PE Explorer Web (Go + React)

Aşağıda **Go backend** (PE analizi) ve **React frontend** (dosya yükleme + sonuçları gösterme) içeren, uçtan uca çalışan bir örnek var. Backend `debug/pe` ile PE (`.exe`, `.dll`) dosyasını çözümler ve JSON döndürür; Frontend ise dosyayı yükler ve sonuçları şık bir arayüzde listeler.

Klasör yapısı (önerilen)

```
pe-explorer-web/
├─ backend/
│  ├─ go.mod
│  └─ main.go
└─ frontend/
   ├─ package.json
   ├─ index.html
   └─ src/App.jsx
```

---

1) Backend (Go)

`backend/go.mod`

```
module pe-explorer-web/backend

go 1.22
```

`backend/main.go`

```go
package main

import (
    "debug/pe"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"
    "os"
    "path/filepath"
    "strings"
)

type PEInfo struct {
    FileHeader struct {
        Machine          uint16 `json:"machine"`
        NumberOfSections uint16 `json:"number_of_sections"`
        TimeDateStamp    uint32 `json:"time_date_stamp"`
        Characteristics  uint16 `json:"characteristics"`
    } `json:"file_header"`

    OptionalHeader any `json:"optional_header"`

    Sections          []SectionInfo `json:"sections"`
    ImportedLibraries []string      `json:"imported_libraries"`
    ImportedSymbols   []string      `json:"imported_symbols"`
    ExportedSymbols   []ExportInfo  `json:"exported_symbols"`
}

type SectionInfo struct {
    Name     string `json:"name"`
    VirtAddr uint32 `json:"virtual_address"`
    VirtSize uint32 `json:"virtual_size"`
    RawSize  uint32 `json:"raw_size"`
}

type ExportInfo struct {
    Ordinal uint16 `json:"ordinal"`
    Address uint32 `json:"address"`
}

func main() {
    mux := http.NewServeMux()
    mux.HandleFunc("/health", func(w http.ResponseWriter, r *http.Request) {
        w.WriteHeader(http.StatusOK)
        w.Write([]byte("ok"))
    })
    mux.HandleFunc("/api/pe/analyze", analyzePE)

    // Basit CORS sarmalayıcı
    h := withCORS(mux)
    addr := ":8080"
    log.Printf("PE Explorer backend listening on %s", addr)
    log.Fatal(http.ListenAndServe(addr, h))
}

func withCORS(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        w.Header().Set("Access-Control-Allow-Origin", "*")
        w.Header().Set("Access-Control-Allow-Methods", "POST, GET, OPTIONS")
        w.Header().Set("Access-Control-Allow-Headers", "Content-Type,
Authorization")
        if r.Method == http.MethodOptions {
            w.WriteHeader(http.StatusNoContent)
            return
        }
```

```go
        next.ServeHTTP(w, r)
    })
}

func analyzePE(w http.ResponseWriter, r *http.Request) {
    if r.Method != http.MethodPost {
        http.Error(w, "only POST is allowed", http.StatusMethodNotAllowed)
        return
    }

    // multipart/form-data'dan dosyayı al
    r.Body = http.MaxBytesReader(w, r.Body, 64<<20) // 64 MB limit
    if err := r.ParseMultipartForm(64 << 20); err != nil {
        http.Error(w, "invalid multipart form: "+err.Error(),
http.StatusBadRequest)
        return
    }
    file, header, err := r.FormFile("file")
    if err != nil {
        http.Error(w, "file not found in form: "+err.Error(),
http.StatusBadRequest)
        return
    }
    defer file.Close()

    // Geçici dosyaya yaz
    tmpDir := os.TempDir()
    name := sanitizeFilename(header.Filename)
    tmpPath := filepath.Join(tmpDir, name)
    out, err := os.Create(tmpPath)
    if err != nil {
        http.Error(w, "temp create error: "+err.Error(),
http.StatusInternalServerError)
        return
    }
    _, copyErr := io.Copy(out, file)
    cerr := out.Close()
    if copyErr != nil {
        http.Error(w, "write error: "+copyErr.Error(),
http.StatusInternalServerError)
        return
    }
    if cerr != nil {
        http.Error(w, "close error: "+cerr.Error(),
http.StatusInternalServerError)
        return
    }
    defer os.Remove(tmpPath)

    // PE dosyasını aç
    pf, err := pe.Open(tmpPath)
    if err != nil {
        http.Error(w, "pe open error: "+err.Error(), http.StatusBadRequest)
        return
    }
    defer pf.Close()

    info := PEInfo{}
    info.FileHeader.Machine = pf.FileHeader.Machine
    info.FileHeader.NumberOfSections = pf.FileHeader.NumberOfSections
    info.FileHeader.TimeDateStamp = pf.FileHeader.TimeDateStamp
    info.FileHeader.Characteristics = pf.FileHeader.Characteristics

    switch oh := pf.OptionalHeader.(type) {
    case *pe.OptionalHeader32:
        info.OptionalHeader = map[string]any{
            "is_64bit":    false,
            "entry_point": fmt.Sprintf("0x%x", oh.AddressOfEntryPoint),
            "image_base":  fmt.Sprintf("0x%x", oh.ImageBase),
            "subsystem":   oh.Subsystem,
        }
    case *pe.OptionalHeader64:
        info.OptionalHeader = map[string]any{
            "is_64bit":    true,
            "entry_point": fmt.Sprintf("0x%x", oh.AddressOfEntryPoint),
            "image_base":  fmt.Sprintf("0x%x", oh.ImageBase),
            "subsystem":   oh.Subsystem,
        }
    default:
        info.OptionalHeader = map[string]any{"note": "no optional header"}
    }

    for _, sec := range pf.Sections {
        info.Sections = append(info.Sections, SectionInfo{
            Name:     sec.Name,
            VirtAddr: sec.VirtualAddress,
            VirtSize: sec.VirtualSize,
            RawSize:  sec.Size,
        })
    }

    if libs, err := pf.ImportedLibraries(); err == nil {
        info.ImportedLibraries = libs
    }
    if funcs, err := pf.ImportedSymbols(); err == nil {
        info.ImportedSymbols = funcs
    }
    if pf.Export != nil {
        for _, ex := range pf.Export.Functions {
            info.ExportedSymbols = append(info.ExportedSymbols,
ExportInfo{Ordinal: ex.Ordinal, Address: ex.Address})
        }
    }

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(info)
}

func sanitizeFilename(s string) string {
    s = filepath.Base(s)
    s = strings.ReplaceAll(s, "..", "_")
    return s
}
```

---

2) Frontend (React)

Aşağıdaki minimal frontend, **Tailwind** + tek dosyalık bir React bileşeni olarak yazıldı. Dosya yükleyip `http://localhost:8080/api/pe/analyze` adresine gönderir ve sonuçları kartlar halinde gösterir. (Tailwind'i eklemek istemezsen temel HTML/CSS ile de çalışır.)

`frontend/package.json` (Vite + React için basit örnek)

```json
{
  "name": "pe-explorer-frontend",
  "private": true,
  "version": "0.0.1",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "vite": "^5.0.0"
  }
}
```

`frontend/index.html`

```html
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>PE Explorer Web</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/App.jsx"></script>
  </body>
</html>
```

`frontend/src/App.jsx`

```
import { useState } from 'react'

export default function App() {
  const [file, setFile] = useState(null)
  const [data, setData] = useState(null)
  const [loading, setLoading] = useState(false)
  const [error, setError] = useState('')

  const upload = async () => {
    if (!file) return
    setLoading(true)
    setError('')
    setData(null)
    try {
      const fd = new FormData()
      fd.append('file', file)
      const res = await fetch('http://localhost:8080/api/pe/analyze', {
        method: 'POST',
        body: fd,
      })
      if (!res.ok) throw new Error('HTTP ' + res.status)
      const json = await res.json()
      setData(json)
    } catch (e) {
      setError(String(e))
    } finally {
      setLoading(false)
    }
  }

  return (
    <div style={{ fontFamily: 'ui-sans-serif, system-ui', padding: 24,
maxWidth: 1100, margin: '0 auto' }}>
      <h1 style={{ fontSize: 28, fontWeight: 700, marginBottom: 12 }}>PE
Explorer Web</h1>
      <p style={{ marginBottom: 16 }}>Bir <code>.exe</code> ya da
<code>.dll</code> dosyası seçin ve analiz için yükleyin.</p>

      <div style={{ display: 'flex', gap: 12, alignItems: 'center',
marginBottom: 24 }}>
        <input type="file" onChange={(e) => setFile(e.target.files?.[0] ??
null)} />
        <button onClick={upload} disabled={!file || loading} style={{
padding: '8px 14px', borderRadius: 12, border: '1px solid #ddd', cursor: (!
file || loading) ? 'not-allowed' : 'pointer' }}>
          {loading ? 'Yükleniyor…' : 'Analiz Et'}
        </button>
      </div>

      {error && (
        <div style={{ background: '#fee2e2', border: '1px solid #fecaca',
padding: 12, borderRadius: 12, marginBottom: 16 }}>
          Hata: {error}
        </div>
      )}

      {data && (
        <div style={{ display: 'grid', gridTemplateColumns: '1fr', gap: 16 }}
>
          <section style={{ padding: 16, border: '1px solid #eee',
borderRadius: 16 }}>
            <h2 style={{ fontSize: 20, fontWeight: 700, marginBottom: 8 }}
>PE Header</h2>
            <pre style={{ whiteSpace: 'pre-wrap', margin: 0 }}
>{JSON.stringify(data.file_header, null, 2)}</pre>
          </section>

          <section style={{ padding: 16, border: '1px solid #eee',
borderRadius: 16 }}>
            <h2 style={{ fontSize: 20, fontWeight: 700, marginBottom: 8 }}
>Optional Header</h2>
            <pre style={{ whiteSpace: 'pre-wrap', margin: 0 }}
>{JSON.stringify(data.optional_header, null, 2)}</pre>
          </section>

          <section style={{ padding: 16, border: '1px solid #eee',
borderRadius: 16 }}>
            <h2 style={{ fontSize: 20, fontWeight: 700, marginBottom: 8 }}
>Sections</h2>
            {data.sections?.length ? (
              <div style={{ overflowX: 'auto' }}>
                <table style={{ width: '100%', borderCollapse: 'collapse' }}>
                  <thead>
                    <tr>
                      <th style={{ textAlign: 'left', borderBottom: '1px
solid #ddd', padding: 8 }}>Name</th>
                      <th style={{ textAlign: 'left', borderBottom: '1px
solid #ddd', padding: 8 }}>VirtualAddress</th>
                      <th style={{ textAlign: 'left', borderBottom: '1px
solid #ddd', padding: 8 }}>VirtualSize</th>
```

```
                      <th style={{ textAlign: 'left', borderBottom: '1px
solid #ddd', padding: 8 }}>RawSize</th>
                    </tr>
                  </thead>
                  <tbody>
                    {data.sections.map((s, i) => (
                      <tr key={i}>
                        <td style={{ borderBottom: '1px solid #f2f2f2',
padding: 8 }}>{s.name}</td>
                        <td style={{ borderBottom: '1px solid #f2f2f2',
padding: 8 }}>0x{(s.virtual_address || 0).toString(16)}</td>
                        <td style={{ borderBottom: '1px solid #f2f2f2',
padding: 8 }}>{s.virtual_size}</td>
                        <td style={{ borderBottom: '1px solid #f2f2f2',
padding: 8 }}>{s.raw_size}</td>
                      </tr>
                    ))}
                  </tbody>
                </table>
              </div>
            ) : (
              <div>Section yok.</div>
            )}
          </section>

          <div style={{ display: 'grid', gridTemplateColumns: '1fr 1fr',
gap: 16 }}>
            <section style={{ padding: 16, border: '1px solid #eee',
borderRadius: 16 }}>
              <h2 style={{ fontSize: 20, fontWeight: 700, marginBottom: 8 }}
>Imported Libraries</h2>
              {data.imported_libraries?.length ? (
                <ul style={{ margin: 0, paddingLeft: 18 }}>
                  {data.imported_libraries.map((lib, i) => <li key={i}>{lib}
</li>)}
                </ul>
              ) : (
                <div>Kayıt yok.</div>
              )}
            </section>

            <section style={{ padding: 16, border: '1px solid #eee',
borderRadius: 16 }}>
              <h2 style={{ fontSize: 20, fontWeight: 700, marginBottom: 8 }}
>Imported Symbols</h2>
              {data.imported_symbols?.length ? (
                <ul style={{ margin: 0, paddingLeft: 18, maxHeight: 240,
overflow: 'auto' }}>
                  {data.imported_symbols.map((fn, i) => <li key={i}>{fn}</
li>)}
                </ul>
              ) : (
                <div>Kayıt yok.</div>
              )}
            </section>
          </div>

          <section style={{ padding: 16, border: '1px solid #eee', 
borderRadius: 16 }}>
            <h2 style={{ fontSize: 20, fontWeight: 700, marginBottom: 8 }}
>Exported Symbols</h2>
            {data.exported_symbols?.length ? (
              <div style={{ overflowX: 'auto' }}>
                <table style={{ width: '100%', borderCollapse: 'collapse' }}>
                  <thead>
                    <tr>
                      <th style={{ textAlign: 'left', borderBottom: '1px 
solid #ddd', padding: 8 }}>Ordinal</th>
                      <th style={{ textAlign: 'left', borderBottom: '1px 
solid #ddd', padding: 8 }}>Address</th>
                    </tr>
                  </thead>
                  <tbody>
                    {data.exported_symbols.map((e, i) => (
                      <tr key={i}>
                        <td style={{ borderBottom: '1px solid #f2f2f2', 
padding: 8 }}>{e.ordinal}</td>
                        <td style={{ borderBottom: '1px solid #f2f2f2', 
padding: 8 }}>0x{(e.address || 0).toString(16)}</td>
                      </tr>
                    ))}
                  </tbody>
                </table>
              </div>
            ) : (
              <div>Export tablosu yok.</div>
            )}
          </section>
        </div>
      )}
    </div>
  )
}
```

Not: Tailwind bağımlılığı eklemedik; inline stiller kullandık ki sıfır konfigürasyonla çalışsın. İstersen Tailwind + shadcn ekleyebileceğin bir sürüm de hazırlayabilirim.

---

3) Çalıştırma Talimatları (Özet)

1. **Backend**

```
cd backend
go mod tidy
go run main.go
# → :8080 dinler
```

2. **Frontend (Vite)**

```
cd frontend
npm i
npm run dev
# → Vite varsayılan olarak http://localhost:5173
```

3. Tarayıcıda **Frontend**'i aç, bir **.exe** veya **.dll** seç, **Analiz Et**'e tıkla.

Backend CORS tüm origin'lere açıktır (örnek amaçlı). Üretimde `Access-Control-Allow-Origin`'i kendi domain'inle sınırlandır.

---

4) Geliştirme Fikirleri

- 🔎İmza (Authenticode)/sertifika bilgilerini gösterme (ayrı bir kütüphane gerekir)
- 🪠RVA → FOA adres dönüşümleri ve adres çözümleyici
- 🧰Section entropy analizi (basit packed/obfuscated sezimi için)
- 🗂️JSON çıktısını indir butonu
- 📖Küçük bir ikon/versiyon bilgisi okuyucu (`.rsrc` içi)