePdfFromHtml(html: string): Promise<Blob> {
 // Yöntem 1: Print API (basit, kullanıcı etkileşimi gerektirir)
 // Yöntem 2: html2pdf.js kütüphanesi (client-side)
 // Yöntem 3: Background script üzerinden chrome.printing API
}

```

```

// html2pdf.js kullanımı:
const { default: html2pdf } = await import('html2pdf.js');

const container = document.createElement('div');
container.innerHTML = html;
container.style.cssText = 'position:absolute;left:-9999px;';
document.body.appendChild(container);

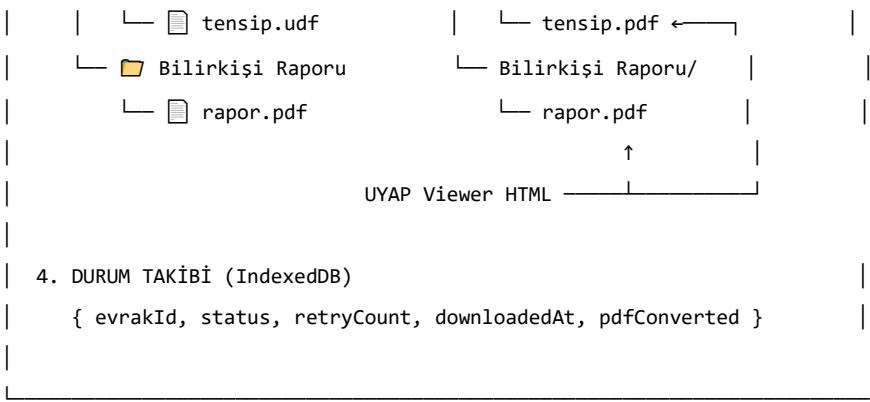
try {
 const pdf = await html2pdf()
 .set({
 margin: [20, 15, 20, 15], // mm
 filename: 'evrak.pdf',
 image: { type: 'jpeg', quality: 0.98 },
 html2canvas: { scale: 2, useCORS: true },
 jsPDF: { unit: 'mm', format: 'a4', orientation: 'portrait' }
 })
 .from(container)
 .outputPdf('blob');

 return pdf;
} finally {
 document.body.removeChild(container);
}
}
```
---
```

📁 Güncellenmiş İndirme Akışı

```

|---+
|   UYAP Evrak İndirme v2
|---+
|
| 1. TARAMA
|   └ parseUyapTreeView() → Tree yapısı + metadata
|
| 2. İNDİRME MODU SEÇİMİ
|   └
|     | o Ham Dosya (UDF/TIFF/PDF olduğu gibi) [Hızlı]
|     | o PDF Dönüştür (UYAP Viewer → PDF) [Yavaş]
|     | o Hibrit (PDF varsa PDF, UDF ise dönüştür) [Önerilen]
|   └
|
| 3. KLASÖR YAPISI KORUMA
|   UYAP Tree:           Lokal:
|   └ Isparta 2. Aile 2021/612 C:/Davalalar/2021-612/
|   └   └ Tensip Zaptı   └ Tensip Zaptı/
|---+
```



❓ Karar Noktaları

1. UDF Dönüşümme Stratejisi:

- A) UYAP Viewer HTML → html2pdf.js (sıfır backend) ÖNERİM
- B) Gotenberg backend (offline çalışır)
- C) Her ikisi de opsiyonel

2. "Dosyaya Eklenen Son 20 Evrak" klasörü:

- Atla (duplikasyon önleme) ÖNERİM
- Dahil et (kullanıcı seçsin)

3. Tooltip metadata kullanımı:

- Dosya adına ekle: tensip_2021-03-15_BirimNo123.pdf
- Sadece JSON manifest'e yaz
- Kullanma

vatandas.uyap bağlamı – Sayfa 1: Giriş Sayfası: vatandas.uyap Bağlam –

Adım 1: Giriş Sayfası URL: vatandas.uyap.gov.tr/main/vatandas/giris.jsp Sayfa: Sisteme Giriş Seçenekleri Giriş Yöntemleri (4 tab): 1. E-Devlet Giriş (#tab_0) – "E-Devlet Aracılığıyla Giriş" butonu → e-Devlet Kapısı'na yönlendirme 2. E-İmza Girişleri (#tab_1) – e-Devlet üzerinden e-imza ile giriş 3. M-İmza Girişleri (#tab_2) – Vodafone/Turkcell/Türk Telekom mobil imza 4. UYAP e-İmza ile Giriş (#tab_3) – Doğrudan UYAP e-imza: Cihaz Seç → Sertifika Seç → Pin Kodu → Giriş Yap Eklenti için önemli notlar: * Giriş sayfasında eklientimizin yapacağı bir şey yok – kullanıcı kendi kimlik doğrulamasını yapmalı * Content script matches pattern'i https://vatandas.uyap.gov.tr/* olduğundan bu sayfa da eşleşir, ama eklienti UI'ı sadece giriş sonrası sayfalarda (index.jsp) gösterilmeli * Sayfa jQuery kullanıyor (daha önce tespit ettik) Navigasyon: Ana Sayfa → Hizmetlerimiz (Vatandaş/Avukat/Kurum/Bilirkişi/Arabulucu/Uzlaştırıcı/Akademî/Portal Yönetim) → SSS → İletişim

Catalogued technical observations, awaited additional documentation

Notları aldım. Sayfa 1 bilgisi kaydedildi:

📄 UYAP Teknik Gözlemler - Sayfa 1/N

Sayfa	URL	Eklenti Durumu
Giriş	/main/vatandas/giris.jsp	✗ Pasif (UI gösterme)

Kritik Kural:

typescript

```

// content.ts'de sayfa kontrolü
const isLoggedInPage = location.pathname.includes('giris.jsp');
const is MainPage = location.pathname.includes('index.jsp');

if (isLoggedInPage) {
    // Eklenti UI'ı gösterme, sadece dinle
    return;
}

```

vatandas.uyap.bağlamlı – Adım 2: e-Devlet Kimlik Doğrulama: vatandas.uyap Bağlam – Adım 2: e-Devlet Giriş Sayfası URL: giriş.turkiye.gov.tr/Giris/gir Sayfa: e-Devlet Kapısı – Kimlik Doğrulama Sistemi OAuth Akışı: * UYAP → e-Devlet'e OAuth2 yönlendirmesi yapıyor * scope: Kimlik-Dogrula;Ad-Soyad (sadece kimlik doğrulama ve ad-soyad bilgisi istiyor) * redirect_uri: vatandas.uyap.gov.tr/login.uyap (başarılı giriş sonrası geri dönüş) * response_type: code (Authorization Code flow) Giriş Yöntemleri (4 tab): 1. e-Devlet Şifresi – TC Kimlik No + e-Devlet Şifresi 2. Elektronik İmza – e-İmza cihazı ile 3. İnternet Bankacılığı – Banka üzerinden doğrulama 4. Diğer Yöntemler – Dropdown ile diğer seçenekler Form Alanları (e-Devlet Şifresi tab'ı): * T.C. Kimlik No (type: number, sanal klavye desteği var) * e-Devlet Şifresi (type: password) * Giriş Yap (submit) / iptal butonları Eklenti için önemli notlar: * Bu sayfa farklı domain'de (giris.turkiye.gov.tr) – content script burada çalışmaz ve çalışmamalı * Eklentimiz bu sürece hiç müdahale etmemeli – kullanıcı kendi kimlik doğrulamasını yapar * Başarılı giriş sonrası vatandas.uyap.gov.tr/login.uyap → ana sayfaya yönlendirilir * Eklenti ancak vatandas.uyap.gov.tr/* domain'inde aktif olmalı

Analyzed UYAP system's OAuth authentication flow and extension scope limitations

Sayfa 2 kaydedildi:

UYAP Teknik Gözlemler - Sayfa 2/N

Sayfa	URL/Domain	Eklenti Durumu
Giriş	vatandas.uyap.gov.tr/giris.jsp	Pasif
e-Devlet OAuth	giris.turkiye.gov.tr/*	Kapsam dışı

Manifest Kontrolü:

```
json
{
  "content_scripts": [
    {
      "matches": ["https://vatandas.uyap.gov.tr/*"]
      //  giris.turkiye.gov.tr dahil DEĞİL – doğru
    }
  ]
}
```

```

\*\*OAuth Akış Notu:\*\*

```
UYAP → e-Devlet (code grant) → /login.uyap → index.jsp
↑ ↓
Eklenti yok Eklenti aktifleşir
```

Recognized compliment and determined gracious response approach

Teşekkür ederim Ufuk, naziksin!

Devam et, Sayfa 3'ü bekliyorum.

Adım 3: Ana Sayfa + Teknoloji Raporu URL: vatandas.uyap.gov.tr/main/vatandas/index.jsp?v={version} SUNUCU TARAFLI TEKNOLOJİ: \* Java EE + JSP (Java Server Pages) – Tüm sayfalar .jsp uzantılı \* Session: JSESSIONID cookie (standart Java Servlet session) \* Ek Cookie: TS01953787 (muhtemelen F5/load balancer veya WAF) \* CSRF: DOM'da CSRF token bulunamadı – eklenti fetch isteklerinde ekstra CSRF header'a gerek yok gibi görünüyor \* Endpoint Pattern: .uyap uzantılı (ör: download\_document\_brd.uyap, login.uyap) \* Chatbot: vatandaschatbot.adilet.gov.tr (Socket.IO) İSTEMCİ TARAFLI TEKNOLOJİ STACKİ: KütüphaneVersiyonKullanımjQuery1.11.2Ana frameworkjQuery UI1.11.2Dialog, draggable vb.Bootstrap3.3.5Layout, modal, tabDataTables1.10.16Tabelo listelemeye (dosya listesi)jQuery Treeview-Dosya ağacı (filetree)jQuery ContextMenu-Sağ tık menüsü (Görüntüle/Kaydet)Select2-Gelişmiş dropdown'larjsTree-Ağaç yapısı (alternatif?)Handlebars-Template engineCKEditor-Zengin metin editörü (evrak oluşturma)Moment.js-Tarih işlemeBlockUI-Loading overlayPace-Progress barBootpag-Sayfalama SAYFA NAVİGASYON MEKANİZMASI:

```
menuButtonClick(menuId)
 → Application.openContentPage(contentId)
 → KodluBilgiler.contentList[contentId] ile JSP path bulunur
 → AJAX ile JSP yüklenir → .page-content içine inject edilir
```

Yani sayfa tam yenilenmez, SPA benzeri bir yapı var – içerik AJAX ile yükleniyor. Bu eklenti için önemli: MutationObserver veya sayfa değişikliği dinleme gerekebilir. SOL MENÜ YAPISI (menuId'ler): MenümenuIdKişisel Bilgiler12568İsim ve Adres Bilgileri12570İletişim Bilgilerim13828UYAP İşlemlerim30209İşlemlerim12572Sorgulama İşlemleri (Dosya Sorgulama)12573 EKLENTİ İÇİN DİKKAT EDİLMESİ GEREKENLER: 1. jQuery 1.11.2 çakışma riski – Eklentimiz Preact kullandığı için jQuery'ye dokunmuyoruz ama UYAP'ın jQuery event'lerini bozmamalıyız. event.stopPropagation() kullanımına dikkat. 2. SPA benzeri navigasyon – Sayfa yenilenmeden menü değişiyor. Content script'in dosya sorgulama sayfasına geçiş tespit etmesi için

URL değişikliği (`menuId` parametresi) veya DOM mutation dinlemesi gereklidir. 3. JSESSIONID bağımlılığı – `fetch()` istekleri otomatik olarak cookie gönderir (`credentials: 'same-origin'`), ayrı session yönetimi gerekmeyez. 4. CSS izolasyonu zorunlu – Bootstrap 3.3.5 + uyap.css + template.css sayfada aktif. Tailwind prefix (uyap-) mutlaka kullanılmalıdır, yoksa stil çakışması kaçınılmaz. 5. ContextMenu plugin'i – UYAP kendi jQuery contextMenu plugin'ini kullanıyor. Eklentimiz bu plugin'e müdahale etmemeli.

Show more

16:03

Catalogued UYAP's technology stack and navigation mechanisms for integration

Mükemmel detay! Sayfa 3 kaydedildi:

## UYAP Teknik Gözlemler - Sayfa 3/N

| Sayfa     | URL                  | Eklenti Durumu                                                                                |
|-----------|----------------------|-----------------------------------------------------------------------------------------------|
| Giriş     | /giris.jsp           |  Pasif       |
| e-Devlet  | giris.turkiye.gov.tr |  Kapsam dışı |
| Ana Sayfa | /index.jsp           |  AKTİF       |

### Kritik Mimari Bulgular

#### 1. SPA Benzeri Navigasyon

javascript

```
// UYAP'ın kendi mekanizması:
menuButtonClick(menuId)
 → Application.openContentPage(contentId)
 → AJAX fetch JSP
 → .page-content'e inject
```

// Eklenti için sonuç: URL değişmez, DOM değişir

// MutationObserver ŞART!

#### 2. Content Script Sayfa Değişikliği Tespiti

typescript

```
// src/content/page-detector.ts
```

```
type UyapPage = 'login' | 'home' | 'dosya-sorgulama' | 'dosya-detay' | 'other';
```

```
const MENU_ID_MAP: Record<string, UyapPage> = {
 '12573': 'dosya-sorgulama', // Sorgulama İşlemleri
 '12572': 'home', // İşlemlerim
};
```

```
export function detectCurrentPage(): UyapPage {
 // URL'den menuId kontrolü
 const url = new URL(location.href);
 const menuId = url.searchParams.get('menuId');

 if (menuId && MENU_ID_MAP[menuId]) {
 return MENU_ID_MAP[menuId];
 }

 // DOM'dan modal kontrolü
 if (document.querySelector('.modal-dialog.modal-lg ul#browser.filetree')) {
 return 'dosya-detay';
 }
```

```

}

// Login sayfası kontrolü
if (location.pathname.includes('giris.jsp')) {
 return 'login';
}

return 'other';
}

/***
 * SPA navigasyonunu dinle
 */
export function observePageChanges(callback: (page: UyapPage) => void): MutationObserver {
let lastPage = detectCurrentPage();

const observer = new MutationObserver(() => {
 const currentPage = detectCurrentPage();
 if (currentPage !== lastPage) {
 lastPage = currentPage;
 callback(currentPage);
 }
});

// .page-content UYAP'ın içerik inject ettiği alan
const target = document.querySelector('.page-content') || document.body;

observer.observe(target, {
 childList: true,
 subtree: true
});

return observer;
}

3. Modal Açıılma Tespiti (Dosya Detay)
typescript
// Modal açıldığında ekleni UI'ı göster
export function observeModalOpen(callback: () => void): MutationObserver {
const observer = new MutationObserver((mutations) => {
 for (const mutation of mutations) {
 for (const node of mutation.addedNodes) {
 if (node instanceof HTMLElement) {
 // UYAP dosya detay modalı açıldı mı?
 if (node.matches?('.modal-dialog.modal-lg') ||
 node.querySelector?('.modal-dialog.modal-lg ul#browser.filetree')) {
 callback();
 return;
 }
 }
 }
 }
})

```

```

 }
});

observer.observe(document.body, {
 childList: true,
 subtree: true
});

return observer;
}

```

### Teknoloji Uyumluluk Matrisi

| UYAP Tech   | Versiyon | Eklenti Riski                              | Çözüm                            |
|-------------|----------|--------------------------------------------|----------------------------------|
| jQuery      | 1.11.2   | <span style="color: yellow;">●</span> Orta | Preact kullan, jQuery'ye dokunma |
| Bootstrap   | 3.3.5    | <span style="color: red;">●</span> Yüksek  | uyap- prefix zorunlu             |
| DataTables  | 1.10.16  | <span style="color: green;">●</span> Düşük | Müdahale yok                     |
| ContextMenu | -        | <span style="color: yellow;">●</span> Orta | Event propagation dikkat         |
| jsTree      | -        | <span style="color: green;">●</span> Düşük | Sadece oku, değiştirme           |

### Güncellenmiş Content Script Init

typescript

```
// src/content/content.ts
```

```

import { detectCurrentPage, observePageChanges, observeModalOpen } from './page-detector';

function init() {
 const page = detectCurrentPage();

 // Login sayfasında hiçbir şey yapma
 if (page === 'login') {
 console.log('[UYAP Ext] Login sayfası, bekleniyor...');
 return;
 }

 // Sayfa değişikliklerini dinle (SPA navigasyon)
 observePageChanges((newPage) => {
 console.log('[UYAP Ext] Sayfa değişti:', newPage);
 handlePageChange(newPage);
 });

 // Modal açılalarını dinle
 observeModalOpen(() => {
 console.log('[UYAP Ext] Dosya detay modalı açıldı');
 injectDownloadUI();
 });

 // İlk yükleme
}

```

```

handlePageChange(page);
}

function handlePageChange(page: UyapPage) {
 switch (page) {
 case 'dosya-sorgulama':
 // Sorgulama sayfası UI'ı
 break;
 case 'dosya-detay':
 injectDownloadUI();
 break;
 }
}

// DOM hazır olduğunda başlat
if (document.readyState === 'loading') {
 document.addEventListener('DOMContentLoaded', init);
} else {
 init();
}

```

## Hata Analizi

### 1. Synchronous XMLHttpRequest (pace.min.js)

Deprecation Warning - UYAP'ın sorunu

Ne: Pace.js progress bar kütüphanesi senkron XHR kullanıyor (eski yöntem) Etkisi: Sayfa donabilir, UX kötüleşir Sana etkisi: Yok – UYAP'ın kendi kütüphanesi

### 2. Unload Event Listeners (chatbot-min.js)

Deprecation Warning - UYAP'ın sorunu

Ne: Chatbot window.onunload veya beforeunload kullanıyor Etkisi: Chrome bu event'i kaldıracak (bfcache uyumluluğu için) Sana etkisi: Yok – UYAP chatbot'unun sorunu

### 3. CORS Hataları (Socket.IO Polling)

Blocked - UYAP sunucu konfigürasyon hatası

Ne oluyor:

vatandas.uyap.gov.tr → vatandaschatbot.adalet.gov.tr

↑

↓

Origin Access-Control-Allow-Origin: YOK

Detay:

- ?EIO=4&transport=polling = Socket.IO Engine.IO v4 polling istekleri
- Chatbot farklı domain'de (adalet.gov.tr)
- Sunucu Access-Control-Allow-Origin header'i göndermiyor
- Browser cross-origin isteği engelliyor

Sana etkisi: Yok – Chatbot çalışmıyor olabilir ama eklientini etkilemez

## Özeti

| Hata     | Kaynak             | Eklentiye Etkisi |
|----------|--------------------|------------------|
| Sync XHR | pace.min.js (UYAP) | Yok              |

| Hata         | Kaynak                 | Eklentiye Etkisi |
|--------------|------------------------|------------------|
| Unload event | chatbot-min.js (UYAP)  | ✗ Yok            |
| CORS blocked | vatandaschatbot (UYAP) | ✗ Yok            |

**Sonuç:** Bunlar UYAP'ın teknik borçları. Eklentin için endişelenme, bunları görmezden gelebilirsin.

İşte vatandas.uyap bağlamı – Adım 4: Dosya Sorgulama Sayfası: vatandas.uyap Bağlam – Adım 4: Dosya Sorgulama URL: vatandas.uyap.gov.tr/main/jsp/vatandas/index.jsp?menuId=12573 SORGULAMA FORMU: Yargı Türü dropdown'ı (#yargiTuru): Ceza (0), Hukuk (1), İcra (2), Adli Tip (5), İdari Yargı (6), Satış Memurluğu (11), Arabuluculuk (25), Tazminat Komisyonu (26) Mahkeme dropdown'ı (#mahkeme): Tümü + kullanıcının dosyası olan mahkemeler (Select2 ile) Sorgulama Tab'ları: Genel Sorgula, Esas No İle Sorgula, Açıılış Tarihi İle Sorgula, Kapanış Tarihi İle Sorgula Sorgulama AJAX isteği:

POST vatandas\_dosyalari\_sorgula.ajax

Params: { yargiTuru: "0", mahkeme: "Tümü", ... }

Response: HTML (DataTables tablosu)

**SONUÇ TABLOSU:** #dosya\_listesi\_table (DataTables) Kolonlar: Taraf Rolü, Mahkeme Adı, Esas No, Dosya Türü, Dosya Açıılış Tarihi, Dosya Durumu, Dosya Görüntüle Her satırda "Dosya Görüntüle" butonu → DosyaBilgileriGoruntuleme.baslat(data) çağrılmıyor DOSYA GÖRÜNTÜLEME AKIŞI (Kritik!):

"Dosya Görüntüle" tıklanır

- DosyaBilgileriGoruntuleme.baslat(dosyaBilgileriJson)
- dosyaBilgileriSet(json) → dosya\_bilgileri global değişkenine atar
- evrakSolClickMenu() → context menu'yu bağlar
- #dosya\_goruntule\_modal açılır (Bootstrap modal)

"Evrak" tab'ı tıklanır

- DosyaBilgileriGoruntuleme.dosyaEvrakBilgileri()
- sendToServer2('dosya\_evrak\_bilgileri\_brd', {dosyaId: dosya\_bilgileri.dosyaId}, callback)
- callback(html) → \$('#dosya\_evrak\_bilgileri\_result').html(html)
- Filetree (ul#browser.filetree) render edilir

UYAP SUNUCU İLETİŞİM PATTERN'I:

```
sendToServer2(endpointName, params, callbackFn)
 → jQuery.ajax POST
 → Endpoint: "{endpointName}" (ör: dosya_evrak_bilgileri_brd)
 → Response: HTML fragment
 → callback(html) ile DOM'a inject
```

Tespit edilen \_brd endpoint'leri: dosya\_ayrinti\_bilgileri\_brd, dosya\_taraf\_bilgileri\_brd, dosya\_evrak\_bilgileri\_brd, dosya\_safahat\_bilgileri\_brd, dosya\_evrak\_turu\_brd, download\_document\_brd (indirme endpoint'i) SOL MENÜ YAPISI (Tam):

- |— Kişisel Bilgiler (12568)
  - | |— İsim ve Adres Bilgileri (12570)
  - | |— İletişim Bilgilerim (13828)
  - | |— Fotoğraf Bilgileri (12569)
  - | |— E-Tebliğat Bilgileri (12571)
  - | |— Mobil Oturum Bilgileri (20480)
- |— UYAP İşlemlerim (30209)
  - | |— İşlemlerim (12572)
  - | |— Ödeme İşlemlerim (13827)
  - | |— Dosya Gizleme (...)
  - | |— ...
- |— Başvuru Yap (18944)
- |— Dava Açıılış İşlemleri (26113)
- |— Sorulama İşlemleri (15360) ← AKTİF
  - | |— Anayasa Mahkemesi Bireysel Başvuru (22016)
  - | |— Dosya Sorulama (12573) ← AKTİF

```

| └── Danıştay Dosya Sorgulama (28934)
| └── Duruşma Sorgulama (12577)
| └── İcra Daireleri İban Sorgulama (28161)
└── Duyurular (28160)
 ├── Genel Duyuru (12585)
 └── Muhafazasına Gerek Kalmayan Mallar (30720)

```

EKLENTİ İÇİN YENİ BULGULAR: 1. Eklenti aktivasyon noktası: menuId=12573 (Dosya Sorgulama) sayfasında veya dosya\_goruntule\_modal açıldığında aktif olmalı. MutationObserver ile modal açılmasını dinle. 2. dosya\_bilgileri global değişkeni: Modal açıldığında dosya\_bilgileri.dosyaId otomatik olarak set ediliyor – eklentimiz bunu doğrudan okuyabilir, DOM'dan regex ile aramaya gerek yok! 3. sendToServer2 fonksiyonu: UYAP'in kendi AJAX wrapper'i. Eklentimiz bunu kullanmak yerine doğrudan fetch() kullanmalı – UYAP'in iç mekanizmasına bağımlı olmamalı. 4. Evrak listesi HTML olarak geliyor: dosya\_evrak\_bilgileri\_brd endpoint'i HTML döndürüyor, \$('#dosya\_evrak\_bilgileri\_result').html(html) ile inject ediliyor. Eklentimiz bu inject sonrasını bekleyip DOM'dan span.file[evrak\_id] ile tarama yapıyor – doğru yaklaşım.

Show more

16:14

Analiz etti, kritik bulguları kaydetti, eklenti entegrasyonunu planladı

Mükemmel! Bu en kritik sayfa. Kaydedildi:

#### UYAP Teknik Gözlemler - Sayfa 4/N

| Sayfa           | URL                     | Eklenti Durumu    |
|-----------------|-------------------------|-------------------|
| Giriş           | /giris.jsp              | ✗ Pasif           |
| e-Devlet        | giris.turkiye.gov.tr    | ✗ Kapsam dışı     |
| Ana Sayfa       | /index.jsp              | ✓ Aktif (dinleme) |
| Dosya Sorgulama | /index.jsp?menuId=12573 | ✓ ANA SAYFA       |

#### Kritik Keşifler

##### 1. Global Değişken Erişimi (Oyun Değiştirici!)

typescript

```
// UYAP zaten dosya bilgisini global'e atıyor!
```

```
declare global {
```

```
 interface Window {
 dosya_bilgileri?: {
 dosyaId: string;
 esasNo: string;
 mahkemeAdi: string;
 dosyaTuru: string;
 // ... diğer alanlar
 };
 }
}
```

```
// Eklentide kullanım - DOM parsing'e gerek YOK!
```

```
function getDosyaId(): string | null {
 return window.dosya_bilgileri?.dosyaId ?? null;
}
```

##### 2. UYAP Endpoint Haritası

typescript

```
// src/config/uyap-endpoints.ts
```

```
export const UYAP_ENDPOINTS = {
```

```

// Sorgulama
DOSYA_SORGULA: 'vatandas_dosyalari_sorgula.ajax',

// Dosya Detay (_brd = "bilgi render" tahminim)
DOSYA_AYRINTI: 'dosya_ayrinti_bilgileri_brd',
DOSYA_TARAF: 'dosya_taraf_bilgileri_brd',
DOSYA_EVRAK: 'dosya_evrap_bilgileri_brd', // ← Evrak ağacı
DOSYA_SAFAHAT: 'dosya_safahat_bilgileri_brd',
EVRAK_TURU: 'dosya_evrap_turu_brd',

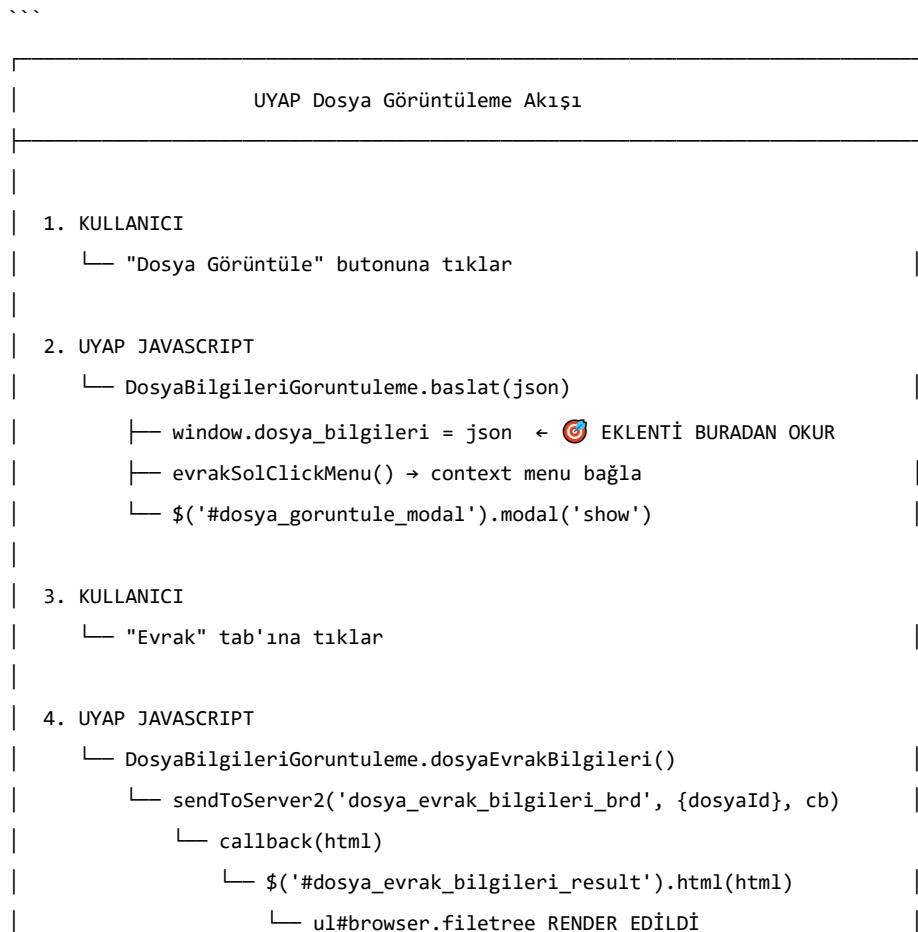
// İndirme
DOWNLOAD: 'download_document_brd.uyap',

// Viewer
VIEWER: '/main/jsp/avukat/evrak_viewer.jsp'
} as const;

3. Evrak İndirme URL Pattern'i
typescript

// Daha önce tespit ettiğimiz pattern doğrulandı:
function buildDownloadUrl(evrapId: string, dosyaId: string, yargiTuru: string = '1'): string {
 return `download_document_brd.uyap?evrapId=${evrapId}&dosyaId=${dosyaId}&yargiTuru=${yargiTuru}`;
}
```
---
```

📡 Güncellenmiş Akış Diyagramı



```
|  
| 5. EKLENTİ (MutationObserver)  
|    └ Filetree DOM'a eklendi tespit  
|        └ dosyaId = window.dosya_bilgileri.dosyaId  
|        └ parseUyapTreeView() → evrak listesi  
|        └ İndirme UI'ı göster  
|  
|
```

Güncellenmiş Modal Observer

typescript

// src/content/modal-observer.ts

```
interface DosyaBilgileri {  
    dosyaId: string;  
    esasNo: string;  
    mahkemeAdi: string;  
    dosyaTuru: string;  
    yargiTuru: string;  
}  
  
/**  
 * UYAP'in global değişkeninden dosya bilgilerini al  
 */  
export function getDosyaBilgileri(): DosyaBilgileri | null {  
    const db = (window as any).dosya_bilgileri;  
    if (!db?.dosyaId) return null;  
  
    return {  
        dosyaId: db.dosyaId,  
        esasNo: db.esasNo || '',  
        mahkemeAdi: db.mahkemeAdi || '',  
        dosyaTuru: db.dosyaTuru || '',  
        yargiTuru: db.yargiTuru || '1'  
    };  
}  
  
/**  
 * Evrak tab'ının yüklenip filetree'nin DOM'a eklenmesini bekle  
 */  
export function observeEvrakTabLoad(callback: (dosyaBilgileri: DosyaBilgileri) => void): MutationObserver {  
    const observer = new MutationObserver((mutations) => {  
        for (const mutation of mutations) {  
            for (const node of mutation.addedNodes) {  
                if (!(node instanceof HTMLElement)) continue;  
  
                // Filetree eklendi mi?  
                const filetree = node.matches?('.ul#browser.filetree')  
                    ? node  
                    : node.querySelector?('.ul#browser.filetree');  
            }  
        }  
    });  
    observer.observe(document.body, { childList: true, subtree: true });  
    callback({ dosyaBilgileri: {} });  
}
```

```

        if (filetree) {
            const dosyaBilgileri = getDosyaBilgileri();
            if (dosyaBilgileri) {
                console.log('[UYAP Ext] Evrak listesi yüklendi, dosyaId:', dosyaBilgileri.dosyaId);
                callback(dosyaBilgileri);
            }
            return;
        }
    }
});

// Modal içeriğini dinle
const modalBody = document.querySelector('#dosya_goruntule_modal .modal-body')
|| document.body;

observer.observe(modalBody, {
    childList: true,
    subtree: true
});

return observer;
}

/***
 * Modal kapanışını dinle (cleanup için)
 */
export function observeModalClose(callback: () => void): void {
    // Bootstrap 3 modal event
    $(document).on('hidden.bs.modal', '#dosya_goruntule_modal', () => {
        callback();
    });
}

```

Eklenti Aktivasyon Mantığı (Güncel)

typescript

// src/content/content.ts

```

import { observeEvrakTabLoad, observeModalClose, getDosyaBilgileri } from './modal-observer';
import { parseUyapTreeView, flattenTree } from '../services/document-service';
import { injectDownloadPanel } from '../ui/download-panel';

let evrakObserver: MutationObserver | null = null;

function init() {
    // Login sayfasında çalışma
    if (location.pathname.includes('giris.jsp')) {
        return;
    }
}

```

```

console.log('[UYAP Ext] Content script yüklendi');

// Evrak tab yüklenliğinde
evrakObserver = observeEvrakTabLoad((dosyaBilgileri) => {
    // 1. Tree'yi parse et
    const tree = parseUyapTreeView();
    if (!tree) {
        console.warn('[UYAP Ext] Filetree parse edilemedi');
        return;
    }

    // 2. Dosyaları düz listeye çevir
    const files = flattenTree(tree);
    console.log(`[UYAP Ext] ${files.length} dosya bulundu`);

    // 3. İndirme UI'ı inject et
    injectDownloadPanel({
        dosyaBilgileri,
        files,
        tree
    });
});

// Modal kapandığında cleanup
observeModalClose(() => {
    console.log('[UYAP Ext] Modal kapandı, cleanup');
    // UI'ı kaldır, state'i resetle
});
}

// Başlat
if (document.readyState === 'loading') {
    document.addEventListener('DOMContentLoaded', init);
} else {
    init();
}

```

vatandas.uyap'in tespit ettiğim teknolojik alt yapısı: SUNUCU TARAFI (Backend) Teknoloji olarak Java EE (Enterprise Edition) kullanılıyor. Sayfalar .jsp (Java Server Pages) ile render ediliyor, AJAX endpoint'leri .ajax ve .uyap uzantılı (ör: vatandas_dosyaları_sorgula.ajax, download_document_brd.uyap). Session yönetimi standart Java Servlet JSESSIONID cookie'si ile yapılıyor. TS01953787 cookie'si muhtemelen F5 BIG-IP load balancer veya WAF (Web Application Firewall) izini. Sunucu tarafında iş mantığı _brd (Bridge) pattern'ı ile adlandırılmış – sendToServer2('dosya_evrak_bilgileri_brd', params, callback) gibi çağrılar yapılıyor, bu da muhtemelen arka planda Java Servlet veya Spring Controller'lara map ediliyor. İSTEMCİ TARAFI (Frontend) Temel framework olarak jQuery 1.11.2 kullanılıyor, üstüne jQuery UI 1.11.2 eklenmiş. Sayfa düzeni ve bileşenler Bootstrap 3.3.5 ile yapılmış, tema olarak Metronic admin template'i kullanılıyor (Metronic.blockUI gibi çağrılarından tespit ettik). Tablo listeleme için DataTables 1.10.16 (Buttons, Select, ColReorder extension'larıyla birlikte), dosya ağacı için jQuery Treeview, sağ tık menüsü için jQuery ContextMenu, gelişmiş dropdown'lar için Select2, ağaç yapısı için jsTree, template rendering için Handlebars, tarih işleme için Moment.js, zengin metin editörü için CKEditor, loading overlay için jQuery BlockUI, sayfa yükleme progress bar'ı için Pace.js, sayfalama için jQuery Bootpag kullanılıyor. MİMARI PATTERN Sayfa navigasyonu SPA benzeri çalışıyor: menuButtonClick(menuId) → Application.openContentPage() → sendToServer2() ile AJAX POST → sunucu HTML fragment döndürülüyor → .page-content içine inject ediliyor. Modal'lar (dosya görüntüleme) Bootstrap modal olarak açılıyor, her tab'in içeriği ayrı _brd endpoint'inden lazy-load ediliyor. Angular/React/Vue gibi modern framework yok – tamamen jQuery + server-rendered HTML mimarisi. Kısaca: Java EE + JSP backend, jQuery + Bootstrap 3 + Metronic tema + çok sayıda jQuery plugin frontend. Eski ama sağlam bir enterprise Java mimarisi.

Show more

Mükemmel analiz! Bunu referans doküman olarak kaydediyorum:

UYAP Teknoloji Stack'i (Referans Doküman)

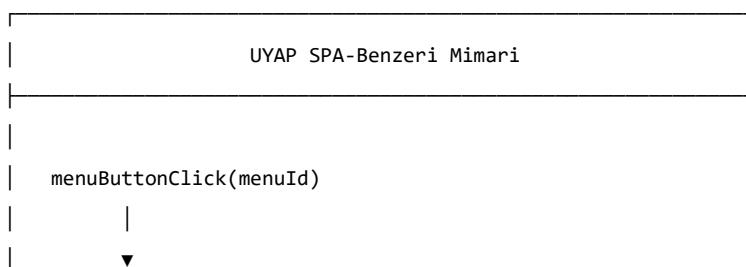
Backend (Sunucu)

Katman	Teknoloji	Detay
Platform	Java EE	Enterprise Java
View	JSP	.jsp dosyaları
AJAX	Custom	.ajax, .uyap uzantıları
Pattern	Bridge	*_brd endpoint adlandırması
Session	Servlet	JSESSIONID cookie
Infra	F5 BIG-IP	TS01953787 cookie (LB/WAF)

Frontend (istemci)

Kategori	Kütüphane	Versiyon
Core	jQuery	1.11.2
	jQuery UI	1.11.2
Layout	Bootstrap	3.3.5
	Metronic	Admin tema
Data	DataTables	1.10.16
Tree	jQuery Treeview	-
	jsTree	-
UI	Select2	Dropdown
	ContextMenu	Sağ tık
	BlockUI	Loading overlay
	Pace.js	Progress bar
Utils	Bootpag	Pagination
	Moment.js	Tarih
	Handlebars	Template
	CKEditor	Rich text

Mimari Pattern



```

| Application.openContentPage(contentId)
| |
| |
| | ▼
| sendToServer2(endpoint_brd, params, callback)
| |
| | ▼ POST (jQuery.ajax)
| |
| | | Java EE Backend
| | | Servlet/Spring → JSP Fragment
| |
| | |
| | | ▼ Response: HTML
| callback(html)
| |
| | ▼
| |
| | | $('.page-content').html(html)
|
| URL DEĞİŞMİYOR! → MutationObserver ŞART
|

```

⚠ Eklenti Geliştirme Riskleri & Çözümler

Risk	Sebep	Çözüm
jQuery çakışması	UYAP jQuery 1.11.2 kullanıyor	✗ jQuery kullanma, ✓ Preact + vanilla JS
CSS çakışması	Bootstrap 3.3.5 + Metronic	✓ uyap- prefix zorunlu
Event çakışması	ContextMenu, Select2 event'leri	✗ Var olan event'lere dokunma
SPA navigasyon	URL değişmiyor, DOM değişiyor	✓ MutationObserver
Lazy loading	Tab içerikleri sonradan yükleniyor	✓ Filetree için ayrı observer
Global değişkenler	dosya_bilgileri, sendToServer2	✓ Okuyabilirsin ama bağımlı olma

▣ Eklenti Tasarım Kuralları (Güncel)

typescript

```
// ✗ YAPMA
import $ from 'jquery'; // jQuery kullanma
document.querySelector('.btn') // UYAP class'larını hedefleme
window.sendToServer2(...) // UYAP fonksiyonlarına bağımlılık
```

// ✓ YAP

```
import { h, render } from 'preact'; // Preact kullan
document.querySelector('.uyap-ext-btn') // Kendi prefix'in
fetch('/endpoint', { credentials: 'include' }) // Native fetch + session cookie
```

Analyzed UYAP XML response structure and parsing requirements

İlginc bulgu! UYAP'ın bazı endpoint'leri XML döndürüyor, HTML değil:

📊 UYAP Response Format Analizi

Endpoint Türü	Response	Örnek
*_brd	HTML fragment	dosya_evrap_bilgileri_brd
*.ajx	XML	Duyurular, listeler
*.uyap	Binary/Mixed	download_document_brd.uyap

XML Yapısı

```
xml
<root>
  <list>
    <DuyuruDVO>          <!-- DVO = Data Value Object (Java pattern) -->
      <ID>97280</ID>
      <baslik>...</baslik>
      <icerik>HTML (escaped)</icerik> <!-- &lt;p&gt; şeklinde -->
      <baslangicTarih>2025-10-09 15:19:52</baslangicTarih>
    </DuyuruDVO>
  </list>
</root>

Tarih formatı: YYYY-MM-DD HH:mm:ss (Java SimpleDateFormat)
```

Eklenti İçin Not

Eğer duyuru veya liste endpoint'lerini kullanacaksan:

typescript

```
// XML parse helper

function parseUyapXml<T>(xmlText: string, itemTag: string): T[] {
  const parser = new DOMParser();
  const doc = parser.parseFromString(xmlText, 'text/xml');
  const items = doc.querySelectorAll(itemTag);

  return Array.from(items).map(item => {
    const obj: Record<string, string> = {};
    for (const child of item.children) {
      obj[child.tagName] = child.textContent || '';
    }
    return obj as T;
  });
}
```

// Kullanım

```
const duyurular = parseUyapXml<DuyuruDVO>(responseText, 'DuyuruDVO');
```

Ama evrap indirme akışı için bu XML endpoint'leri gerekli değil – dosya_evrap_bilgileri_brd HTML döndürüyor ve DOM'dan parse ediyoruz.

İşte vatandas.uyap bağlımı – Adım 5: Dosya Görüntüleme Modal'ı + Evrap Sekmesi: vatandas.uyap Bağlam – Adım 5: Dosya Görüntüleme Modal'ı Modal ID: #dosya_goruntule_modal (Bootstrap .modal-dialog.modal-lg) MODAL AÇILIŞ AKIŞI:

"Dosya Görüntüle" butonu tıklanır

- DosyaBilgileriGoruntuleme.baslat(dosyaBilgileriJson)
- dosya_bilgileri global değişkenine atanır
- Modal açılır

dosya_bilgileri Global Nesne (kritik!): KeyTipAçıklamadosyaIdstringİndirme endpoint'inde kullanılacak dosya IDdosyaNostringDosya numarası (ör: 2021/612)birimIdstringMahkeme birim IDbirimAdistringMahkeme adıdosyaTurKodstringDosya

türü kodu (ör: "15") yarlıTuru string Yargı türü (boş olabilir, select2'den alınmalı) tabList array[16] Tüm tab listesi gösterilecek tabList array[7] Görüntülenecek tab'lar dosyaDurumu string Kapalı/Açık vb. EVRAK SEKMESİ YÜKLENME AKIŞI:

"Evrak" tab'ına tıklanır

```
→ DosyaBilgileriGoruntuleme.dosyaEvrakBilgileri()
→ Metronic.blockUI() (loading overlay)
→ sendToServer2('dosya_evrak_bilgileri_brd', {dosyaId: dosya_bilgileri.dosyaId}, callback)
```

Network İsteği:

POST dosya_evrak_bilgileri_brd.ajax

Body: { dosyaId: "{dosyaId}" }

Response: HTML fragment (filetree yapısı)

```
→ callback(html) → $('#dosya_evrak_bilgileri_result').html(html)
```

Filetree Yapısı (yüklenmekten sonra): * 240 dosya (span.file[evrak_id]), 46 klasör (span.folder) * Kök: "İsparta 2. Aile Mahkemesi 2021/612" * "Dosyaya Eklenen Son 20 Evrak" (20 dosya) * "Tüm Evraklar" (220 dosya) * Alt klasörler: evrak türüne göre (Tensip Zaptı, Kapalı Tebligat, vb.) CONTEXT MENU (Sağ Tık):

span.file contextmenu event → cmenu.show(this, e)

```
→ .context-menu.context-menu-theme-vista gösterilir
→ "Görüntüle" (click) → viewer.jsp açar
→ "Kaydet" (click) → downloadDoc(evrakId, dosyaId, yargiTuru)
```

İNDİRME ZİNCİRİ (UYAP'ın Kendi Mekanizması):

downloadDoc(evrakId, dosyaId, yargiTuru)

```
→ downloadDocCustom({evrakId, dosyaId, yargiTuru})
→ url = getDownloadURL(params) // URL: download_document_brd.uyap?evrakId=X&dosyaId=Y&yargiTuru=Z
→ downloadDocURL(url, values)
→ createElement('a') + createEvent('click') + dispatchEvent
→ Fallback: postToURL_target (form POST)
```

UYAP'ın kendisi de programatik olarak <a> elementi oluşturup tıklama simülasyonunu ediyor. Bizim eklentideki fetch() + Base64 + chrome.downloads yaklaşımı bundan bağımsız ve daha kontrollü çalışıyor – doğru tercih. EKLENTİ İÇİN YENİ BULGULAR: 1. dosya_bilgileri.dosyaId doğrudan erişilebilir – DOM'dan regex ile aramaya gerek yok. Ama yargiTuru boş olabiliyor, #yargiTuru select'inden alınmalı. 2. Evrak listesi lazy-load: "Evrak" tab'ına tıklanana kadar filetree yüklenmiyor. Eklenti, filetree'nin DOM'a inject edilmesini MutationObserver ile beklemeli. 3. sendToServer2 → .ajax endpoint: UYAP tüm AJAX isteklerini {endpointName}.ajax olarak yapıyor. İndirme ise download_document_brd.uyap üzerinden (farklı uzantı).

Synthesized UYAP modal architecture and download mechanism details

Mükemmel detay! Bu modal+evrak akışı eklentinin kalbi. Kaydediyorum:

UYAP Teknik Gözlemler - Sayfa 5/N

dosya_bilgileri Global Nesne (Tam Yapı)

typescript

```
// src/types/uyap.ts
```

interface DosyaBilgileri {

```
  dosyaId: string;           // ✅ Kritik - indirme için
  dosyaNo: string;           // "2021/612"
  birimId: string;           // Mahkeme ID
  birimAdi: string;          // "İsparta 2. Aile Mahkemesi"
  dosyaTurKod: string;        // "15"
  yargiTuru: string;          // ⚠ BOŞ OLABİLİR!
  dosyaDurumu: string;        // "Kapalı" / "Açık"
  tabList: TabInfo[];          // [16] tüm tab'lar
  gosterilecekTabList: TabInfo[]; // [7] görünen tab'lar
}
```

```
// Global'den güvenli okuma
```

```
declare global {
```

```
interface Window {  
    dosya_bilgileri?: DosyaBilgileri;  
}  
}
```

⚠ Kritik: `yargiTuru` Fallback

typescript

// `src/utils/uyap-helpers.ts`

```
/**  
 * yargiTuru'nu güvenli şekilde al  
 * Global'de boşsa Select2'den oku  
 */  
export function getYargiTuru(): string {  
    // 1. Global'den dene  
    const fromGlobal = window.dosya_bilgileri?.yargiTuru;  
    if (fromGlobal && fromGlobal.trim() !== '') {  
        return fromGlobal;  
    }  
  
    // 2. Select2'den oku (Dosya Sorgulama sayfasındaki dropdown)  
    const select = document.querySelector<HTMLSelectElement>('#yargiTuru');  
    if (select?.value) {  
        return select.value;  
    }  
  
    // 3. Default: Hukuk (en yaygın)  
    console.warn('[UYAP Ext] yargiTuru bulunamadı, default=1 (Hukuk)');  
    return '1';  
}  
  
/**  
 * Download URL oluştur  
 */  
export function buildDownloadUrl(evvakId: string): string {  
    const dosyaId = window.dosya_bilgileri?.dosyaId;  
    if (!dosyaId) {  
        throw new Error('dosyaId bulunamadı');  
    }  
  
    const yargiTuru = getYargiTuru();  
  
    return `download_document_brd.uyap?evvakId=${evvakId}&dosyaId=${dosyaId}&yargiTuru=${yargiTuru}`;  
}  
...  
---
```

🌳 Dosya Ağacı Yapısı (Gerçek Örnek)

...

```

📁 Isparta 2. Aile Mahkemesi 2021/612
└── 📁 Dosyaya Eklenen Son 20 Evrak    ← ⚠ DUPLIKASYON RİSKİ
|   └── 📄 Tensip Zaptı (evrak_id=123)
|   └── 📄 Dava Dilekçesi (evrak_id=124)
|   └── ... (20 dosya)
|
└── 📁 Tüm Evraklar      ← ✅ ANA KAYNAK
    ├── 📁 Tensip Zaptı
    |   └── 📄 Tensip Zaptı (evrak_id=123) ← Aynı evrak_id
    ├── 📁 Kapalı Tebliğat
    |   └── 📄 Tebliğat (evrak_id=125)
    ├── 📁 Dilekçeler
    |   └── 📄 Dava Dilekçesi (evrak_id=124) ← Aynı evrak_id
    |   └── 📄 Cevap Dilekçesi (evrak_id=126)
    └── ... (46 klasör, 220 dosya)

```

Toplam: 240 span.file, ama 20'si duplike → gerçek evrak sayısı ~220

⚠ Duplikasyon Problemi & Çözüm

typescript

```
// src/services/document-service.ts
```

```

interface ParseOptions {
  skipDuplicateFolder?: boolean; // "Son 20 Evrak" klasörünü atla
  deduplicateByEvrakId?: boolean; // Aynı evrak_id'yi tekrar ekleme
}

export function flattenTree(
  tree: TreeNode,
  options: ParseOptions = {}
): EvrakFile[] {
  const files: EvrakFile[] = [];
  const seenEvrakIds = new Set<string>();

  function traverse(node: TreeNode, path: string) {
    // "Son 20 Evrak" klasörünü atla (opsiyonel)
    if (options.skipDuplicateFolder &&
        node.type === 'folder' &&
        node.name.includes('Son 20 Evrak')) {
      console.log('[UYAP Ext] "Son 20 Evrak" atlandı (duplikasyon önleme)');
      return;
    }

    if (node.type === 'file' && node.evrakId) {
      // Duplikasyon kontrolü
      if (options.deduplicateByEvrakId && seenEvrakIds.has(node.evrakId)) {
        return; // Zaten eklendi
      }
      seenEvrakIds.add(node.evrakId);
    }
  }
}

```

```

    files.push({
      evrakId: node.evrakId,
      name: node.name,
      path: path,
      metadata: node.metadata
    });
  }

  for (const child of node.children || []) {
    const childPath = node.type === 'folder'
      ? `${path}/${node.name}`
      : path;
    traverse(child, childPath);
  }
}

traverse(tree, '');
return files;
}

```

Endpoint Özeti (Güncel)

Endpoint	Method	Request	Response	Kullanım
dosya_evrak_bilgileri_brd.ajax	POST	{dosyaId}	HTML	Evrak ağacı
download_document_brd.uyap	GET	?evrakId&dosyaId&yargiTuru	Binary	Dosya indir
evrak_viewer.jsp	GET	?mimeType&evrakId&dosyaId&yargiTuru	HTML	Evrak görüntüle

Güncellenmiş MutationObserver

```

typescript
// src/content/observers.ts

/**
 * Evrak tab'ı yüklendiğinde filetree'yi yakala
 */
export function observeEvrakTab(onFiletreeReady: (files: EvrakFile[]) => void): MutationObserver {
  const observer = new MutationObserver((mutations) => {
    for (const mutation of mutations) {
      for (const node of mutation.addedNodes) {
        if (!(node instanceof HTMLElement)) continue;

        // Filetree inject edildi mi?
        const filetree = node.querySelector?('.ul#browser.filetree');
        if (!filetree) continue;

        // Parse et
        const tree = parseUyapTreeView();
        if (!tree) continue;

        // Flatten (duplikasyon önleme aktif)
      }
    }
  });
}

```

```

const files = flattenTree(tree, {
  skipDuplicateFolder: true,
  deduplicateByEvrakId: true
});

console.log(`[UYAP Ext] ${files.length} benzersiz evrak bulundu`);
onFiletreeReady(files);

return; // Bir kez çalış
}
}
});

// #dosya_evrak_bilgileri_result'i dinle
const target = document.querySelector('#dosya_evrak_bilgileri_result')
  || document.querySelector('#dosya_goruntule_modal .modal-body')
  || document.body;

observer.observe(target, { childList: true, subtree: true });

return observer;
}
```

```

### ## 📁 Tam İndirme Akışı (Eklenti)

```

