

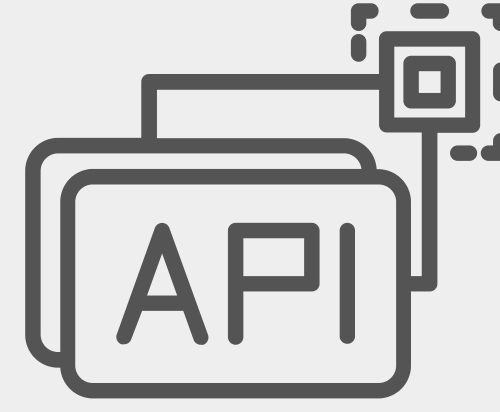
ToDo List API PROJESİ

ToDoListAPI, kullanıcıların yapılacaklar listesini yönetmelerine olanak tanıyan bir RESTful API'dir. Bu API, kullanıcıların görev ekleme, güncelleme, silme ve listeleme işlemlerini kolaylıkla yapabilmelerini sağlar.

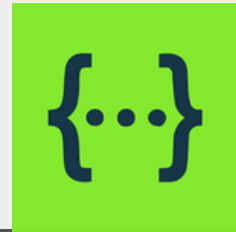
Proje Amacı

Kullanıcıların günlük görevlerini yönetmelerine yardımcı olacak bir API. Görev **ekleme**, **güncelleme**, **silme** ve **listeleme** özelliklerini içerir.

Kullanılan Teknolojiler



- **Backend:** Java 17 ve Spring Boot kullanılarak geliştirilmiştir.
- **Veritabanı:** PostgreSQL ile veri depolama ve yönetimi sağlanmıştır.
- **API Dokümantasyonu:** Swagger (Springdoc OpenAPI) kullanılarak API dokümantasyonu oluşturulmuştur.
- **Diğer Araçlar:**
 - **ModelMapper:** DTO dönüşümleri için kullanılmıştır.



Proje Yapısı Katmanlı Mimari

1. CONTROLLER KATMANI:

- API endpoint'lerini tanımlar.
- Gelen istekleri alır ve iş mantığını çalıştırmak için Service katmanına yönlendirir.
- Kullanıcı dostu yanıtlar döndürür.

2. SERVICE KATMANI:

- İş mantığını içerir.
- Veritabanı işlemleri için Repository katmanı ile iletişim kurar.
- DTO (Data Transfer Object) dönüşümleri ve diğer işlemler burada yapılır.

3. REPOSITORY KATMANI:

- Veritabanı ile doğrudan iletişim kurar.
- CRUD işlemleri için JPA/Hibernate yöntemlerini sağlar.

4. ENTITY KATMANI:

- Veritabanı tablolarını temsil eden sınıfları içerir.

5. DTO KATMANI:

- Entity'lerin direkt olarak dış dünyaya açılmasını engeller

6.EXCEPTION KATMANI

- Proje genelindeki hata yönetimini sağlar.
- Özel hata sınıfları ve global exception handler içerir.

7.CONFIG KATMANI

- Swagger yapılandırması

API Özellikleri

- **GÖREV YÖNETİMİ:**

- Yeni görev ekleme
- Mevcut görevi güncelleme
- Görev silme
- Tüm görevleri listeleme
- ID'ye göre görev getirme

- **DOKÜMANTASYON:**

- Swagger UI ile etkileşimli API dokümantasyonu sağlanmıştır.

- **HATA YÖNETİMİ:**

- Global exception handling ile özelleştirilmiş BaseException ve ErrorMessage sınıfları ile hata yönetilmiştir.
-

API Endpoints Tablosu

HTTP Metodu	Endpoint	Açıklama	Girdi Parametreleri
GET	/api/todos	Tüm görevleri listeleme	Yok
GET	/api/todos/{id}	ID'ye göre görev getirme	id (Path)
POST	/api/todos	Yeni görev oluşturma	Gövde (JSON)
PUT	/api/todos/{id}	Mevcut bir görevi güncelleme	id (Path), Gövde
DELETE	/api/todos/{id}	Görev silme	id (Path)

Hata Yönetimleri

GET

Code	Details
400 <i>Undocumented</i>	Error: response status is 400
Response body	
<pre>{ "status": 400, "exception": { "hostName": null, "path": "/api/todos/4777", "createTime": "2025-01-11T10:31:16.250+00:00", "message": "Kayıt Bulunamadı : Kayıt bulunamadı: ID 4777" } }</pre>	

DELETE

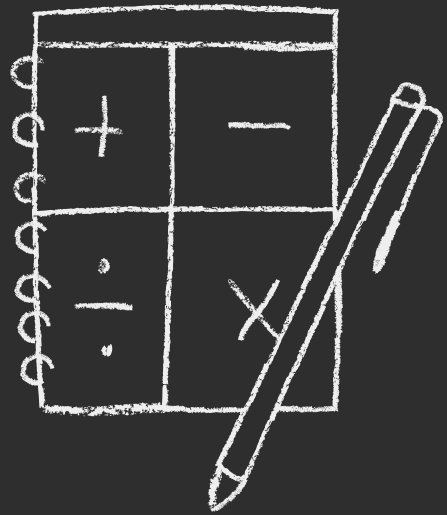
Code	Details
400 <i>Undocumented</i>	Error: response status is 400
Response body	
<pre>{ "status": 400, "exception": { "hostName": null, "path": "/api/todos/20854", "createTime": "2025-01-11T11:19:52.280+00:00", "message": "Kayıt Bulunamadı : Silinecek kayıt bulunamadı: ID 20854" } }</pre>	

POST

Code	Details
400 <i>Undocumented</i>	Error: response status is 400
Response body	
<pre>{ "status": 400, "exception": { "hostName": null, "path": "/api/todos", "createTime": "2025-01-11T10:19:21.909+00:00", "message": "Genel Bir Hata Oluştı : Aynı başlığa sahip bir Todo : " } }</pre>	

PUT

400 <i>Undocumented</i>	Error: response status is 400
Response body	
<pre>{ "status": 400, "exception": { "hostName": null, "path": "/api/todos/788", "createTime": "2025-01-11T10:28:25.036+00:00", "message": "Kayıt Bulunamadı : Güncellenecek kayıt bulunamadı: ID 788" } }</pre>	



Swagger Ekran Görüntüleri

ENDPOINTS

Servers

http://localhost:4444 - Generated server url

todo-controller

GET /api/todos/{id}

PUT /api/todos/{id}

DELETE /api/todos/{id}

GET /api/todos

POST /api/todos



POST

/api/notes

Parameters

No parameters

Request body

```

{
  "id": 0,
  "text": "test",
  "is_completed": false,
  "created_at": "2020-01-01T00:00:00",
  "updated_at": "2020-01-01T00:00:00"
}

```

Execute

200 OK

200 OK

Response body

```

{
  "id": "0",
  "text": "test successfully created!"
}

```

Response headers

```

content-type: text/plain
content-type: application/json
date: Wed, 01 Jan 2020 00:00:00 GMT
server: gunicorn/19.9.0
transfer-encoding: chunked

```

```

// Request body
const body = {
  "id": 1,
  "name": "John",
  "email": "john.doe@example.com",
  "password": "123456"
};

// Request headers
const headers = {
  "Content-Type": "application/json"
};

// Request options
const options = {
  method: "POST",
  headers: headers,
  body: JSON.stringify(body)
};

// Fetch API call
fetch("https://api.example.com/users", options)
  .then(response => {
    if (!response.ok) {
      throw new Error("HTTP error, status = " + response.status);
    }
    return response.json();
  })
  .then(data => {
    console.log("User created successfully:", data);
  })
  .catch(error => {
    console.error("Error:", error);
  });

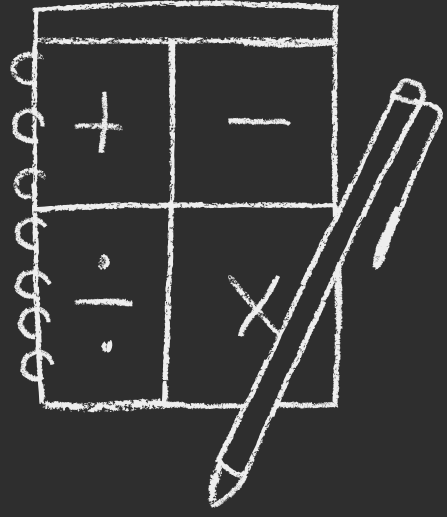
```

[illegible]

The screenshot shows a Node.js REPL environment. At the top, there's a header bar with "Node" and "Repl" tabs. Below it, the prompt "js>" is followed by a red error message "SyntaxError". The user has entered the command "fetch('http://api.github.com/users/octocat').then(res => res.json())" which has been executed. The output shows the JSON response from the GitHub API for the user octocat.

```
js> fetch('http://api.github.com/users/octocat').then(res => res.json())  
[{"login": "octocat", "id": 1, "avatar_url": "https://avatars.githubusercontent.com/u/1?v=4", "gravatar_id": "", "name": "Octocat", "company": null, "blog": null, "location": null, "email": null, "hireable": true, "is_verified": true, "created_at": "2011-12-02T13:29:31Z", "updated_at": "2015-05-12T13:29:31Z", "public_repos": 2, "public_gists": 0, "followers": 26, "following": 0, "subscriptions": 0, "organizations": 0, "repos": [{"id": 1, "owner": {"login": "octocat", "id": 1}, "repo": "HelloWorld", "url": "https://api.github.com/repos/octocat/HelloWorld"}], "gists": []}]
```

Başarısız



Swagger Ekran Görüntüleri

Delete İşlemleri

DELETE /api/todos/{id}

Parameters

Name	Description
id	Integer (int64)

Execute

Responses

Call

```
curl -X DELETE 'http://localhost:3000/api/todos/1' -H 'Accept: */*'
```

Response (200)

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Server response

Code **Details**

400 **Unprocessable Entity** Error: response status is 400

Response body

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Başarılı

DELETE /api/todos/{id}

Parameters

Name	Description
id	Integer (int64)

Execute

Responses

Call

```
curl -X DELETE 'http://localhost:3000/api/todos/1' -H 'Accept: */*' -H 'Content-Type: application/json'
```

Response (200)

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Server response

Code **Details**

200 **OK**

Response body

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Başarısız

Get İşlemleri (/id)

GET /api/todos/{id}

Parameters

Name	Description
id	Integer (int64)

Execute

Responses

Call

```
curl -X GET 'http://localhost:3000/api/todos/1' -H 'Accept: */*' -H 'Content-Type: application/json'
```

Response (200)

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Server response

Code **Details**

200 **OK**

Response body

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Başarılı

GET /api/todos/{id}

Parameters

Name	Description
id	Integer (int64)

Execute

Responses

Call

```
curl -X GET 'http://localhost:3000/api/todos/1' -H 'Accept: */*' -H 'Content-Type: application/json'
```

Response (200)

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Server response

Code **Details**

400 **Unprocessable Entity** Error: response status is 400

Response body

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Başarısız

Get İşlemleri

GET /api/todos

Parameters

Name	Description
id	Integer (int64)

Execute

Responses

Call

```
curl -X GET 'http://localhost:3000/api/todos' -H 'Accept: */*' -H 'Content-Type: application/json'
```

Response (200)

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```

Server response

Code **Details**

200 **OK**

Response body

```
{ "id": 1, "text": "Learn to use Swagger", "completed": false }
```



teşekkürler.

- CEYLAN YILDIRIM
-