**SIM Inventory Management (React + Windsurf)**

Profesyonel bir **SIM Card Inventory Management** uygulamasını, Windsurf aracılığıyla sıfırdan (React + TypeScript) geliştirmek için uçtan uca plan ve hazır **prompt paketi**. Aşağıdaki akış, gerçek API (Swagger) ile entegre olup, erişim yoksa MSW ile **mock** modunda da çalışır.

**0) Kapsam & Demo Senaryoları (E2E)**

1. **Onboard SIM Batch (MVNE)**
   * Admin CSV ile fiziksel SIM ICCID veya eSIM profillerini içe aktarır.
   * Sistem benzersizlik/format validasyonlarını yapar, **SIM Resource** olarak kaydeder (status = Available).
2. **Search SIM Resources**
   * Admin; ICCID, IMSI, status, type (eSIM/Physical), batchId, tarih aralıklarına göre arar.
   * Sunucu tarafı pagination/sort/filters.
3. **Create SIM Order (MVNO)**
   * MVNO belirli SIM’leri tahsis etmek için **Order** oluşturur.
   * Order; SIM Resource referanslarını ve hedef lifecycle aksiyonunu içerir.
4. **Update SIM Lifecycle (MVNO)**
   * Activate, Suspend, Terminate, Release, Retire aksiyonları.
   * SIM status güncellenir, Order detail’de yansır.
5. **View Order Status**
   * Admin, Order detayında bağlı SIM kaynaklarının son durumlarını görür (timeline/aktivite log).
6. **Retire SIM Resource (MVNE)**
   * SIM’i Retired yapma veya Available ise **Delete** (guard: sadece Available iken silinebilir).

**Roller**: MVNE\_Admin, MVNO\_User (RBAC ile yetki kısıtları)

**1) Domain Model (TMF Resource Inventory esinli)**

**SIM Resource**

* id: string (UUID)
* iccid: string (unique, required)
* imsi?: string
* type: "Physical" | "eSIM"
* status: "Available" | "Allocated" | "Active" | "Suspended" | "Terminated" | "Retired"
* batchId?: string
* profileType?: "prepaid" | "postpaid" | string
* createdDate: string (ISO)
* lastModifiedDate: string (ISO)

**Order**

* id: string
* requestedAction: "Allocate" | "Activate" | "Suspend" | "Terminate" | "Release" | "Retire"
* resourceIds: string[] (SIM id listesi)
* status: "Pending" | "InProgress" | "Completed" | "Failed"
* createdDate, lastModifiedDate
* events: { at: string; status: string; message?: string }[]

**State Geçişleri (özet)**

* Available -> Allocated -> Active
* Active -> Suspended -> Active | Terminated
* Allocated -> Release -> Available
* Any -> Retired (terminal)
* **Delete**: sadece Available iken.

**Validasyon Kuralları**

* iccid benzersiz + zorunlu, type sadece Physical|eSIM, create’da status=Available default.

**2) Teknoloji & Mimarî**

* **Build**: Create React App (react-scripts) + React 18 + TypeScript
* **UI:** Ant Design v5, @ant-design/icons, kompakt toolbar ve Popover kullanımları
* **Veri yönetimi:** react-query v3 (QueryClient proje genelinde), axios
* **Form/Doğrulama:** antd Form; ek olarak react-hook-form ve yup kullanılabilir durumdadır
* **Kimlik Doğrulama:** Keycloak (`src/contexts/KeycloakContext.tsx`), token otomatik yenileme ve `apiService.setAuthToken`
* **Stil/Temalar**: Ant Design tema değişkenleri (CSS variables) üzerinden özelleştirme; tema paneli için `ThemeContext.tsx`
* **Routing**: React Router
* **Mock**: MSW (Service Worker) — Swagger erişim yoksa
* **Test**: Vitest + Testing Library (unit), Playwright (e2e – opsiyonel)
* **Kalite**: ESLint + Prettier + Husky (pre-commit)
* **CI/CD**: GitHub Actions; Dockerfile + multi-stage
* **Env**: .env içinde VITE\_API\_BASE\_URL, VITE\_USE\_MOCK

**Dizin Yapısı (öneri)**

src/

  App.tsx

  index.tsx

  i18n.ts

  routes/

    AppRoutes.tsx

  contexts/

    KeycloakContext.tsx

    ThemeContext.tsx

  components/

    layout/ (MainLayout, navbar/sidebar)

    common/ (LoadingSpinner, vb.)

  pages/

    dashboard/ (Dashboard.tsx)

    sim-resources/ (list, detail, create, batch import)

    sim-orders/ (list, detail, create)

    reports/ (Reports.tsx)

    auth/, user/

  services/

    api.service.ts

  types/

    sim.types.ts

  styles/ (global stiller)

  utils/

**3) API Entegrasyon Stratejisi**

**Swagger**: <https://dri-api.dnext-pia.com/api/resourceInventoryManagement/v4/swagger-ui/index.html>

Base URLs:

- Resource Inventory (TMF639): <https://dri-api.dnext-pia.com/api/resourceInventoryManagement/v4>

- Resource Ordering (TMF652): <https://dro-api.dnext-pia.com/api/resourceOrderingManagement/v4>

- Party Management (TMF632): https://dri-api.dnext-pia.com/api/partyManagement/v4

1. **- Kimlik doğrulama:** Keycloak token, axios interceptor ile `Authorization: Bearer <token>`
2. **Axios Instance & Interceptor**
   * Authorization bearer token ya da mTLS vs. gerekiyorsa buradan enjekte edilir.
   * Hata kodlarını (400/401/403/404/409/500) kullanıcı-dostu mesajlara map’leyin.
3. **Mock Modu (MSW)**
   * .env ile VITE\_USE\_MOCK=true ise MSW açılır.
   * Handlers: /sim-resources, /orders, /batches uçları için CRUD + lifecycle.

**4) UI Bilgi Mimarisi & Ekranlar**

**Rotalar**

* / Dashboard (istatistikler, hızlı aksiyonlar)
* /sims Liste & filtre & bulk işlemler
* /sims/:id Detay (status timeline, order ilişkileri)
* /orders Liste & oluşturma
* /orders/:id Detay (linked SIMs + event log)
* /batches Liste & onboarding geçmişi
* /onboarding Batch Upload Wizard (CSV)
* /admin Ayarlar, profil, RBAC görünümü

**Temel Bileşenler**

* SimTable, SimFilters, StatusBadge, LifecycleMenu
* BatchUploadDialog (CSV şablon indirme + örnek validasyon)
* OrderForm (SIM seçimi + hedef aksiyon)
* ConfirmDialog, Toast (optimistic updates + undo)

**UX Notları**

* Büyük listeler için **server-side pagination**.
* Filtreler URL query string’e yazılsın (paylaşılabilir link).
* Hata/empty/loading durum kartları.
* Erişilebilirlik (A11y), klavye navigasyonu.

**5) Kurulum Komutları (öneri)**

# Proje iskeleti

npm create vite@latest sim-inventory -- --template react-ts

cd sim-inventory

# Kitler

npm i axios @tanstack/react-query react-router-dom \

zod react-hook-form \

class-variance-authority clsx \

@radix-ui/react-avatar @radix-ui/react-dialog @radix-ui/react-dropdown-menu \

lucide-react

# Tailwind

npm i -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

# shadcn/ui

npm i -D shadcn-ui@latest

npx shadcn-ui@latest init

# component eklerken: npx shadcn-ui@latest add button input table dialog toast badge

# OpenAPI codegen (opsiyonel)

npm i -D openapi-typescript-codegen

# MSW + Testler

npm i -D msw vitest @testing-library/react @testing-library/jest-dom @types/jest \

playwright @playwright/test eslint prettier husky lint-staged

# Git hook

npx husky init

**tailwind.config.cjs**: content: ["./index.html","./src/\*\*/\*.{ts,tsx}"]

**.env** (ör.)

VITE\_API\_BASE\_URL=https://dri-api.dnext-pia.com/api/resourceInventoryManagement/v4

VITE\_USE\_MOCK=false

**6) React Query – Query Key & Cache Stratejisi**

* ['sims', {page, q, status, type, batchId }] - `['simResources', searchCriteria, currentPage, pageSize]` – liste
* ` ['simResource', id] ` – detay
* ` ['simStatistics', token] ` – istatistikler
* ` ['recentSims', token] `, `['recentOrders', token]` – dashboard listeleri (default sıralama uygulanır)
* ` ['simOrders', token, searchParams, currentPage, pageSize]` – sipariş listesi

**Invalidation:**

* - Lifecycle/Allocation/Import sonrası: `simResource`, `simResources`, `simStatistics`
* - Sipariş işlemleri sonrası: `simOrders`

**7) CSV Şablonu (Batch Onboarding)**

Kolonlar: iccid,imsi,type,profileType,batchId

Örnek:

iccid,imsi,type,profileType,batchId

8937201001234567890,286011234567890,Physical,prepaid,2025-09-01-A

8937201001234567891,,eSIM,postpaid,2025-09-01-A

Validasyonlar:

* iccid zorunlu + regex/length kontrolü (genelde 19–20 hane Luhn varyantı)
* type ∈ {Physical,eSIM}
* imsi opsiyonel; varsa numeric/length

**8) Hata Haritalama (Örnek)**

* **400**: Validasyon – kullanıcıya form alanı bazlı göster.
* **401/403**: Oturum/Yetki uyarısı, tekrar giriş.
* **404**: Kayıt bulunamadı.
* **409**: Çakışma (duplicate ICCID) – onboarding sırasında satır işaretleme.
* **500**: Beklenmeyen – retry öner.

**9) Güvenlik & RBAC**

* Context’te role sakla (MVNE\_Admin vs.).
* UI, role’a göre aksiyonları gizler/disable eder (örn. Delete sadece Admin ve status=Available iken).
* Token yenileme (silent refresh) interceptor’da.

**10) Prompt Paketi (Windsurf)**

Aşağıdaki promptları sırasıyla verin. Her birinde **dosya yollarını**, **oluşturulacak/ değiştirilecek dosyaları** açık yazdırmasını isteyin. Kabul kriterlerini her promptun sonuna ekleyin.

**Prompt 01 — Proje İskeleti & Temel Kurulum**

Windsurf, TypeScript ile Vite tabanlı bir React projesi kur. Proje adı: sim-inventory.

Tailwind + shadcn/ui + React Router + React Query + React Hook Form + Zod kur ve yapılandır.

Aşağıdaki dosya ağacını oluştur ve içeriklerini doldur:

- src/app/providers/QueryProvider.tsx (QueryClient, devtools opsiyon)

- src/app/providers/ThemeProvider.tsx

- src/lib/axios.ts (axios instance, baseURL=import.meta.env.VITE\_API\_BASE\_URL)

- src/styles/globals.css (Tailwind base/components/utilities)

- src/app/router/index.tsx (temel layout, routes stub)

- src/components/ui/\* (shadcn eklenenler)

- src/pages/Home.tsx (dummy)

- src/main.tsx (Providers sarımı)

Kabul: `npm run dev` ile proje açılır, ana sayfa ve temel layout görülür.

**Prompt 02 — UI Kiti & Tema**

Minimal, kurumsal bir tema uygula. Navbar, Sidebar, Content layout bileşenleri oluştur.

StatusBadge, EmptyState, ErrorState, LoadingSpinner komponentlerini ekle.

Kabul: / sayfasında örnek status badgeleri ve boş içerik kartları görünsün.

**Prompt 03 — OpenAPI Client Entegrasyonu**

Swagger UI’den OpenAPI JSON kaynağını kullanarak openapi-typescript-codegen ile client üret.

Çıktıyı `src/lib/api` altına yerleştir. `src/lib/apiClient.ts` içinde client’i tek noktadan ihraç et.

`src/lib/axios.ts` interceptor: Authorization header (localStorage’dan token), hata mapping.

Kabul: `src/lib/api` içinde modeller ve endpoint fonksiyonları oluşsun; example bir `health` çağrısı göster.

**Prompt 04 — SIM Listesi (Search + Pagination)**

/sims rotasını oluştur. SimTable, SimFilters, useSimQueries hook’ları yaz.

Filtreler: iccid, imsi, status, type, batchId. Server-side pagination (page, size, sort).

Query key: ['sims', {page,q,status,type,batchId}]. URL query sync.

Kabul: Mock API ile 100+ kayıt listelenir; filtre/sayfalama çalışır; boş/hata durumları gösterilir.

**Prompt 05 — SIM Detay & Lifecycle Aksiyonları**

/sims/:id rota. SimDetail sayfasında temel bilgiler, status timeline, ilişkili orders.

LifecycleMenu: Activate, Suspend, Terminate, Release, Retire butonları.

Optimistic update + rollback. Başarılı işlemde toast.

Kabul: Status değişimleri UI’da anında yansır; hatada eski duruma döner.

**Prompt 06 — Batch Onboarding Wizard (CSV Upload)**

/onboarding sayfasında 3 adımlı wizard:

1) CSV yükle (şablon indir). 2) Önizleme + satır bazlı validasyon. 3) Sunucuya gönder.

409 (duplicate ICCID) dönen satırları kırmızı işaretle, tekrar deneme imkânı ver.

Kabul: Örnek CSV ile en az 2 başarılı, 1 hatalı satır senaryosu gösterilir.

**Prompt 07 — Order Oluşturma (MVNO)**

/orders sayfası ve OrderForm: SIM çoklu seçimi (Listeden seç veya ICCID ile ekle), requestedAction seç.

Oluşturulan order'ı /orders/:id sayfasına yönlendir. Order ile SIM ilişkisini göster.

Kabul: Pending -> InProgress -> Completed/Failed durum akışı simüle edilsin (mock event stream).

**Prompt 08 — Retire & Delete Guard’ları**

/sims/:id üzerinde Retire aksiyonu ekle. Delete butonu sadece status=Available ve role=MVNE\_Admin ise aktif.

Silmede confirm dialog + `are you sure` metni. Hata 409/400 durumlarını yakala.

Kabul: RBAC ve guard kuralları UI’da doğrulanır.

**Prompt 09 — Order Detayı & Event Timeline**

/orders/:id sayfasında event timeline, bağlı SIM listesi, durum rozetleri.

`refetchInterval` ile 5 sn’de bir güncelle (mockta event ilerlesin).

Kabul: Completed olduğunda success alert, Failed’de retry butonu görünür.

**Prompt 10 — Auth Mock & RBAC**

Basit login ekranı: kullanıcı adıyla role seçimi (MVNE\_Admin veya MVNO\_User).

Role, Context’te saklansın; aksiyonlar role’a göre gizlensin.

Kabul: MVNO\_User Delete’i göremez; MVNE\_Admin görür.

**Prompt 11 — Testler & MSW**

Unit: SimFilters validasyonları (Zod), hooks’ların query key’leri.

Component: SimTable render + filtre etkileşimi.

E2E: Playwright ile /sims listeleme, filtre, detay, lifecycle.

Kabul: `npm test` başarıyla koşar, en az 1 e2e senaryosu geçer.

**Prompt 12 — CI/CD + Docker**

Dockerfile (multi-stage) ve GitHub Actions workflow (build+test) ekle.

Preview deploy için Vite preview kullan.

Kabul: `docker build .` başarılı; CI pipeline testleri çalıştırır.

**11) Kabul Kriterleri (Özet)**

* **Performans**: Liste sayfası 10k kayıt mock’ta akıcı scroll (sanallaştırma opsiyonel: @tanstack/react-virtual).
* **Erişilebilirlik**: Form alanları label’lı; kontrast uygun; klavye ile erişilebilir.
* **Hata Yönetimi**: API hataları kullanıcı anlayacağı dille; tekrar dene imkanı.
* **Güvenlik**: Yetkisiz aksiyonlar UI’da mümkün değil; token var ise header’a eklenir.

**12) Riskler & Alternatifler**

* Swagger erişimi kısıtlı olabilir → **MSW** ile offline geliştirme.
* OpenAPI şeması tam değil → zod ile UI-level ek validasyon.
* Lifecycle akışları farklı iş kuralları gerektirebilir → Hook’larda guard fonksiyonları.

**13) Sonraki Adımlar**

1. Prompt 01’i çalıştır, repo’yu oluştur.
2. Prompt 03 ile OpenAPI client’i üret (Swagger JSON URL hazırla).
3. Prompt 04–06 ile temel işlevleri tamamla; 07–09 ile sipariş/lifecycle akışını bitir.
4. 10–12 ile test/CI ve dağıtım.

Hazır olduğunda Prompt 01 ile başlayalım. JSON linkini alabilirsen Prompt 03’te doğrudan codegen’e geçeriz; alamazsan, MSW ile mock akıştan ilerleriz.

**Ekstra Komutlar**

**1) Projemde profesyonel bir i18n yapısı kurmak istiyorum.**

Şartlar:

1. react-i18next kütüphanesi kullanılacak.

2. src/locales/tr.json ana dil dosyası olacak, diğer dillere OpenAI API üzerinden otomatik çevrilecek.

3. translate.js dosyası: - en.json’u okuyacak - tr.json, de.json, fr.json gibi dillere çevirecek kullanıcı anlık seçebilecek - OpenAI resmi SDK (npm openai) ile çalışacak - .env dosyasındaki OPENAI\_API\_KEY değerini kullanacak

4. i18n.js yapılandırması olacak (React tarafında çoklu dil desteği için).

5. GitHub Actions pipeline (translate.yml): - src/locales/tr.json dosyası değiştiğinde tetiklenecek - translate.js çalışacak - oluşan çeviri dosyaları commit edilecek

6. Bana şu dosyaları tam kodlarıyla üret: - translate.js - i18n.js - örnek src/locales/tr.json - örnek src/locales/en.json - .github/workflows/translate.yml

**2)Projemde bir "Tema Ayarları" butonu istiyorum.**

Bu butona tıklandığında sol tarafta sidebar açılsın içinde kullanıcı uygulamanın temasını özelleştirebilmeli.

İstediğim özelleştirmeler:

- Navbar arka plan rengi

- Sidebar arka plan rengi

- Buton rengi

- Sayfa arka plan rengi

- Yazı rengi

Gereksinimler:

- Her seçenek için color picker (veya hazır renk dropdown) olsun

- Seçilen renkler localStorage’a kaydedilsin (sayfa yenilense bile kalsın)

- Renkler CSS değişkenleri (CSS variables) ile dinamik olarak uygulansın

- Varsayılana döndür butonu olsun - React functional components ve hooks ile yazılsın

- UI sade ve kullanışlı olsun