

BLM2537

2025-2026 Güz Dönemi



PROJE TANITIM

Hazırlayan:

Ömer Faruk Şahin

24290571

VIDEO: <https://www.youtube.com/watch?v=7llg1mB3QzM>

GITHUB: <https://github.com/ofsahof/terminal-blog>

PROJE ÖZETİ VE AMACI

Günümüz web dünyasında kişisel portfolyo siteleri genellikle statik ve tekdüze tasarımlara sahiptir. Bu proje, özellikle yazılım geliştiricilere hitap eden, etkileşimli ve nostaljik bir Komut Satırı Arayüzü (CLI) deneyimi sunarak bu standartlaşmaya bir alternatif getirmeyi amaçlamaktadır.

TerminalFolio, React kütüphanesi üzerine inşa edilmiş, tarayıcı tabanlı bir terminal emülatörüdür. Kullanıcılar, Linux benzeri komutlarla (ls, cd, cat) sanal bir dosya sistemi üzerinde gezinebilir, sistemin temasını dinamik olarak değiştirebilir ve sistem içerisine gömülü (embedded) harici web uygulamalarını çalıştırabilirler. Proje, modern web teknolojileri ile klasik arayüz tasarımını birleştirerek özgün bir kullanıcı deneyimi sunmaktadır.



Genel Tanıtım

KULLANILAN TEKNOLOJİLER:

- Frontend: React 18, Vite
- Dil: JavaScript (ES6+), JSX
- Stil: Vanilla CSS & CSS Değişkenleri
- Veri Yapısı: JSON tabanlı VFS & Stack Algoritması

KLASÖR YAPISI:

```
src/
├── App.jsx           # Görünüm Yöneticisi (View Controller / Shell)
├── hooks/
│   └── useTerminal.jsx # Çekirdek Mantık ve State Yönetimi
├── commands/
│   ├── index.js      # Komut Kayıt Defteri (Registry Pattern)
│   └── *.js/jsx      # Modüler Komutlar (ls, cd, help, theme...)
├── games/
│   └── index.js       # Oyun Entegrasyon Sistemi
└── utils/
    ├── filesystem.js  # Sanal Dosya Sistemi Verisi (JSON Tree)
    └── pathHelper.js  # Yol Çözümleme Algoritması (Stack Logic)

public/
├── content/          # Sanal Dosya Sistemi (VFS) için Statik Veriler
│   ├── posts/        # Blog Yazıları (.org formatında)
│   ├── projects/     # Proje Açıklamaları
│   └── welcome.txt    # Açılış Ekranı ASCII Art Mesajı
├── odev/             # Sisteme Gömülen Saf HTML/CSS Modülü (Legacy)
│   ├── index.html     # Ödev Ana Sayfası (Semantik HTML)
│   ├── style.css      # Ödev Stilleri (Grid & Flexbox)
│   └── script.js      # Ödev Mantığı (DOM Manipülasyonu)
└── project_screenshot.png # Dokümantasyon Görselleri
```

Sistem Mimarisi

Çekirdek Yapı:

- Sistemin beyni ve dış görünümü birbirinden tamamen izole edilmiştir.
- **Görünüm Kontrolcüsü (App.jsx):** Bu bileşen herhangi bir terminal mantığı barındırmaz. Sadece uygulamanın o anki görünüm durumunu (view: 'terminal' | 'game') yönetir. gameRegistry üzerinden hangi bileşenin (Terminal veya Oyun) render edileceğine karar verir.
- **Mantık Katmanı (useTerminal.jsx):** Sistemin beyni olarak çalışan bu "Custom Hook", tüm durum yönetimini üstlenir:
 - history: Ekranda gösterilen satırların listesi (JSX veya String).
 - command: Kullanıcının input alanındaki anlık girdisi.
 - path: Mevcut sanal dosya yolu (Örn: ~/projects).
 - commandHistory: Geçmiş komutların tutulduğu yığın (Stack).

Komut Yaşam Döngüsü:

- Sistemde her komut eşit değildir. useTerminal kancası içerisindeki handleCommand fonksiyonu, komutları çalışma prensiplerine göre iki ana kategoriye ayırır:
- **Durum Değiştirenler (State Mutators):** cd, clear, play gibi komutlardır. Bunlar ekrana basit bir metin basmak yerine uygulamanın durumunu değiştirir. Fonksiyondan özel bir sinyal objesi (Örn: { isPathUpdate: true }) döndürürler ve zincirleme reaksiyonu durdururlar.
- **İçerik Üretenler (Content Renderers):** ls, cat, help gibi komutlardır. Doğrudan JSX veya String döndürürler. Bu içerik, kullanıcının girdiği komutla birlikte history dizisine eklenerek ekrana basılır.

Modüler Eklenti Sistemi:

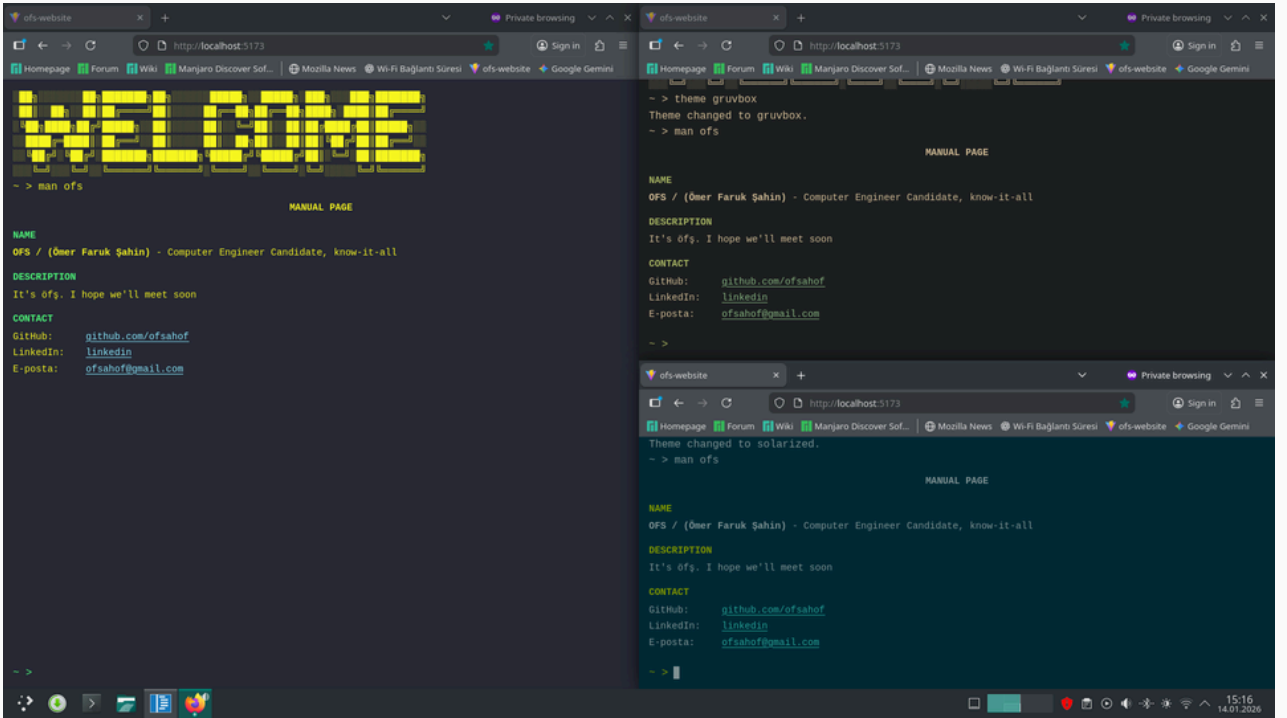
- Çekirdek yapı, çalıştırdığı komutlardan bağımsızdır
- **Komut Kayıt Defteri (commands/index.js):** Tüm komutlar (cat.jsx, ls.jsx vb.) burada toplanır ve tek bir obje olarak dışarı aktarılır. Sistem, ls komutu geldiğinde bu kayıt defterine bakarak ilgili fonksiyonu dinamik olarak çağırır. Bu sayede ana koda dokunmadan yeni komut eklemek mümkündür.
- **Oyun Kayıt Defteri (games/index.js):** Benzer şekilde, sisteme eklenecek mini oyunlar burada tanımlanır ve play komutu ile dinamik olarak çağırılır.

Sanal Dosya Sistemi (VFS):

- Dosya sistemi, veri ve mantık olarak ikiye ayrılmıştır:
- **Veri Katmanı (filesystem.js):** Sadece dosya ve klasörlerin hiyerarşisini tutan saf bir JSON ağaç yapısıdır.
- **Mantık Katmanı (pathHelper.js):** Dosya sisteminde gezinmek için gereken algoritmaları içerir. resolvePath fonksiyonu, .. (geri git) veya göreceli yolları çözümllemek için bir Yığın (Stack) algoritması kullanır.
- **İçerik Yönetimi (fetchHelper.js):** Performans optimizasyonu için geliştirilmiştir. Kullanıcı bir dosyayı okumak istediğinde (cat komutu ile), bu modül devreye girer ve dosya içeriğini asenkron olarak (fetch API ile) sunucudan veya public/content klasöründen dinamik olarak çeker. Bu "Lazy Loading" (Tembel Yükleme) mantığı, uygulamanın açılış hızını önemli ölçüde artırır.

Tema Motoru:

- Tema sistemi global CSS deęiřkenleri zerine kuruludur.
- **index.css** dosyasında her tema iin (Dracula, Gruvbox vb.) renk paletleri (**--bg**, **--fg**) tanımlanmıřtır.
- **theme komutu** alıřtırıldığında, **JavaScript DOM maniplasyonu ile <body> etiketinin sınıfını deęiřtirir**. Tm bileřenler CSS deęiřkenlerini kullandığı iin renk deęiřimi anlık ve tm sistemde eř zamanlı gerekleřir.



Oyunlar:

Sitede aktif olarak yılan oyunu ve istenen asıl ödevi görüntüleyen bir tarayıcı vardır.

