Nisan 2024

Proje Dokümantasyonu

Back-end Developer Case Question

Önsöz

Bu belge, Invent Analytics şirketinin Backend Developer (Node.js) pozisyonu için sunduğum başvurum kapsamında gerçekleştirdiğim ve sizin değerli incelemenize sunmak için hazırladığım "*Back-end Developer Case Question*" dokümantasyonunu içermektedir.

Dokümantasyonda projenin genel yapısı hakkında kısa bir özet sunulmaktadır. Projenin temel fonksiyonaliteleri ve kullanıcıların beklediği özellikler açıklanmaktadır. Aynı zamanda ekran görüntüleri ve proje ile ilgili kısa bir video da içermektedir.

Projeyi oluştururken, geliştirme sürecinde karşılaşılan zorlukları aşmak ve en iyi uygulamaları takip etmek için çaba gösterdim. İsterlere odaklanarak, kullanımı kolay ve esnek bir platform oluşturmayı amaçladım.

Umarım bu dökümantasyon değerlendirmenize yardımcı olur.

Saygılarımla,

Müberra BÜLBÜL

Proje Tanımı ve Amaç

Bu proje, bir kütüphane için üyelerin ve kitapların ödünç alınması işlemlerini yönetmek amacıyla bir uygulama geliştirme gereksinimini karşılamaktadır. Geliştirilecek uygulama, kullanıcıların kütüphane kaynaklarını görüntüleyebilmesini, kullanıcı bilgilerine erişebilmesini, yeni kullanıcılar ekleyebilmesini, kitapları görüntüleyebilmesini ve ödünç alabilmesini, ödünç alınan kitapları iade edebilmesini ve kitaplara değerlendirme puanı verebilmesini sağlamalıdır.

Kullanılan Teknolojiler:

- **Node.js:** JavaScript çalışma zamanı ortamıdır. Sunucu tarafı uygulamaları geliştirmek için kullanıldı.
- **Express.js:** Node.js tabanlı bir web uygulama çatısıdır. Web uygulamalarının hızlı ve kolay bir şekilde geliştirilmesine olanak sağladığı için kullanıldı.
- **Veritabanı Yönetim Sistemi (PostgreSQL):** İlişkisel veritabanı yönetim sistemidir. Verileri saklamak, yönetmek ve sorgulamak için kullanıldı.
- **ORM Kütüphanesi (Sequelize):** JavaScript ile ilişkisel veritabanı işlemlerini kolaylaştıran bir ORM (Object-Relational Mapping) kütüphanesidir. Nesne yönelimli olarak veritabanı işlemleri yapmayı sağladığı için kullanıldı.
- **Veri Doğrulama Kütüphanesi (Joi):** JavaScript ile giriş verilerinin doğrulanmasını sağlayan bir kütüphanedir. Veri doğrulama ve şema tanımlama için kullanıldı.
- **Express Validator:** Express.js ortamında gelen isteklerin gövde verilerini doğrulamak için kullanılan bir kütüphanedir. Giriş doğrulama ve veri doğrulama için kullanıldı.

Bu teknolojilerin bir araya gelmesi verimli bir RESTful API'nin temelini oluşturmuştur.

Proje Yapısı ve Modüller

Projenin yapısı, modüler bir yaklaşımı benimseyerek, farklı işlevleri ayrı modüllere böler. Her bir modül, belirli bir işlevi veya bileşeni temsil eder ve proje kodlarının düzenli ve yönetilebilir olmasını sağlar.

- **Routes (Yönlendiriciler):** HTTP isteklerini işlemek ve yönlendirmek için kullanılan bu modül, farklı API endpoint'lerini yönetir ve istekleri doğru işleme yönlendirir. *API rotalarının tanımlandığı dosyaları içerir.*

- **Controllers (Denetleyiciler):** Routes modülünden gelen istekleri işleyen ve gerekli veri manipülasyonunu gerçekleştiren bu modül, iş mantığını yönetir. *API rotalarının iş mantığı ve veri işlemlerini yöneten kontrolcü dosyaları içerir.*

- **Models (Modeller):** Veritabanı tablolarını temsil eden ve veri işlemlerini gerçekleştiren bu modül, Sequelize ORM kullanılarak oluşturulur. *Veritabanı modellerinin tanımlandığı dosyaları içerir.*

- **Migrations (Göçler):** Veritabanı şemalarını kontrol eden ve yöneten dosyaların bulunduğu klasördür. Bu dosyalar, veritabanı şemasının değişikliklerini kolayca izlemeyi ve uygulamayı sağlar. *Veritabanı şemasının değişikliklerini kontrol etmek ve yönetmek için kullanılan dosyaları içerir.*

- **Config (Yapılandırma):** Projede kullanılan yapılandırma dosyalarının bulunduğu klasördür. *Proje yapılandırma dosyalarını içerir (örneğin, veritabanı bağlantısı gibi).*

- **Validators (Doğrulayıcılar):** Kullanıcı girdilerini doğrulamak için kullanılan dosyaların bulunduğu klasördür. Bu dosyalar, gelen verilerin doğruluğunu kontrol etmek ve hataları yönetmek için kullanılır. *Gelen istek verilerinin doğrulandığı dosyaları içerir (Joi veya Express Validator gibi kütüphanelerle).*

Bu modüller, projenin parçalarını organize ederek kod tekrarını azaltır, bakımı kolaylaştırır ve geliştirme sürecini daha verimli hale getirir.

Kurulum ve Başlangıç

- **Bağımlılıkların Yüklenmesi:**

  Terminalde projenin kök dizinine gidin ve aşağıdaki komutu çalıştırarak projenin bağımlılıklarını yükleyin:

  ```
  npm install
  ```

- **Veritabanı Bağlantısının Ayarlanması:**

  config/config.json dosyasını açın ve aşağıdaki bağlantı bilgilerini düzenleyin:

  ```
  "development": {
    "username": "postgres_username",
    "password": "postgres_password",
    "database": "database_name",
    "host": "127.0.0.1",
    "dialect": "postgres"
  }
  ```

  ➔ **username:** PostgreSQL veritabanına erişmek için kullanılan kullanıcı adı.
  ➔ **password:** PostgreSQL veritabanına erişmek için kullanılan şifre.
  ➔ **database:** Kullanılacak olan PostgreSQL veritabanı adı.
  ➔ **host:** PostgreSQL veritabanına erişilecek olan adres (varsayılan olarak 127.0.0.1 olarak belirtilmiştir).
  ➔ **dialect:** Kullanılan veritabanı türü (varsayılan olarak postgres olarak belirtilmiştir).

- **Veritabanı Migration'larının Uygulanması:**

  Terminalde aşağıdaki komutu çalıştırarak veritabanı migration'larını uygulayın:

  ```
  npx sequelize-cli db:migrate
  ```

- **Projeyi Başlatma:**

  Terminalde projeyi başlatmak için aşağıdaki komutu çalıştırın:

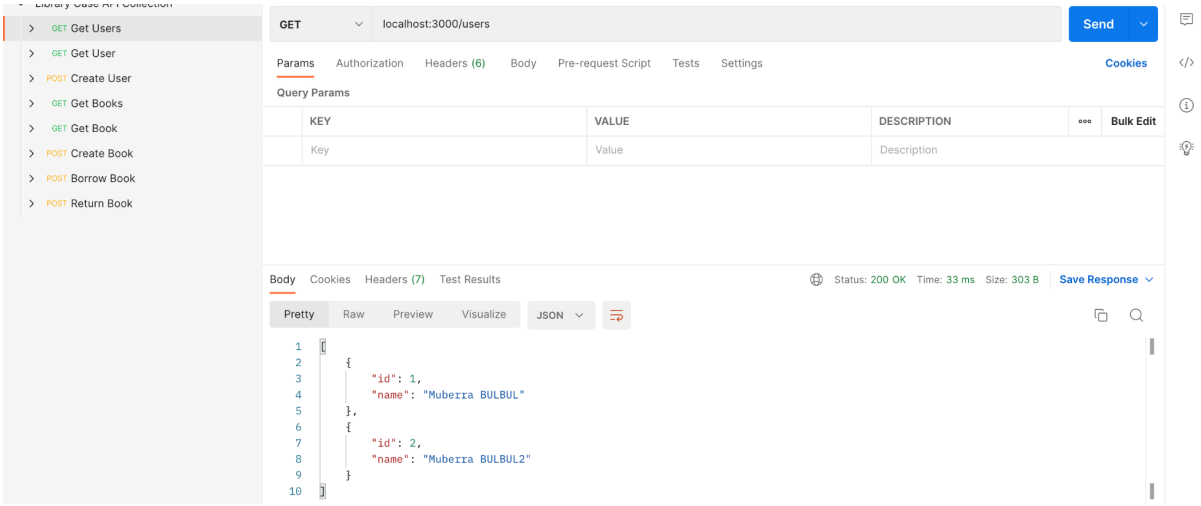
  ```
  npm start
  ```

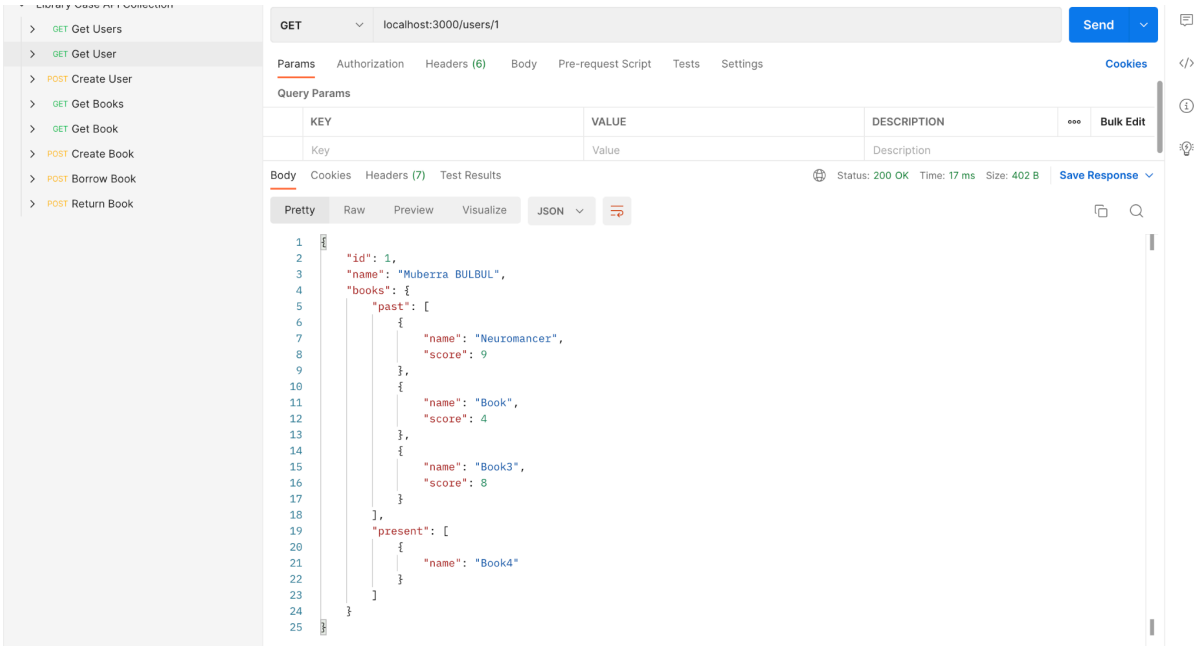
Kullanım Kılavuzu & API Belgesi

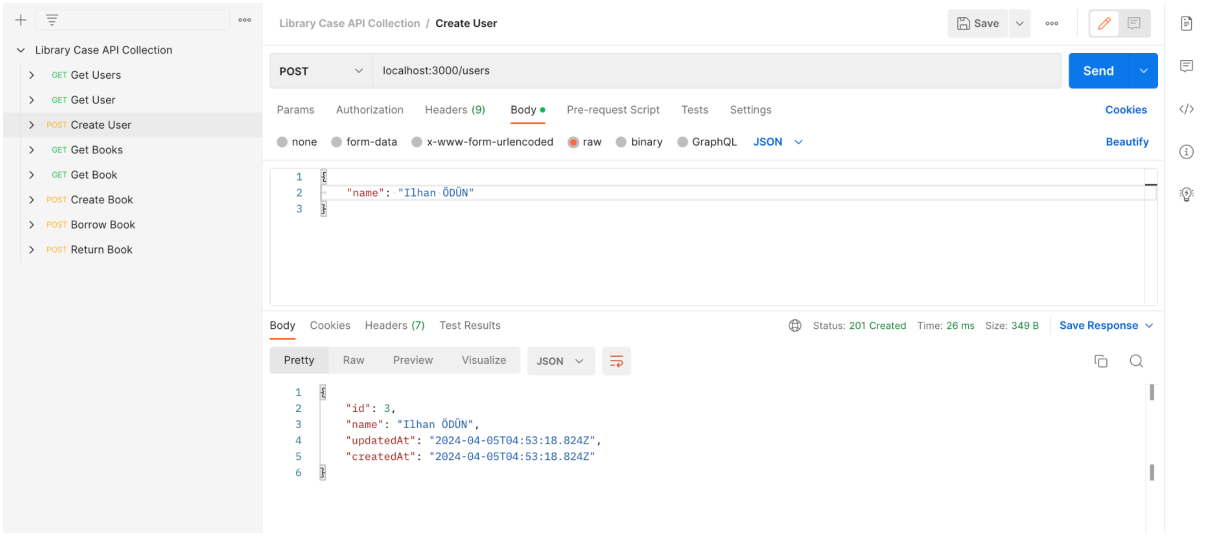
Projeyi geliştirirken gönderdiğiniz Postman Collection'a uygun bir formatta istekleri alıp yanıt döndürmeyi sağladım. Uygulamada hatalı durumlar için (örneğin, mevcut olmayan bir kitabın ödünç alınmaya çalışılması, mevcut olmayan bir kullanıcının kitap almaya çalışması, ödünç alınmış bir kitabın başkası tarafından tekrar ödünç almaya çalışılması vb.) uygun hata işlemlerini gerçekleştirdim. Bu hatalar durumunda API yanıtında hatanın bulunduğunu belirttim.

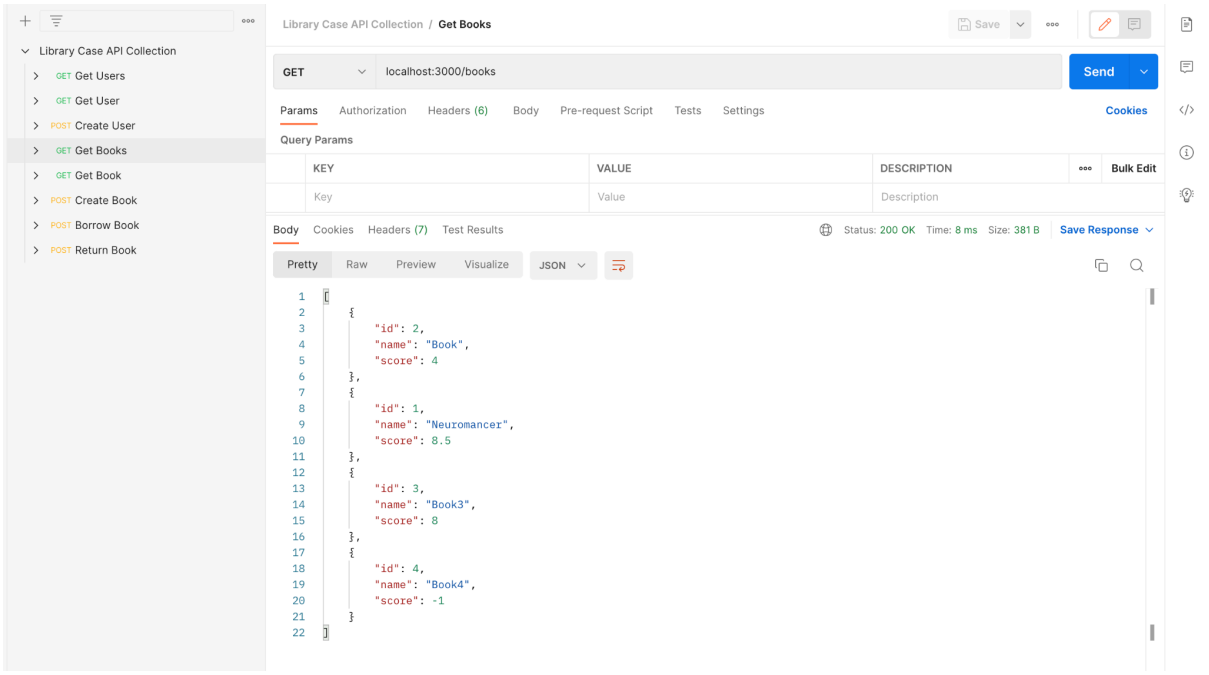
Aşağıda Postman' de API'leri test ederken aldığım video bazı ekran görüntüleri yer almaktadır.

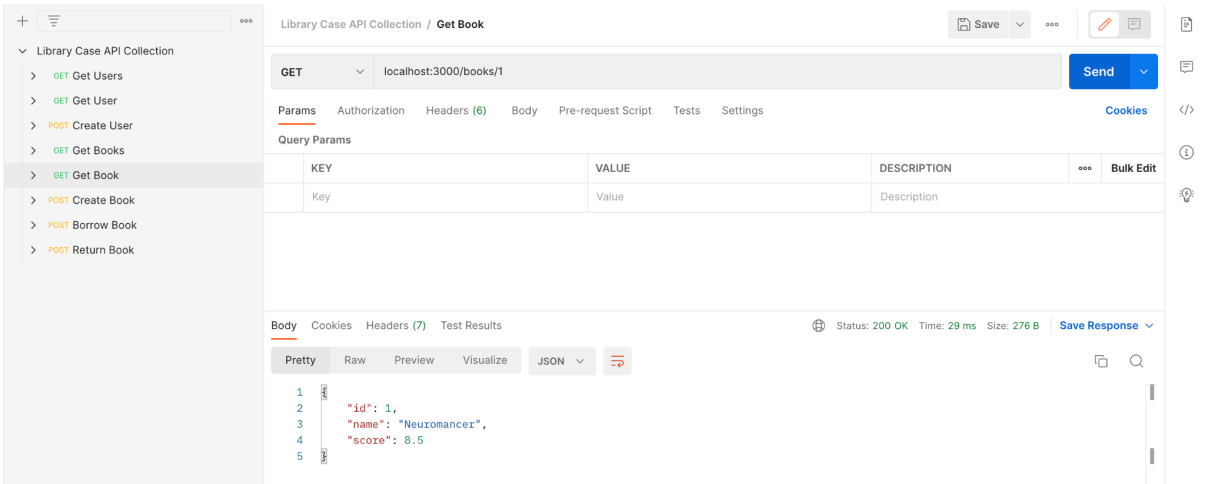
Video Linki :
https://drive.google.com/file/d/1-iE4CrV51w5w3VLp-SpDgBS9a172Ivh_/view?usp=drive_link

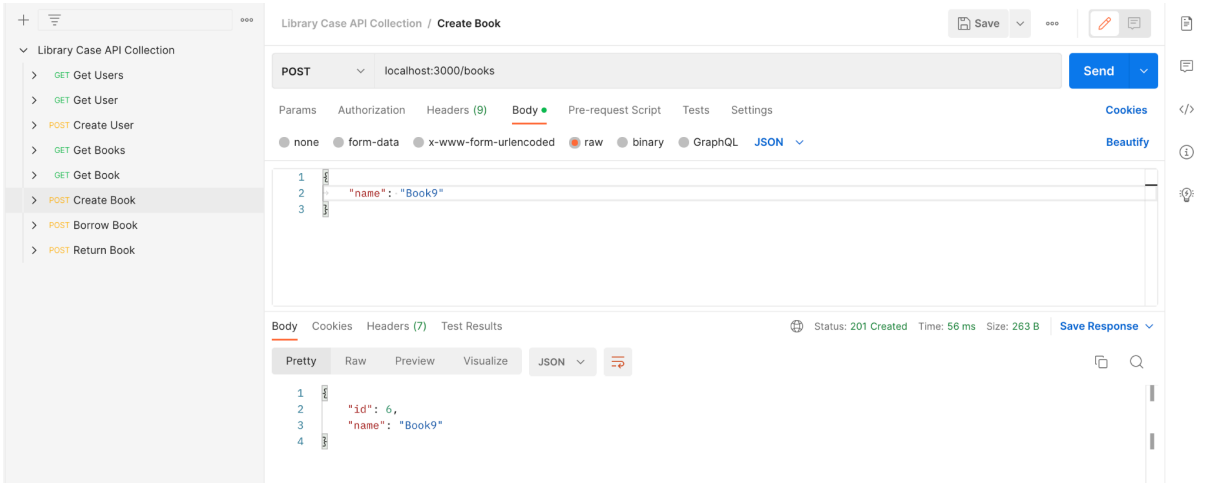


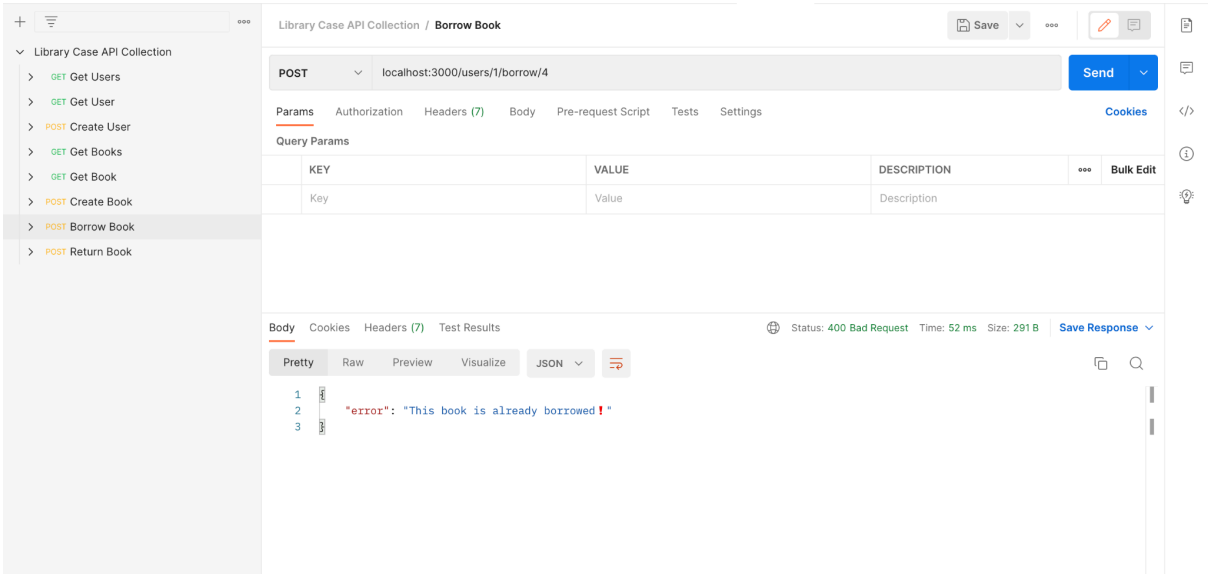


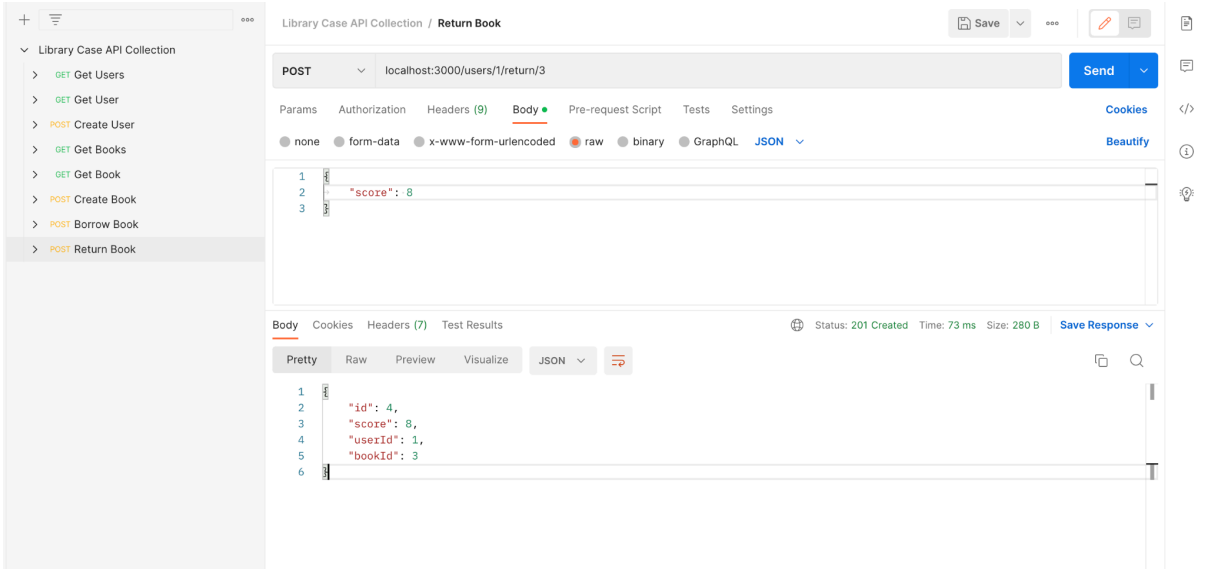

Library Case API Collection
GET Get Users
GET Get User
POST Create User
GET Get Books
GET Get Book
POST Create Book
POST Borrow Book
POST Return Book
Library Case API Collection / Create User
Save
POST
localhost:3000/users
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"name": "Ilhan ÖDÜN"
}
Body Cookies Headers (7) Test Results
Status: 201 Created Time: 26 ms Size: 349 B
Save Response
Pretty Raw Preview Visualize JSON
{
"id": 3,
"name": "Ilhan ÖDÜN",
"updatedAt": "2024-04-05T04:53:18.824Z",
"createdAt": "2024-04-05T04:53:18.824Z"
}


Library Case API Collection
GET Get Users
GET Get User
POST Create User
GET Get Books
GET Get Book
POST Create Book
POST Borrow Book
POST Return Book
Library Case API Collection / Get Books
Save
GET
localhost:3000/books
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 8 ms Size: 381 B
Save Response
Pretty Raw Preview Visualize JSON
[
{
"id": 2,
"name": "Book",
"score": 4
},
{
"id": 1,
"name": "Neuromancer",
"score": 8.5
},
{
"id": 3,
"name": "Book3",
"score": 8
},
{
"id": 4,
"name": "Book4",
"score": -1
}
]


Library Case API Collection
GET Get Users
GET Get User
POST Create User
GET Get Books
GET Get Book
POST Create Book
POST Borrow Book
POST Return Book
Library Case API Collection / Get Book
Save
GET
localhost:3000/books/1
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 29 ms Size: 276 B
Save Response
Pretty Raw Preview Visualize JSON
{
"id": 1,
"name": "Neuromancer",
"score": 8.5
}


Library Case API Collection
GET Get Users
GET Get User
POST Create User
GET Get Books
GET Get Book
POST Create Book
POST Borrow Book
POST Return Book
Library Case API Collection / Create Book
Save
POST
localhost:3000/books
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"name": "Book9"
}
Body Cookies Headers (7) Test Results
Status: 201 Created Time: 56 ms Size: 263 B
Save Response
Pretty Raw Preview Visualize JSON
{
"id": 6,
"name": "Book9"
}


Library Case API Collection
GET Get Users
GET Get User
POST Create User
GET Get Books
GET Get Book
POST Create Book
POST Borrow Book
POST Return Book
Library Case API Collection / Borrow Book
Save
POST
localhost:3000/users/1/borrow/4
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (7) Test Results
Status: 400 Bad Request Time: 52 ms Size: 291 B
Save Response
Pretty Raw Preview Visualize JSON
{
"error": "This book is already borrowed!"
}


Library Case API Collection
GET Get Users
GET Get User
POST Create User
GET Get Books
GET Get Book
POST Create Book
POST Borrow Book
POST Return Book
Library Case API Collection / Return Book
Save
POST
localhost:3000/users/1/return/3
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"score": 8
}
Body Cookies Headers (7) Test Results
Status: 201 Created Time: 73 ms Size: 280 B
Save Response
Pretty Raw Preview Visualize JSON
{
"id": 4,
"score": 8,
"userId": 1,
"bookId": 3
}