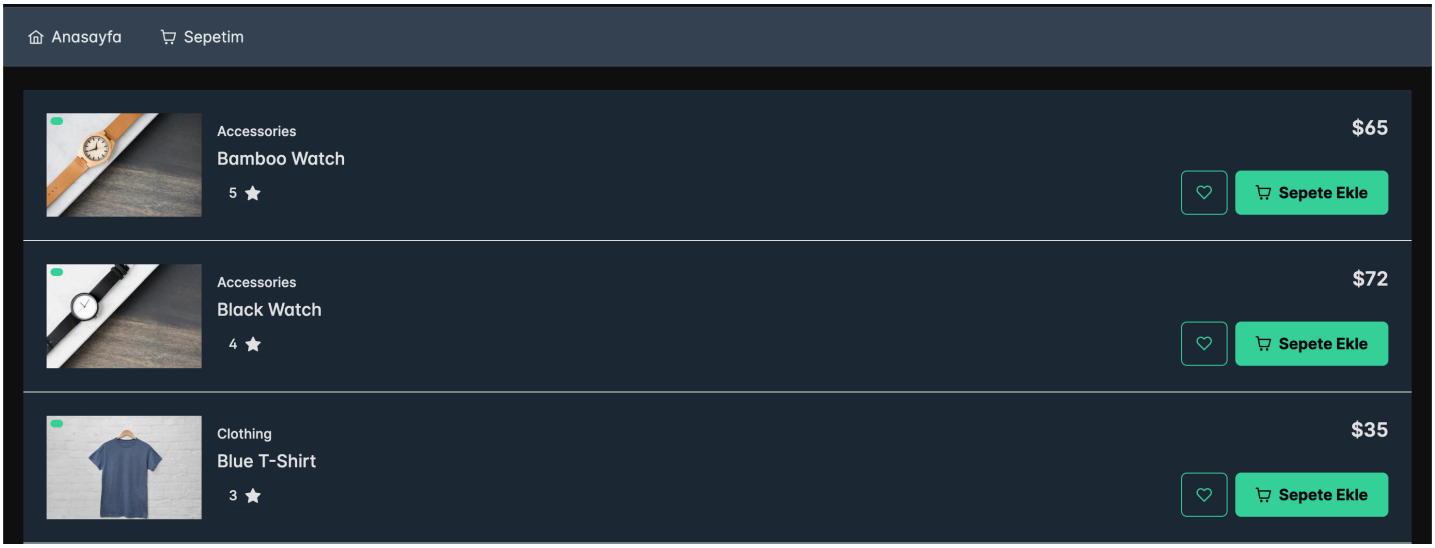
Proje 4 / E-Ticaret Sitesi



Genel Bilgi

Bu projenin amacı, kullanıcıların çevrim içi olarak ürünleri görüntüleyebileceği, sepete ekleyebileceği ve temel e-ticaret işlevlerini gerçekleştirebileceği bir web uygulaması geliştirmektir. Proje, hem frontend (kullanıcı arayüzü) hem de backend (sunucu tarafı) bileşenlerinden oluşmaktadır. Database için AWS, Sunucuyu çalıştırmak için Google Cloud kullanılmıştır.

```
<script>
import DataView from "primevue/dataview";
import Tag from "primevue/tag";
import Button from "primevue/button";
import Menubar from "primevue/menubar";
import Dialog from "primevue/dialog";
import Badge from "primevue/badge";
```

Frontend için Vue.js ve Prime Vue kullandım. Menüler, butonlar ve genel görüntü için Prime Vue kullandım. Bunun için gerekli bileşenleri import ederek genel bir e-ticaret sitesi görünümü yakalamaya çalıştım.

API Routing Yapısı

Backend tarafında Express.js framework'ü kullanılarak modüler bir yönlendirme (routing) yapısı kurulmuştur. Ana uygulama dosyasında, ürünlerle ilgili API endpoint'ler routes/products.js dosyasında ayrı olarak tanımlanmış ve aşağıdaki şekilde uygulamaya entegre edilmiştir:

```
const productRoutes = require("./routes/products");
app.use("/api/products", productRoutes);
```

Bu yapı sayesinde, /api/products adresine gelen tüm HTTP istekleri products.js dosyasına yönlendirilmekte, böylece proje daha düzenli ve ölçeklenebilir bir hale getirilmektedir. Bu yöntem, farklı kaynaklar (örneğin kullanıcılar, siparişler vb.) için benzer şekilde ayrı route dosyaları oluşturulmasına da olanak tanımaktadır

Ürün Verilerinin Frontend'e Aktarılması

Projenin frontend kısmı Vue.js framework'ü ile geliştirilmiştir. Ürün verileri, backend tarafından sağlanan bir REST API üzerinden çekilmektedir. Vue bileşeni yüklendiğinde, fetch() yöntemi ile Google Cloud üzerinde çalışan Node.js backend'e HTTP GET isteği gönderilmekte ve gelen JSON formatındaki veriler Vue bileşenine aktarılmaktadır.
Aşağıda kullanılan kod parçası gösterilmektedir:

```
mounted() {
  fetch("http://34.32.89.235:80/api/products")
    .then((res) => res.json())
    .then((data) => {
      this.products = data;
    })
    .catch((err) => console.error("Ürünleri çekerken hata:", err));
},
```

Bu örnekte, Google Cloud üzerinde çalışan backend sunucusu varsayılan olarak 3000 portunda hizmet vermektedir. Ancak dış dünyadan erişim için Google Cloud Load Balancer kullanılmış ve gelen istekler 80 portu üzerinden alınarak backend sunucunun 3000 portuna yönlendirilmiştir. Böylece kullanıcılar tarayıcılarında doğrudan http://34.32.89.235/api/products adresi üzerinden ürün verilerine erişebilmektedir.

Veritabanı Yapısı ve AWS RDS PostgreSQL Entegrasyonu

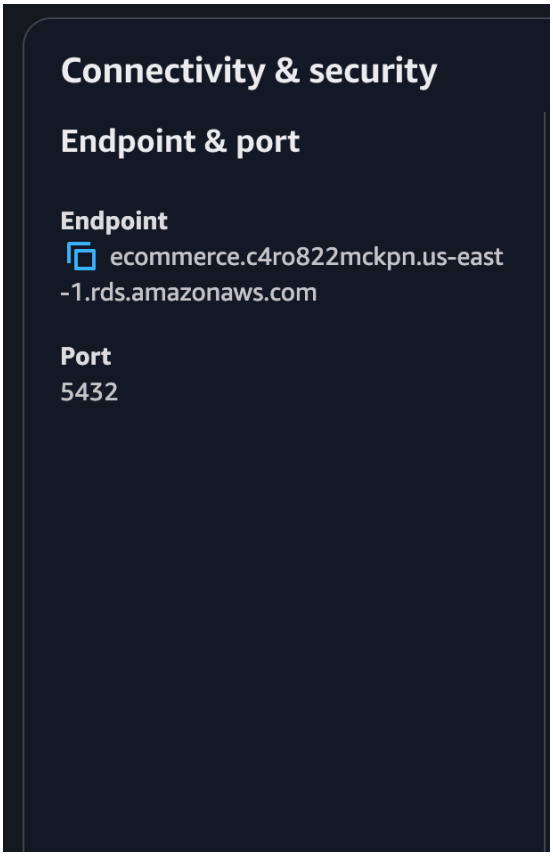
E-ticaret uygulamasının veritabanı altyapısı **Amazon Web Services (AWS)** üzerinde barındırılmıştır. Veritabanı yönetimi için **Amazon RDS (Relational Database Service)** servisi kullanılmış ve veritabanı motoru olarak doğrudan **PostgreSQL** tercih edilmiştir.
Veritabanına ürün verilerinin yüklenmesi için aşağıdaki gibi bir seed.js dosyası hazırlanmıştır:

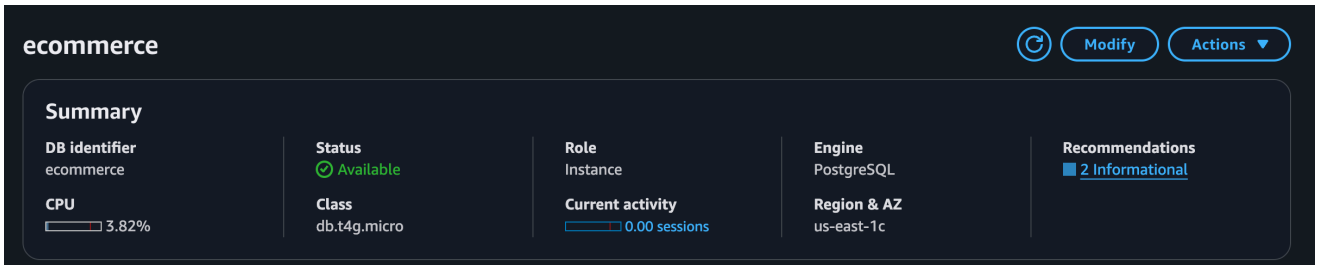
```
const { Client } = require("pg");
require("dotenv").config();

const client = new Client({
  host: process.env.DB_HOST,
  port: process.env.DB_PORT,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_DATABASE,
  ssl: {
    rejectUnauthorized: false,
  },
});
```

Script içerisinde dotenv kütüphanesi kullanılarak hassas bilgiler .env dosyasından okunmuştur:

```
DB_HOST=ecommerce.c4ro822mckpn.us-east-1.rds.amazonaws.com
DB_PORT=5432
DB_USER=postgres
DB_PASSWORD=****
DB_DATABASE=postgres
```





Yukarıda AWS arayüzü üzerindeki görseller yer almaktadır

Ürün Verilerinin Veritabanına Eklenmesi

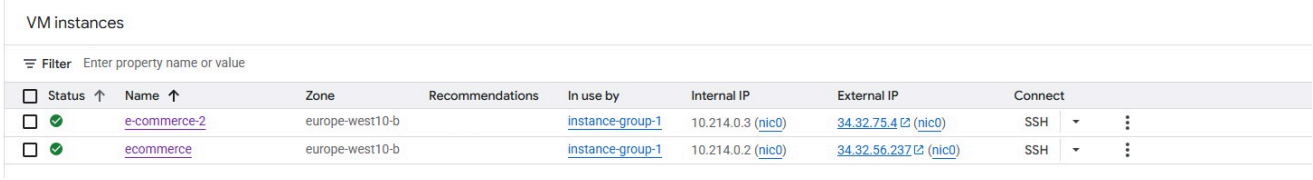
PostgreSQL veritabanına bağlanılır ve products adlı tablo oluşturulur. Eğer tablo zaten varsa, yeniden oluşturmadan devam eder. Ardından örnek ürün verileri tabloya eklenir:

```
await client.query(`
  CREATE TABLE IF NOT EXISTS products (
    id SERIAL PRIMARY KEY,
    name TEXT,
    category TEXT,
    image TEXT,
    price REAL,
    inventoryStatus TEXT,
    rating INTEGER
  )
`);
```

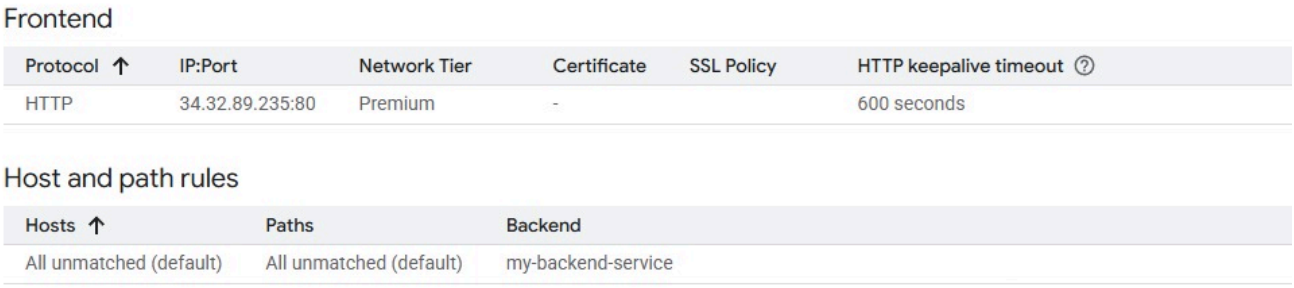
Google Cloud Platform (GCP) Üzerinde Backend Sunucusunun Barındırılması

E-ticaret uygulamasının backend kısmı, Google Cloud Platform (GCP) üzerinde barındırılmıştır. Bu kapsamda GCP üzerinde Ubuntu işletim sistemine sahip bir sanal makine (VM Instance) oluşturulmuş ve uygulama bu makine üzerinde çalıştırılmıştır. SSH bağlantısı üzerinden VM'e erişilerek Node.js, npm ve git gibi gerekli araçlar kurulmuş; ardından backend projesi GitHub üzerinden klonlanarak bağımlılıkları yüklenmiştir. Sunucu, 3000 portu üzerinden hizmet verecek şekilde çalıştırılmıştır.

Uygulamanın internet üzerinden erişilebilir hale gelmesi için Google Cloud Load Balancer yapılandırılmıştır. Load Balancer aracılığıyla gelen HTTP (80 portu) istekleri, arka plandaki backend sunucunun 3000 portuna yönlendirilmiştir. Bu yapı sayesinde, frontend uygulaması backend'e doğrudan IP ve port belirtmeden, Load Balancer üzerinden erişim sağlayabilmektedir. Ayrıca, Load Balancer kullanımı; yüksek erişilebilirlik, daha iyi trafik yönetimi ve ölçeklenebilirlik gibi avantajlar da sunmaktadır. VM'ye ait güvenlik duvarı kuralları da uygun şekilde yapılandırılarak, Load Balancer'ın trafik yönlendirebilmesi sağlanmıştır.




```
scengiz2002@ecommerce:~$ ls
project4-e-commerce-app
scengiz2002@ecommerce:~$ cd project4-e-commerce-app/backend/
scengiz2002@ecommerce:~/project4-e-commerce-app/backend$ node index.js
Backend http://localhost:3000 adresinde çalışıyor
```



Load Balancer sayfasından ekran görüntüsü