Web API'yi oluştururken kullanılan temel teknolojilerden olan **Node.js** ve **Express.js**'in ne olduğunu ve API'leri test etmek için yaygın olarak kullanılan **Postman** gibi bir aracın ne anlama geldiğini aşağıda detaylıca açıklıyorum.

**1. Node.js Nedir?**

**Node.js**, JavaScript'in sunucu tarafında çalıştırılabilmesini sağlayan bir platformdur. Normalde JavaScript, yalnızca tarayıcı üzerinde çalışan bir dil olarak bilinir. Ancak Node.js sayesinde, JavaScript ile tarayıcı dışında, sunucularda da uygulamalar geliştirmek mümkündür.

* **Temel Özellikleri:**
  + **Tek işlemli (single-threaded)** ve **olay tabanlı (event-driven)** bir mimaride çalışır. Bu, Node.js'in yüksek performansla birçok isteği aynı anda işleyebilmesini sağlar.
  + **Non-blocking (engelleyici olmayan) I/O** yapısıyla çok sayıda isteği aynı anda işlemede oldukça etkilidir. Özellikle çok sayıda veri tabanı isteği veya dosya işlemleri gibi görevlerde öne çıkar.
  + **V8 Motoru** kullanır. Bu motor, Google Chrome'un JavaScript motoru olup, JavaScript kodunu çok hızlı bir şekilde çalıştırır.
* **Ne İşe Yarar?** Node.js ile web sunucuları, RESTful API'ler, gerçek zamanlı uygulamalar (chat uygulamaları gibi) ve daha birçok sunucu taraflı uygulama geliştirilebilir. Özellikle, API'lerin geliştirilmesi için oldukça popülerdir çünkü sunucuya gelen HTTP isteklerini (GET, POST vb.) kolayca yönetebilir.

**2. Express.js Nedir?**

**Express.js**, Node.js üzerine inşa edilmiş minimal ve esnek bir web uygulama framework'üdür. Express, Node.js'in temel işlevlerini geliştirerek, daha kolay ve hızlı bir şekilde web uygulamaları ve API'ler oluşturmayı sağlar.

* **Temel Özellikleri:**
  + Express, **basit ve minimal bir yapı** sağlar. Ancak esnekliği sayesinde büyük ölçekli projelerde de kullanılabilir.
  + Web uygulamaları ve API'ler için gerekli olan **HTTP yönlendirmesi (routing)**, **middleware yönetimi** ve **istek/yanıt işleme** gibi işlemleri daha kolay hale getirir.
  + Express ile bir sunucu oluşturabilir, çeşitli HTTP isteklerini (GET, POST, PUT, DELETE vb.) işleyebilir ve bu istekler sonucunda cevaplar döndürebilirsiniz.
* **Ne İşe Yarar?** Express.js, web API'ler ve web uygulamaları geliştirmek için Node.js üzerine kurulmuş bir araçtır. Örneğin, bir kullanıcının istekte bulunduğu bir URL'yi işleyip ona uygun bir yanıt döndürmek için Express.js kullanılabilir. Aynı zamanda, farklı veri formatlarını işlemek (JSON, form verileri vb.), veri tabanı entegrasyonu sağlamak ve API geliştirme süreçlerini hızlandırmak için idealdir.

**3. Web API Nedir?**

**Web API (Application Programming Interface)**, iki farklı uygulamanın veya sistemin birbirleriyle veri alışverişinde bulunmasını sağlayan bir arabirimdir. Örneğin, bir mobil uygulama, merkezi bir sunucudan kullanıcı bilgilerini almak istediğinde Web API üzerinden istek gönderir.

* Web API genellikle **RESTful API** olarak adlandırılır ve HTTP protokolü üzerinden çalışır.
  + **GET**: Veri almak için kullanılır.
  + **POST**: Yeni veri oluşturmak veya veri göndermek için kullanılır.
  + **PUT**: Mevcut veriyi güncellemek için kullanılır.
  + **DELETE**: Veri silmek için kullanılır.

Web API'ler, sunucular arasında veri transferi için standart bir yol sunar ve genellikle JSON formatında veri dönerler.

**4. Postman Nedir?**

**Postman**, API'leri geliştirmek, test etmek ve belgelemek için kullanılan popüler bir araçtır. Geliştiriciler, API'lerine HTTP istekleri (GET, POST, PUT, DELETE vb.) göndererek API'lerinin nasıl çalıştığını test edebilirler.

* **Ne İşe Yarar?**
  + Postman, bir API'ye veri göndermeden önce isteklerin doğru çalışıp çalışmadığını kontrol etmenizi sağlar.
  + Herhangi bir tarayıcı veya uygulama yazmanıza gerek kalmadan, doğrudan Postman üzerinden API'nize istek gönderip yanıtları inceleyebilirsiniz.
  + İsteklerinize çeşitli parametreler, başlıklar (headers) ve body ekleyerek farklı test senaryoları oluşturabilirsiniz.
  + API'nizin döndürdüğü yanıtları (örneğin JSON veri) Postman üzerinde detaylı bir şekilde görebilir ve analiz edebilirsiniz.

**Adım Adım Basit Bir Web API Projesi Yapmak**

**Adım 1: Node.js ve NPM (Node Package Manager) Kurulumu**

* İlk olarak, Node.js'in [resmi web sitesinden](https://nodejs.org/) Node.js’i indirin ve bilgisayarınıza kurun. Kurulum sırasında Node.js ile birlikte NPM (Node Package Manager) de yüklenecektir.
* Kurulduktan sonra, komut satırında şu komutlarla versiyonlarını kontrol edebilirsiniz:



**Adım 2: Proje Klasörü Oluşturma**

* Bir proje klasörü oluşturun ve terminalden bu klasör içine gidin:



Proje için package.json dosyasını oluşturmak için:



Bu dosya, projenizin bağımlılıklarını ve diğer bilgilerini tutar.

**Adım 3: Express.js'i Kurma**

* Express.js'i projenize bağımlılık olarak ekleyin:



**Adım 4: Basit Bir API Sunucusu Oluşturma**

Proje klasörünüzde bir index.js dosyası oluşturun ve aşağıdaki kodu ekleyin.

Tabii! Kodunu satır satır açıklayarak ne anlama geldiklerini basitçe anlatayım:

// express modülünü dahil ediyoruz

const express = require('express');

* Bu satırda **Express.js** adlı bir modülü projeye dahil ediyoruz. Express.js, web uygulamaları ve API'ler oluşturmak için kullanılan bir Node.js framework'üdür.

const app = express();

* express() fonksiyonunu çağırarak bir **Express uygulaması** oluşturuyoruz. Bu app değişkeni, sunucumuzla ilgili işlemleri yapmak için kullanacağımız ana nesne olacak.

const PORT = 3000;

* Bu satırda, sunucunun hangi port üzerinde çalışacağını belirliyoruz. Port, bilgisayarınızda bir uygulamanın çalıştığı adres gibi düşünülebilir. Burada port numarasını 3000 olarak belirledik.

// JSON verilerini işlemesi için Express'e built-in middleware ekliyoruz

app.use(express.json());

* Bu satır, sunucuya gelen isteklerde **JSON verilerini** anlayabilmesi için gerekli olan **middleware** (ara katman yazılımı) ekliyor. Bu sayede, sunucu gelen JSON verilerini (örneğin POST isteklerinde) otomatik olarak işleyebilir.

// Basit bir GET endpointi (route) oluşturuyoruz

app.get('/', (req, res) => {

res.send('Merhaba, bu basit bir Web API!');

});

* Bu satır, bir **GET isteği** için bir endpoint (route) oluşturuyor. app.get fonksiyonu, birisi sunucuya '/' yolundan (yani ana sayfa gibi düşünebilirsiniz) bir istek yaptığında çalışır.
  + req: Gelen isteği (request) temsil eder.
  + res: Geri döneceğimiz yanıtı (response) temsil eder.
  + res.send('Merhaba, bu basit bir Web API!'): İstek yapıldığında kullanıcıya bu mesajı döndürür.

// Bir dizi (array) kullanarak basit bir veri modeli oluşturuyoruz

let users = [

{ id: 1, name: 'Ali' },

{ id: 2, name: 'Ayşe' },

{ id: 3, name: 'Mehmet' }

];

* Bu satırda, örnek kullanıcıları temsil eden bir **dizi (array)** oluşturuyoruz. Bu dizi, id ve name alanlarına sahip üç kullanıcının bilgilerini içeriyor. Basit bir veri modeli olarak düşünebilirsiniz.

// Kullanıcıları listelemek için GET endpointi

app.get('/api/users', (req, res) => {

res.json(users);

});

* Bu satır, /api/users yoluna yapılan bir **GET isteğini** işleyen bir endpoint oluşturuyor.
  + Sunucuya /api/users yolundan bir istek yapıldığında, users dizisindeki tüm kullanıcıları **JSON formatında** döndürüyoruz (res.json(users)).

// Yeni bir kullanıcı eklemek için POST endpointi

app.post('/api/users', (req, res) => {

const newUser = {

id: users.length + 1,

name: req.body.name

};

users.push(newUser);

res.status(201).json(newUser);

});

* Bu satır, /api/users yoluna yapılan **POST isteklerini** işlemek için kullanılıyor.
  + req.body: Gelen POST isteğinin **gövdesini** (body) ifade eder. Burada yeni bir kullanıcı oluşturmak için name bilgisi JSON formatında gönderilmelidir.
  + const newUser = { id: users.length + 1, name: req.body.name };: Bu satır, yeni bir kullanıcı oluşturur. id, mevcut kullanıcı sayısına göre bir artışla belirlenir, name ise isteğin body kısmından alınır.
  + users.push(newUser);: Yeni kullanıcıyı users dizisine ekliyoruz.
  + res.status(201).json(newUser);: Bu satır, yeni oluşturulan kullanıcıyı geri döner ve HTTP yanıt kodu olarak **201 (Created)** döner, bu da başarılı bir şekilde veri oluşturulduğunu ifade eder.

// Sunucuyu başlatıyoruz

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

* Bu satırda, Express uygulamamızı başlatıyoruz ve belirtilen PORT üzerinden sunucuyu dinlemeye başlıyoruz.
  + app.listen(PORT, ...): Sunucunun 3000 portu üzerinde çalışmaya başladığını ifade eder.
  + console.log(...): Sunucu başladığında konsola bir mesaj yazdırır. Bu, sunucunun hangi adreste çalıştığını gösterir.

**Özet**

Bu kod, temel bir **Node.js** ve **Express.js** ile yapılmış **RESTful API** örneğidir. Basitçe:

* Bir GET isteğiyle kullanıcı listesini alıyoruz.
* Bir POST isteğiyle yeni bir kullanıcı ekliyoruz.
* 3000 portunda çalışan bir sunucu başlatıyoruz.

Bu sayede kullanıcı verilerini almak ve yeni kullanıcı eklemek için bir API oluşturmuş olduk.

**Adım 5: Sunucuyu Çalıştırma**

* Terminalden API'yi çalıştırmak için şu komutu kullanın:



metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ve sonuç :

metin, ekran görüntüsü, yazı tipi, menü içeren bir resim

Açıklama otomatik olarak oluşturuldu