

İnnova-Patika Spring Bootcamp Bitirme Projesi

Kimlik numarasıyla kredi sorgulama

Öğretmen: Hamit Mızrak
Öğrenci: Ozan Aydoğan

Projenin amacı

Sisteme Ad, Soyad, Email, Kimlik numarası, Maaş ve Telefon Numarası bilgileriyle kayıt olan kişilerin, Kredi skorlarına ve maaş bilgilerine göre alabilecekleri kredi miktarının hesaplanmasıdır.

Kullanılan teknolojiler

*Spring Framework

*ReactJs

*MySQL

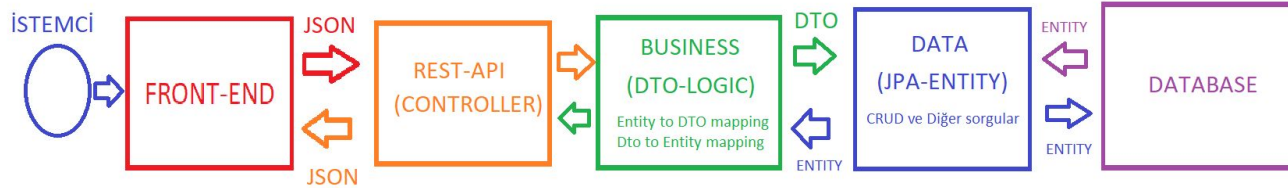
Proje'ye genel bakış

Kullanıcıların sistemi kullanabilecekleri front-end yapısı ReactJS ile geliştirildi. Proje kaynak kodları, herhangi bir port üzerinde çalıştırılıp, tarayıcı üzerinden bu sayfalara erişilebilir. Ön yüz sayfasında kullanıcılar sisteme kayıt olabilir. mevcut bilgilerini güncelleyebilir, kullanıcı bilgilerini silebilir ve “Kullanıcı kredi Limiti”ni hesaplayabilir. Bu hesaplama işlemleri için Back-end kısmında Spring ile bir yapı oluşturuldu. Back-end’de oluşturulan end-pointler ve bu end-pointlere yapılan HTTP istekleriyle (Front-end olmadan, Post-man ile bu istekler yapılabilir), front-end’den Back-end’e gönderilen JSON tipindeki veriler işlenir ve kullanıcılara çıktılar sağlanır (Request - Response). istemci ile sunucu arasındaki bu bağlantı “Rest-API”ler sayesinde sağlanır. Back-end’de oluşturulan End-point yapıları, Front-end kaynak klasörü içerisinde, “service” klasörünün içerisinde tanımlanmıştır. Back-end kaynak kodlarında ise end-point’ler ui.rest klasörü içerisinde bulunmaktadır.

Proje'ye genel bakış

Sisteme kayıt olmak isteyen kullanıcı, ön yüzde gerekli kısımları doldurur. bu veriler JSON tipindeki yapıda key-value mantığıyla tutulur ve back-end'de oluşturulan <http://localhost:8081/api/v1/users> end-pointine “Post” isteğiyle iletilir. ardından JSON olarak gelen bu veriler, DTO yapısı olarak kabul edilir ve JpaRepository yapısıyla MySql veritabanına kayıt edilir. Bu mantığa göre DB'den veri çekme, güncelleme, silme, id'ye göre veri çekme, TC kimlik numarasına göre veri çekme, “TC kimlik numarasına göre Kredi Limiti sorgulama” gibi işlemler, oluşturulmuş olan end-pointler sayesinde gerçekleştirildi.

Projenin yaşam döngüsü



Back-end Mimarisi ve yapılar

Back-end yapısı 5 paketten oluşmaktadır.

ui.rest = istemci sunucu arasındaki bağlantıyı sağlayan api yapıları burada oluşturuldu.

data = database ile alakalı işlemlerin gerçekleşeceği ve Entity yapıları bu pakette bulunuyor

business = Servis yapıları ve DTO yapıları bu katmanda bulunmaktadır.

core = projenin herhangi bir yerinde kullanılabilecek ortak yapılar bu katmanda bulunuyor

bean = bean yapılarının bulunduğu katman

ui.rest

CheckRestController ve UserRestController olarak 2 yapıdan oluşur.

UserRestController yapısında, istemci tarafından, “Kullanıcı” yapısının Kaydının oluşturulması, güncellenmesi, silinmesi, id’ye göre kullanıcının alınması ve veritabanındaki tüm kullanıcıların listesinin dönülmesi gibi işlemleri gerçekleştirecek end-pointler oluşturuldu ve bu yapı kullanıma sunuldu

ui.rest (UserRestController)

//LIST

// http://localhost:8081/api/v1/users

@GetMapping("/users")

//FIND

// http://localhost:8081/api/v1/users/1

@GetMapping("/users/{id}")

//UPDATE

// http://localhost:8081/api/v1/users/2

@PutMapping("/users/{id}")

//SAVE

// http://localhost:8081/api/v1/users

@PostMapping("/users")

//DELETE

// http://localhost:8081/api/v1/users/7

@DeleteMapping("/users/{id}")

ui.rest (CheckRestController)

// Kullanıcı kimlik numarasına göre kullanıcıyı veritabanından çekme

//http://localhost:8081/api/v1/users/citizenshipNumber/1

@GetMapping("/users/citizenshipNumber/{citizenshipNumber}")

// Kullanıcı kimlik numarasına göre kullanıcın kredi limitini sorgulama

//http://localhost:8081/api/v1/users/citizenshipNumber/creditscore/1

@GetMapping("/users/citizenshipNumber/creditscore/{citizenshipNumber}")

ui.rest ve Service bağlantısı

ui.rest katmanında bulunan controller yapılarında tanımlı olan ve Business katmanında daha önce oluşturulmuş olan @Servis annotasyolu “CheckCreditScoreService” ve “UserServices” yapılarına, bu yapıların interface’lerini @Autowired olarak tanımladıktan sonra erişilebilir. Böyle ui.rest katmanındaki controller yapılarında bu servis yapılarının methodlarına ve özelliklerine erişip gerekli işlemleri gerçekleştirebiliriz.

Business katmanı

Business katmanında, bizim kullanacağımız logic işlemleri yapılmaktadır. Bu servis yapılarında tanımladığımız “**Data**” katmanının öğeleriyle, örneğin “UserService” yapısında tanımlanan “**@Autowired private IUserRepository IUserRepository;**” yapıyla, veritabanına gerçekleştirilecek CRUD işlemleri gerçekleştirilir. CheckCreditScore servisiyle, bu yapıda tanımlanan **@Autowired private IMyRepository myRepositoryImp;** ve **@Autowired private IUserRepository IUserRepository;** gibi yapılarla, veritabanından, kullanıcının kimlik numarasına göre sorgular yapıp, gerçekleşen logic işlemlerinin ardından tekrar kullanıcıları veritabanına kayıt işlemleri gerçekleştirilebilir.

Kısaca, Data katmanında bulunan yapılar sayesinde veritabanından yapılan sorgular ve elde edilen entity yapıları üzerinde gerçekleşen logic işlemleri Business katmanında gerçekleşir

DTO ve Entity'ler

İstemciden alınan ve istemciye gönderilen veriler JSON şeklinde transfer edilir. end-point yapılarında bulunan Servislerde, **JSON-DTO** dönüşüm işlemleri gerçekleşir. Dönüşüm işlemleri sonucunda back-end'imizde kullanabileceğimiz user verileri obje içerisinde tutulur (DTO). Verilerimizi, veritabanına "**Entity**" şeklinde kaydederiz.

DATA katmanı (entity)

Veri tabanına verilerimizi entity olarak kaydederiz. Projemizde “BaseEntity” abstract classı ve “UserEntity” adında BaseEntity’i extends eden bir Entity yapımız bulunmaktadır. Bu yapı sayesinde kullanıcıların verileri saklanır. **Kullanıcıların Kimlik numaraları ve ID’leri unique olarak tanımlanmıştır. Bunun sebebi gerçek hayatta aynı kimlik numarasına sahip 2 kişi yoktur. veri tabanı işlemlerinin gerçekleşebilmesi içinde aynı id’ye sahip 2 entity bulunmamalıdır.**

DATA katmanı (repository)

Entity yapılarının veritabanı işlemleri için “Repository” yapıları kullanılır. Projemizde IMyRepository ve IUserRepository adında 2 repository interface yapısı bulunmaktadır. IUserRepository yapısı JpaRepository yapısını implement etmektedir. böylece user ile alakalı gerçekleştirilecek veritabanı işlemleri için bize fonksiyonlar sağlanır. IMyRepository yapısıyla, veritabanından “kimlik numarası”na göre gerçekleştirilecek sorgu işlemleri için kullanılacak olan method tanımlandı ve bu interface’i implement eden MyRepositoryImp yapısında, veritabanı sorgu işlemini gerçekleştirilecek yapı kuruldu.

Log dosyası

@Log4j2 yapısı sayesinde, gerçekleşen işlemler, log klasörü altında bulunan patika.log dosyasında kaydedilir.

görselde de görüldüğü gibi en son gerçekleşen işlem 2461970108 kimlik numaralı kişinin kredi skorunun hesaplanması.

```
2022-02-27 22:06:51.046 INFO 9628 --- [http-nio-8081-exec-9] c.o.ui.rest.UserRestController : user = UserDto(id=4, firstName=omer, lastName=ustunay, creditLimit=null, citizenshipNumber=27710901078, creditScore=245,
2022-02-27 22:06:52.971 INFO 9628 --- [http-nio-8081-exec-10] c.o.ui.rest.UserRestController : user = UserDto(id=3, firstName=fırat, lastName=albayrak, creditLimit=2712.0, citizenshipNumber=24410700178, creditScore=
2022-02-27 22:06:52.971 INFO 9628 --- [http-nio-8081-exec-10] c.o.ui.rest.UserRestController : user = UserDto(id=2, firstName=mehmet, lastName=komurcu, creditLimit=20000.0, citizenshipNumber=22410701078, creditScore=
2022-02-27 22:06:52.971 INFO 9628 --- [http-nio-8081-exec-10] c.o.ui.rest.UserRestController : user = UserDto(id=4, firstName=omer, lastName=ustunay, creditLimit=null, citizenshipNumber=27710901078, creditScore=245,
2022-02-27 22:07:25.921 INFO 9628 --- [http-nio-8081-exec-2] c.o.ui.rest.UserRestController : DB'ye eklendiUserDto(id=null, firstName=sait, lastName=Celik, creditLimit=null, citizenshipNumber=24619701087, creditScor
2022-02-27 22:07:25.939 INFO 9628 --- [http-nio-8081-exec-3] c.o.ui.rest.UserRestController : user = UserDto(id=3, firstName=fırat, lastName=albayrak, creditLimit=2712.0, citizenshipNumber=24410700178, creditScore=
2022-02-27 22:07:25.940 INFO 9628 --- [http-nio-8081-exec-3] c.o.ui.rest.UserRestController : user = UserDto(id=2, firstName=mehmet, lastName=komurcu, creditLimit=20000.0, citizenshipNumber=22410701078, creditScore=
2022-02-27 22:07:25.940 INFO 9628 --- [http-nio-8081-exec-3] c.o.ui.rest.UserRestController : user = UserDto(id=4, firstName=omer, lastName=ustunay, creditLimit=null, citizenshipNumber=27710901078, creditScore=245,
2022-02-27 22:07:25.941 INFO 9628 --- [http-nio-8081-exec-3] c.o.ui.rest.UserRestController : user = UserDto(id=5, firstName=sait, lastName=Celik, creditLimit=null, citizenshipNumber=24619701087, creditScore=1578, d
2022-02-27 22:07:27.594 INFO 9628 --- [http-nio-8081-exec-4] c.o.ui.rest.UserRestController : 5 numaralı id getirildi<200 OK OK,UserDto(id=5, firstName=sait, lastName=Celik, creditLimit=null, citizenshipNumber=2461
2022-02-27 22:08:48.400 INFO 9628 --- [http-nio-8081-exec-6] c.o.ui.rest.CheckRestController : 24619701087TC numaralı kisinin Kredi Skoru hesaplandıUserDto(id=5, firstName=sait, lastName=Celik, creditLimit=44892.0, d
```


Test işlemleri (TestUserEntity)

TestUserEntity yapısıyla, `@Autowired IUserRepository userRepository;` yapısında gerçekleşen CRUD işlemlerinin test işlemleri gerçekleşir. Create, Update, FindbyId, Listall, Delete gibi işlemlerin gerçekleşip gerçekleşmediği işlemleri bu test yapısında kontrol edilir.

Tüm test işlemlerinden başarıyla geçti.

Test İşlemleri(TestCheckCreditScoreService)

Kredi Limitinin hesaplandığı “checkScore” methodunun çalışıp çalışmadığını test edebilmek için, görseldeki test işlemi yapılır. yeni bir entity oluşturur ve bu entity’ye belli değerler atanır. checkScore fonksiyonu aktif hale getirilir ve beklenen değer girilir. eğer beklenen değer ile, fonksiyondan çıkan sonuç aynıysa test başarılı sonuçlanır.

```
@Test
void testCheckScore(){
    UserEntity userEntity = UserEntity.builder().firstName("Ozan").lastName("Aydogan").creditScore(1999).emailId("ozanaydogan1@hotmail.com").salary(9955.0).phoneNumber(5434945208L).citizenshipNumber(86349701078L).build();

    userRepository.save(userEntity);

    Assertions.assertEquals( expected: 39820.0, checkCreditScoreService.checkScore(userEntity.getCitizenshipNumber()).getCreditLimit(), message: "hatalı");
}
```

Run: TestCheckCreditScoreService.testCheckScore x

Tests passed: 1 of 1 test - 740 ms

Test Results 740 ms

- TestCheckCreditScoreService 740 ms
 - testCheckScore() 740 ms

Tests passed: 1

Git Run TODO Problems Profiler Terminal Endpoints Build Dependencies Spring

Tests passed: 1 (moments ago)

23:16:37.027 [main] DEBUG org.springframework...
23:16:37.044 [main] DEBUG org.springframework...
23:16:37.111 [main] DEBUG org.springframework...
23:16:37.133 [main] INFO org.springframework...
23:16:37.140 [main] DEBUG org.springframework...
23:16:37.142 [main] DEBUG org.springframework...
23:16:37.142 [main] INFO org.springframework...

Projeden görüntüler

Front-end, Back-end ve veritabanımızı belirli portlarda ayağa kaldırdıktan sonra, front-end kısmında kullanıcıların listelendiği kısım şu şekilde görebiliriz. veritabanında 2 kayıt bulunmaktadır. listelenen tüm kayıtlar, back-end'de tanımladığımız `http://localhost:8081/api/v1/users @GetMapping("/users") public List<UserDto> getAllUsers()` end-pointine erişmemiz sonucunda bize dönen response sayesinde olmaktadır.

User Credit Limit Check App

User List

[Register User](#) [Check User](#)

User First Name	User Last Name	User Email Id	Actions
ozan	Aydogan	ozanaydogan1@hotmail.com	Update Delete View
mehmet	komurcu	mehmetkomurcu@gmail.com	Update Delete View

User Credit Limit Check App

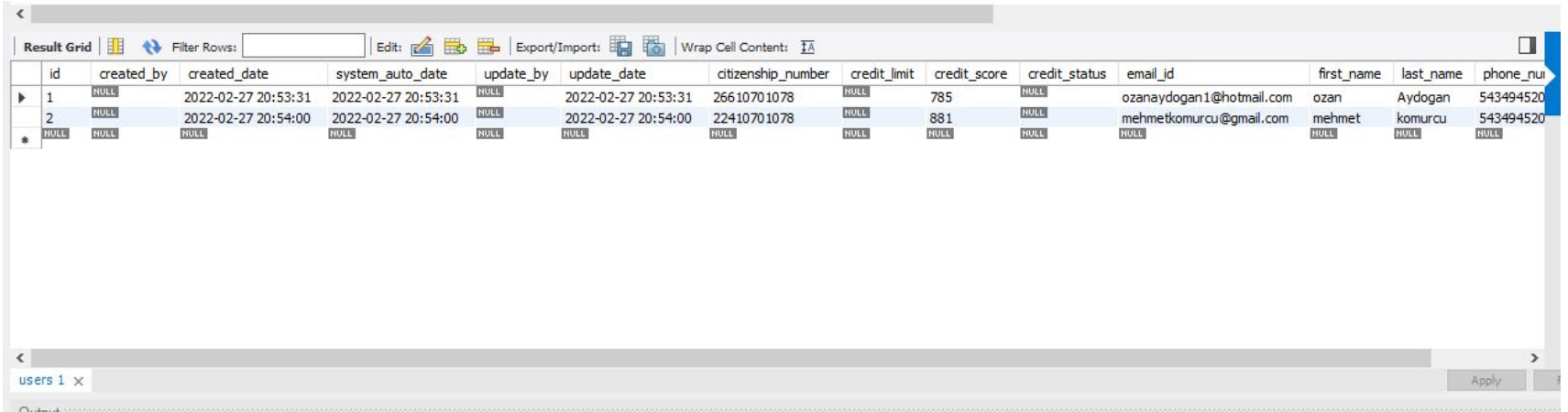
Listelenen kayıtlarda, View butonuna tıklayarak, kullanıcıların tüm verileri görüntülenir. Bu işlem, back-end'de tanımladığımız `http://localhost:8081/api/v1/users/1 @GetMapping("/users/{id}") public ResponseEntity<UserDto> getUserById(@PathVariable Long id)` end-pointine atılan requeste dönen response'un sonucudur. Bu kısımda kullanıcının **User Credit Score**'unu görebiliriz.

View User Details

User First Name:ozan
User Last Name:Aydogan
User Email:ozanaydogan1@hotmail.com
User CitizenshipNumber:26610701078
User Phone Number:5434945208
User Salary:12000
User Credit Score:785

User Credit Limit Check App

Veritabanında kayıtlar görseldeki gibi tutulmaktadır.



	id	created_by	created_date	system_auto_date	update_by	update_date	citizenship_number	credit_limit	credit_score	credit_status	email_id	first_name	last_name	phone_number
▶	1	NULL	2022-02-27 20:53:31	2022-02-27 20:53:31	NULL	2022-02-27 20:53:31	26610701078	NULL	785	NULL	ozanaydogan1@hotmail.com	ozan	Aydogan	543494520
*	2	NULL	2022-02-27 20:54:00	2022-02-27 20:54:00	NULL	2022-02-27 20:54:00	22410701078	NULL	881	NULL	mehmetkomurcu@gmail.com	mehmet	komurcu	543494520

users 1 x Apply

Eğer herhangi bir kullanıcıyı güncellemek istersek Update butonuna tıklarız. Bu buton sayesinde, istemciden alınan veriler JSON içerisinde tutularak, back-end'de oluşturduğumuz `http://localhost:8081/api/v1/users/1 @PutMapping("/users/{id}") public ResponseEntity<UserDto> updateUser(@PathVariable Long id, @RequestBody UserDto userDetails)` endpointine yapılan put işlemi sonucunda gerçekleşir. ozan adlı kaydının maaş bilgisini 12000 olarak belirlemiştik, 13000 olarak değiştirmek istersek,

Update User

First Name:

ozan

Last Name:

Aydogan

Email:

ozanaydogan1@hotmail.com

PhoneNumber:

5434945208

UserSalary:

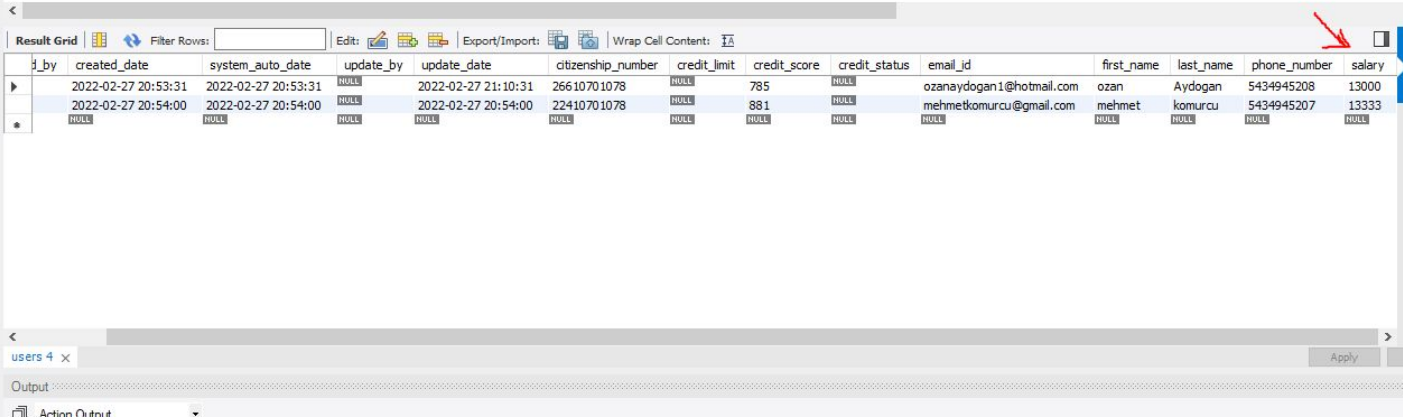
13000

Update

Cancel

User Credit Limit Check App

maaş bilgisi 13000 olarak değiştirildi.



The screenshot shows a data grid interface with a table of user information. The table has 14 columns: id, created_date, system_auto_date, update_by, update_date, citizenship_number, credit_limit, credit_score, credit_status, email_id, first_name, last_name, phone_number, and salary. The first two rows of data are visible. The first row shows a user with a salary of 13000. The second row shows a user with a salary of 13333. A red arrow points to the salary column header in the first row.

id	created_date	system_auto_date	update_by	update_date	citizenship_number	credit_limit	credit_score	credit_status	email_id	first_name	last_name	phone_number	salary
1	2022-02-27 20:53:31	2022-02-27 20:53:31	NULL	2022-02-27 21:10:31	26610701078	NULL	785	NULL	ozanaydogan1@hotmail.com	ozan	Aydogan	5434945208	13000
2	2022-02-27 20:54:00	2022-02-27 20:54:00	NULL	2022-02-27 20:54:00	22410701078	NULL	881	NULL	mehmetkomurcu@gmail.com	mehmet	komurcu	5434945207	13333

Eğer bir kullanıcıyı veri tabanından silmek istersek Delete butonunu kullanabiliriz. back-end'de `http://localhost:8081/api/v1/users/1 @DeleteMapping("/users/{id}") public ResponseEntity<Map<String, Boolean>> deleteUser(@PathVariable Long id)` end-pointine istek atmış oluruz.

Register User

Check User

User First Name	User Last Name	User Email Id	Actions
ozan	Aydogan	ozanaydogan1@hotmail.com	<div>UpdateDeleteView</div>
mehmet	komurcu	mehmetkomurcu@gmail.com	<div>UpdateDeleteView</div>

User Credit Limit Check App

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

id	created_by	created_date	system_auto_date	update_by	update_date	citizenship_number	credit_limit	credit_score	credit_status	email_id	first_name	last_name	phone_n
2		2022-02-27 20:54:00	2022-02-27 20:54:00		2022-02-27 20:54:00	22410701078		881		mehmetkomurcu@gmail.com	mehmet	komurcu	54349452

User List

[Register User](#)[Check User](#)

User First Name	User Last Name	User Email Id	Actions
mehmet	komurcu	mehmetkomurcu@gmail.com	Update Delete View

User Credit Limit Check App

Kullanıcının Maaş bilgisi ve Kredi skoruna göre alabileceği kredi limitini öğrenebilmesi için, Check User butonuna tıklarız(görseldeki sayfaya yönlendiriliriz). ardından karşımıza görüntüdeki gibi ekran gelir

Check Credit Limit

First Name:

Last Name:

Citizenship Number:

Check

User Credit Limit Check App

Kişi eğer sisteme kayıtlıysa, gerekli yerleri doldurur ve Check butonuna tıklar. Kullanıcıdan alınan isim, soyisim ve kimlik numarası bilgilerinden, isim ve soyisim kısmı sadece görüntü olsun diye koyuldu. arka planda gerçekleşen sorgu işlemi sadece kimlik numarasına göre gerçekleşmektedir. Sistemde şu an kayıtlı olan 22410701078 kimlik numaralı kişi kredi limiti sorgulamak isterse, sadece kimlik numarasını yazarakta sorgulayabilir.

Check Credit Limit

First Name:

Last Name:

Citizenship Number:

Check

User Credit Limit Check App

Bu işlemin gerçekleşebilmesi için back-end'de bulunan

<http://localhost:8081/api/v1/users/citizenshipNumber/creditscore/22410701078>

```
@GetMapping("/users/citizenshipNumber/creditscore/{citizenshipNumber}")public  
ResponseEntity<UserDto> getUserCreditScore(@PathVariable(name="citizenshipNumber") Long  
citizenshipNumber)
```

end-pointine istek atılır. ve servis kısmında gerçekleşen logic işlemlerinden sonra bize bir response döner. bu response sonucunda kullanıcı alacağı kredi miktarını ve kredi statusunu görebilir

View User Details

User First Name:mehmet

User Last Name:komurcu

User Email:mehmetkomurcu@gmail.com

User CitizenshipNumber:22410701078

User Phone Number:5434945207

User Salary:13333

User Credit Score:881

User Credit Status:true

User Credit Limit:20000

User Credit Limit Check App

kredi skoru 1000'in üzerinde olan bir kullanıcı için kredi limitinin sorgulanması:

View User Details

User First Name:ırat
User Last Name:albayrak
User Email:ıratılbayrak@gmail.com
User CitizenshipNumber:24410700178
User Phone Number:5434945202
User Salary:11242
User Credit Score:1260
User Credit Status:true
User Credit Limit:44968

User Credit Limit Check App

Eğer kullanıcı maaş bilgisini güncellediyse, Kredi limitini öğrenmek için tekrar sorgu yapabilir. bir önceki görselde bulunan kişinin maaş bilgisi 11242 liraydı. bu maaş bilgisini 678 olarak update edelim ve tekrar sorgu yapalım

Update User

First Name:

Last Name:

Email:

PhoneNumber:

UserSalary:

User Credit Limit Check App

Check Credit Limit

First Name:

Last Name:

Citizenship Number:

User Credit Limit Check App

Sorgu sonucunda çıkan görüntü aşağıdaki gibidir.

View User Details

User First Name: `first`

User Last Name:albayrak

User Email: firatalbayrak@gmail.com

User CitizenshipNumber:24410700178

User Phone Number:5434945202

User Salary:678

User Credit Score:1260

User Credit Status:true

User Credit Limit:2712

User Credit Limit Check App

Post-man çıktıları

//SAVE

```
http://localhost:8081/api/v1/users @PostMapping("/users") public UserDto createUser(@RequestBody  
UserDto userDto)
```

http://localhost:8081/api/v1/users

Save



POST

http://localhost:8081/api/v1/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

```
1 [tab]
2 {
3   "firstName": "omer",
4   "lastName": "ustunay",
5   "emailId": "omerustunay@gmail.com",
6   "citizenshipNumber": "27710901078",
7   "phoneNumber": "5434945290",
8   "salary": 12000.0
9 }
10 [tab]
```

Body

Cookies (1)

Headers (8)

Test Results



Status: 200 OK

Time: 33 ms

Size: 552 B

Save Response

Pretty

Raw

Preview

Visualize

JSON



```
3 {
4   "lastName": "ustunay",
5   "creditlimit": null,
6   "citizenshipNumber": 27710901078,
7   "creditScore": 245,
8   "creditStatus": null,
9   "phoneNumber": 5434945290,
10  "salary": 12000.0,
11  "createdBy": null,
12  "createdDate": null,
13  "updateBy": null,
14  "updatedDate": null,
15  "date": null,
16  "id": 1
17 }
```

[illegible]

Post-man çıktıları

```
//FINDbyID http://localhost:8081/api/v1/users/4 @GetMapping("/users/{id}") public  
ResponseEntity<UserDto> getUserById(@PathVariable Long id)
```

http://localhost:8081/api/v1/users/4

Save



GET

http://localhost:8081/api/v1/users/4

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

Body

Cookies (1)

Headers (8)

Test Results



Status: 200 OK

Time: 16 ms

Size: 630 B

Save Response

Pretty

Raw

Preview

Visualize

JSON



```
1  {
2    "id": 4,
3    "firstName": "omer",
4    "lastName": "ustunay",
5    "creditLimit": null,
6    "citizenshipNumber": 27710901078,
7    "creditScore": 245,
8    "creditStatus": null,
9    "phoneNumber": 5434945290,
10   "salary": 12000.0,
11   "createdBy": null,
12   "createdDate": "2022-02-27T18:51:02.000+00:00",
13   "updateBy": null,
14   "updatedAt": "2022-02-27T18:51:02.000+00:00",
15   "date": "2022-02-27T18:51:02.000+00:00",
16   "emailId": "omerustunay@gmail.com"
17 }
```

Post-man çıktıları

```
//UPDATE http://localhost:8081/api/v1/users/4 @PutMapping("/users/{id}") public  
ResponseEntity<UserDto> updateUser(@PathVariable Long id, @RequestBody UserDto userDetails)
```

omer adlı kullanıcının maaş bilgisini 12000 olarak kaydetmiştik. bu maaş bilgisini 32000 olarak değiştirdikten sonra end-pointe istek atalım

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8081/api/v1/users/4`
- Method:** PUT
- Body:**

```
{  "firstName": "omer",  "lastName": "ustunay",  "emailId": "omerustunay@gmail.com",  "citizenshipNumber": "27710901078",  "phoneNumber": "5434945290",  "salary": 32000.0}
```
- Response:** Status: 200 OK, Time: 21 ms, Size: 553 B
- Response Body (JSON):**

```
{  "firstName": "omer",  "lastName": "ustunay",  "creditLimit": null,  "citizenshipNumber": "27710901078",  "creditScore": null,  "creditStatus": null,  "phoneNumber": "5434945290",  "salary": 32000.0,  "createdBy": null,  "createdDate": null,  "updateBy": null,  "updatedDate": null,  "date": null,  "emailId": "omerustunay@gmail.com"}
```


d_by	created_date	system_auto_date	update_by	update_date	citizenship_number	credit_limit	credit_score	credit_status	email_id	first_name	last_name	phone_number	salary
	2022-02-27 21:32:07	2022-02-27 21:32:07		2022-02-27 21:36:59	24410700178	2712	1260	1	firatalbayrak@gmail.com	firat	albayrak	5434945202	678
	2022-02-27 20:54:00	2022-02-27 20:54:00		2022-02-27 21:30:43	22410701078	20000	881	1	mehmetkomurcu@gmail.com	mehmet	komurcu	5434945207	13333
	2022-02-27 21:51:02	2022-02-27 21:51:02		2022-02-27 21:58:30	27710901078		245		omerustunay@gmail.com	omer	ustunay	5434945290	32000

Post-man çıktıları

//find by CitizenshipNumber

<http://localhost:8081/api/v1/users/citizenshipNumber/27710901078>

```
GetMapping("/users/citizenshipNumber/{citizenshipNumber}") public ResponseEntity<UserDto>  
getUserByCitizenshipNumber(@PathVariable(name="citizenshipNumber") Long citizenshipNumber)
```

http://localhost:8081/api/v1/users/citizenshipNumber/27710901078

GET http://localhost:8081/api/v1/users/citizenshipNumber/27710901078

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "firstName": "omer",
3   ... "lastName": "ustunay",
4   ... "emailId": "omerustunay@gmail.com",
5   ... "citizenshipNumber": "27710901078",
6   ... "phoneNumber": "5434945290",
7   ... "salary": 32000.0
8
9
10 }
```

Body Cookies (1) Headers (8) Test Results

Status:

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "firstName": "omer",
4   "lastName": "ustunay",
5   "creditLimit": null,
6   "citizenshipNumber": 27710901078,
7   "creditScore": 246,
8   "creditStatus": null,
9   "phoneNumber": 5434945290,
10  "salary": 32000.0,
11  "createdBy": null,
12  "createdDate": "2022-02-27T18:51:02.000+00:00",
13  "updateBy": null,
14  "updatedAt": "2022-02-27T18:58:30.000+00:00",
15  "date": "2022-02-27T18:51:02.000+00:00",
16  "emailId": "omerustunay@gmail.com"
17 }
```

Post-man çıktıları (Kullanıcı kredi limiti sorgusu)

// kredi limiti sorgusu

<http://localhost:8081/api/v1/users/citizenshipNumber/creditscore/27710901078>

```
@GetMapping("/users/citizenshipNumber/creditscore/{citizenshipNumber}") public  
ResponseEntity<UserDto> getUserCreditScore(@PathVariable(name="citizenshipNumber") Long  
citizenshipNumber)
```

Kimlik numarasına göre sorgu işlemi gerçekleşti ve bu end-pointten bize bir response döndü. kullanıcının kredi skoru 500'ün altında olduğu için çıktı **FALSE** olarak döndü

The screenshot displays a REST client interface with a GET request to `http://localhost:8081/api/v1/users/citizenshipNumber/creditscore/27710901078`. The request body is a JSON object containing user details. The response status is 200 OK, and the response body is a JSON object containing user details and credit information.

Request:

```
1 {
2   "firstName": "omer",
3   "lastName": "ustunay",
4   "emailId": "omerustunay@gmail.com",
5   "citizenshipNumber": "27710901078",
6   "phoneNumber": "5434945290",
7   "salary": 32000.0
8 }
9
10
```

Response:

```
1 {
2   "id": 4,
3   "firstName": "omer",
4   "lastName": "ustunay",
5   "creditLimit": null,
6   "citizenshipNumber": 27710901078,
7   "creditScore": 245,
8   "creditStatus": false,
9   "phoneNumber": 5434945290,
10  "salary": 32000.0,
11  "createdBy": null,
12  "createdDate": "2022-02-27T18:51:02.000+00:00",
13  "updateBy": null,
14  "updatedDate": "2022-02-27T18:58:30.000+00:00",
15  "date": "2022-02-27T18:51:02.000+00:00",
16  "emailId": "omerustunay@gmail.com"
17 }
```

Kredi skoru 1000'in üzerinde olan "sait" kullanıcısının kredi skorunu hesaplamak istersek

[illegible]

sait kullanıcısının kredi skoru 1000 üzerinde olduğu için bu kullanıcının kredi limiti 44892.0 lira olarak hesaplanır. kredi statü'sü ise **TRUE** olarak döner

The screenshot displays a REST client interface with a GET request to the endpoint `http://localhost:8081/api/v1/users/citizenshipNumber/creditscore/24619701087`. The request body is a JSON object containing user details. The response is also in JSON format, showing a user record with a credit limit of 44892.0 and a credit status of true.

Request:

```
1 {
2   "firstName": "omer",
3   "lastName": "ustunay",
4   "emailId": "omerustunay@gmail.com",
5   "citizenshipNumber": "27710901078",
6   "phoneNumber": "5434945290",
7   "salary": 32000.0
8 }
9
10
```

Response:

```
1 {
2   "id": 5,
3   "firstName": "sait",
4   "lastName": "Celik",
5   "creditLimit": 44892.0,
6   "citizenshipNumber": 24619701087,
7   "creditscore": 1578,
8   "creditStatus": true,
9   "phoneNumber": 5434945602,
10  "salary": 11223.0,
11  "createdBy": null,
12  "createdDate": "2022-02-27T19:07:26.000+00:00",
13  "updateBy": null,
14  "updatedDate": "2022-02-27T19:07:26.000+00:00",
15  "date": "2022-02-27T19:07:26.000+00:00",
16  "emailId": "saitcelik@gmail.com"
17 }
```

[illegible]

OZAN AYDOĞAN

KOCAELİ ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ ÖĞRENCİSİ