



DefineX Java Spring Boot Bootcamp Kredinizde Dokümantasyonu

İlker KUŞ

İÇİNDEKİLER

1. Giriş.....	3
2. Proje Tanımı	3
3. Proje Planı	4
3.1 Zaman Çizelgesi	4
4. Gereksinimler.....	5
4.1 İşlevsel Gereksinimler	5
4.2 Teknik Gereksinimler	5
5. Tasarım.....	6
6. Geliştirme.....	6
6.1 Modeller için oluşturulan End-Pointler:	6
6.2 Kullanılan Tasarım Desenleri:	7
6.3 Gateway/Service Discovery:.....	8
6.4 RabbitMQ/Kafka:	9
6.5 Docker:.....	9
7. Test	10

1. Giriş

Bu proje, kullanıcıların çeşitli bankalarca internet üzerinden sunulan, kredi ve kredi kartı başvurularını yapabilmelerini sağlayan bir örnek bir projedir. Bu uygulama kullanıcıların sisteme kayıt olma, kredi başvurusu yapma, mail adresi ile kullanıcı sorgulama, kullanıcı başvurularını sorgulama gibi temel işlemlere sahiptir. Bu işlevler ile ilgili detaylı bilgi ilerleyen bölümlerde açıklanmıştır.

2. Proje Tanımı

Bu proje, web tabanlı bir Spring Boot uygulaması olarak geliştirilmiştir. Sistemin end-pointleri tanımlanmış ve bu end-pointler için örneklendirmeleri gerçekleştirilmiştir. Sisteme eklenen kullanıcılar, belirtilen kredi türüne göre başvurusunu yapabilir ilgili bankanın servisine yönlendirme işlemi yapılmaktadır. Bu işlemler sonunda veri tabanına kayıt işlemi gerçekleştirilir. Projede kredibizde-servisi, ilgili banka servisleri için MySQL veri tabanı ile kayıt işlemi gerçekleştirilmiştir. Sisteme bir kullanıcı kayıt olduğunda log kaydını tutmak üzere Kafka ile MongoDB kullanılarak kullanıcı kayıt – log işlevi gerçekleştirilmiştir. Aynı zamanda sistemde bulunan Notification servisi ile sisteme belirtilen türe göre bildirim gönderimi geliştirilmesi yapılmıştır.

3. Proje Planı

3.1 Zaman Çizelgesi



Tarih	Görev
09-03-2024	Proje Modellerinin Oluşturulması
10-03-2024	Model implementasyonları gerçekleştirilmesi
16-03-2024	Modeller için end-pointlerin yazılması
17-03-2024	Docker kurulumları gerçekleştirilmesi
23-03-2024	Feign client ile banka servislerinin entegrasyonu
24-03-2024	Kafka ve RabbitMQ ile asenkron iletişim entegrasyonu
30-03-2024	Veri Tabanı modellerinin oluşturulması
31-03-2024	MySQL entegrasyonu
06-04-2024	Servisler için unit test implementasyonu

4. 4. Gereksinimler

4.1 İşlevsel Gereksinimler

- Kullanıcıların sisteme eklenebilmesi
- Kullanıcıların mail adresine göre sorgulanması.
- Kullanıcıların bilgilerinin güncellenmesi.
- Kullanıcıların İhtiyaç, Araç, Konut Kredi tiplerine başvurabilmesi.
- Sisteme banka eklenebilmesi.
- Sisteme bankalar için kredi kartı eklenebilmesi.



4.2 Teknik Gereksinimler

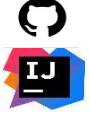
- Spring Boot projesi.
- Banka servisleri ile iletişim için Feign client.
- Notification servis haberleşmesi için RabbitMQ.
- Log servis haberleşmesi için Kafka.
- DB bağlantısı hızlandırmak için Redis.
- MySQL ve MongoDB veritabanı kullanımı
- JUnit ve Mockito ile test gerçekleştirilmesi



5. Tasarım

İlgili proje modelleri ve DB ile ilgili diyagramlar ek dosya olarak paylaşılmıştır.

6. Geliştirme



Projenin geliştirme sürecinde, GitHub versiyon kontrol sistemi kullanılmıştır. Geliştirme ortamı olarak IntelliJ Idea Ultimate sürümü tercih edilmiştir.

6.1 Modeller için oluşturulan End-Pointler:

▼ kredinbizden
▼ akbak-service
POST create applicaton
GET get all applications
▼ garanti-service
POST create application
GET get all applications
▼ kredinbizde-service
▼ user/application
PUT update user
POST save user
GET gel all users
GET get user by email
GET get user by userId
POST save application
GET get all applications
GET get applications by userId
▼ bank/campaign/creditcard
POST save applications creditcard
POST save bank
GET get all banks
GET get banks {name}
POST save credit cards
GET get all credit cards
GET get credit cards {bankname}
GET get all campaigns
▼ gw
POST save user
GET get all users

Akbank Service End-Point

Save Application : localhost:8081/api/akbank/v1/applications

Get All Applications: localhost:8081/api/akbank/v1/applications

Garanti Service End-Point

Save Application : localhost:8085/api/garanti/v1/applications

Get All Applications: localhost:8085/api/garanti/v1/applications

KredinBizde End-Point

Update User : localhost:8080/api/users/{email}

Save User : localhost:8080/api/users

Get All Users : localhost:8080/api/users

Get User With Email : localhost:8080/api/users/{email}

Get User With Id : localhost:8080/api/users/id/{id}

Save Applications : localhost:8080/api/applications

Get All Applications : localhost:8080/api/applications

Get Applications User Id:localhost:8080/api/applications/id/{userId}

Save Bank : localhost:8080/api/banks

Get All Banks : localhost:8080/api/banks

Get Bank With Name : localhost:8080/api/banks/{bankName}

Save Credit Card : localhost:8080/api/creditcards

Get All Credit Cards : localhost:8080/api/creditcards

Get CC. via BankName: localhost:8080/api/creditcards/{bankName}

Get All Campaigns : localhost:8080/api/campaigns

6.2 Kullanılan Tasarım Desenleri:

- **Singleton Pattern :**

Proje gelişim aşamasında Managerlar üzerinde daha sonra spring tarafında ise bean'lerin yaratılmasında kullanılmıştır.

- **Strategy Pattern :**

Notification Servis için kod tabanlı yaklaşımdan, dışarıdan alınan girdiye göre bildirim durumu değişen bir yapı kullanılmıştır.

```
1  {
2      "user": {
3          "name": "cem",
4          "surname": "dirman",
5          "birthDate": "1997-04-23",
6          "email": "cem2@gmail.com",
7          "password": "password",
8          "phoneNumber": 5394443322,
9          "isActive": true,
10         "address": {
11             "addressTitle": "Home",
12             "addressDescription": "İstanbul, Taksim",
13             "province": "İstanbul"
14         },
15         "applicationList": []
16     },
17     "notificationType": "SMS"
18 }
```

- **Builder Pattern :**

Spring anotasyonları ile ApplicationResponse, NotificationDTO üretimlerinde kullanılmıştır.

```
public class MailNotification implements INotificationProducer {
    no usages
    @Override
    public NotificationDTO produceNotification(User user) {
        return NotificationDTO.builder()
            .notificationType(NotificationType.EMAIL)
            .email(user.getEmail())
            .message("User Sisteme Kaydedildi.")
            .build();
    }
}
```


6.3 Gateway/Service Discovery:

User kayıt işlemi yönlendirmesi için gateway örneklendirmesi gerçekleştirilmiştir.

Clientlar bağlanmak üzere bir service-discovery örneklendirmesi gerçekleştirilmiştir.

Gateway : localhost:8084/api/users

Service-Discovery : localhost:8761



System Status

Environment	test
Data center	default

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones
AKBANK-SERVICE	n/a (1)	(1)
GARANTI-SERVICE	n/a (1)	(1)
KREDINBIZDE-GATEWAY	n/a (1)	(1)

Service Discovery Connection Ex.

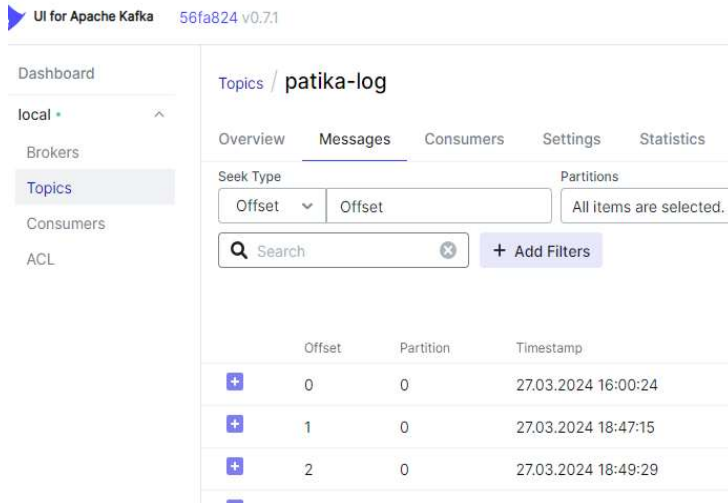
6.4 RabbitMQ/Kafka:

Notification Servisi ile asenkron haberleşme sağlamak amacıyla RabbitMQ konfigürasyonu yapılmıştır.

Log Servisi MongoDB kayıdı yapmak amacı ile asenkron iletişim için Kafka konfigürasyonu yapılmıştır.

RabbitMQ Web Tool : localhost:15672

Kafka UI : localhost:9090



UI for Apache Kafka 56fa824 v0.7.1

Dashboard

local * ^

Brokers

Topics

Consumers

ACL

Topics / patika-log

Overview Messages Consumers Settings Statistics

Seek Type

Offset Offset

Partitions

All items are selected.

Q Search + Add Filters

	Offset	Partition	Timestamp
+	0	0	27.03.2024 16:00:24
+	1	0	27.03.2024 18:47:15
+	2	0	27.03.2024 18:49:29

Kafka Queue



RabbitMQ™ RabbitMQ 3.13.0 Erlang 26.2.2

Overview Connections Channels Exchanges Queues and Streams

Queues

All queues (1)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

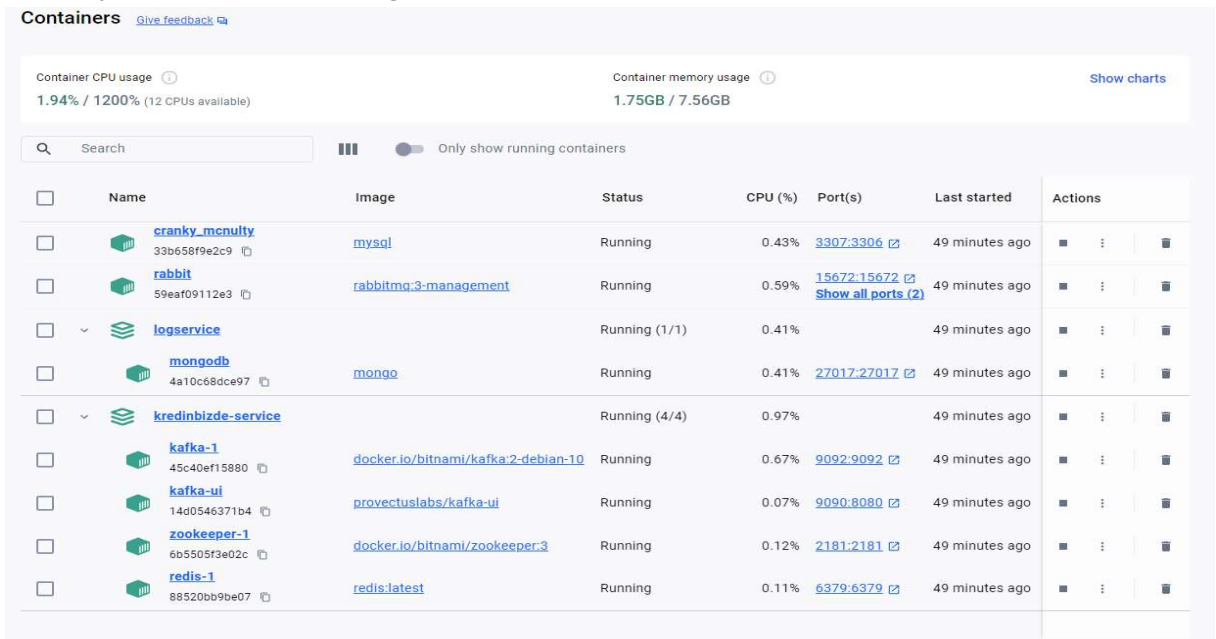
Overview					Messages		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total
/	patika.queue	classic		running	0	0	0

Add a new queue

Rabbit Queue

6.5 Docker:

MySQL, RabbitMQ, MongoDB, Kafka, Redis tooları için docker kullanılmıştır.



Containers Give feedback

Container CPU usage 1.94% / 1200% (12 CPUs available)

Container memory usage 1.75GB / 7.56GB

Show charts

Search

Only show running containers

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	cranky_mcnulty	mysql	Running	0.43%	3307:3306	49 minutes ago	
	rabbit	rabbitmq:3-management	Running	0.59%	15672:15672	49 minutes ago	
	logservice		Running (1/1)	0.41%		49 minutes ago	
	mongodb	mongo	Running	0.41%	27017:27017	49 minutes ago	
	kredinizde-service		Running (4/4)	0.97%		49 minutes ago	
	kafka-1	docker.io/bitnami/kafka:2-debian-10	Running	0.67%	9092:9092	49 minutes ago	
	kafka-ui	provectuslabs/kafka-ui	Running	0.07%	9090:8080	49 minutes ago	
	zookeeper-1	docker.io/bitnami/zookeeper:3	Running	0.12%	2181:2181	49 minutes ago	
	redis-1	redis:latest	Running	0.11%	6379:6379	49 minutes ago	

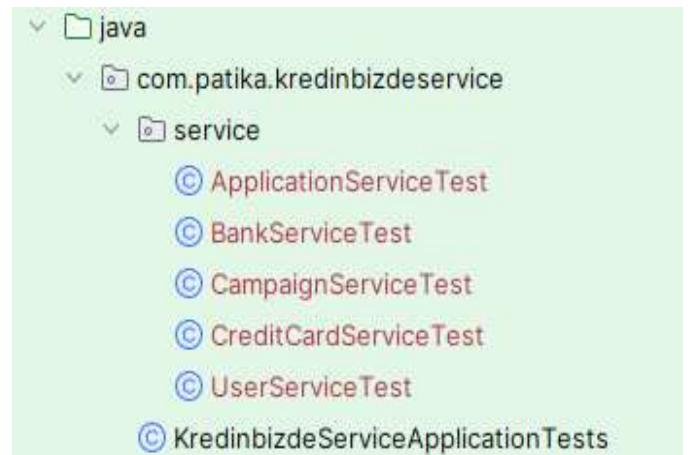
7. Test

Projenin test kısımları için JUnit ve Mockito kütüphaneleri kullanılmış olup, ilgili servisler için create,getAll,getByEmail gibi metotlar için unit testler uygulanmıştır. Tüm unit testler için standartlara uygun isimlendirme yapılmıştır.

İlgili projenin end-pointlerini test için ise ilgili Postman dosyası eklenmiştir.

```
1  {
2    "loan": {
3      "type": "consumer",
4      "amount": 500000,
5      "installment": 12,
6      "bank": {
7        "bankId": 2,
8        "name": "Garanti"
9      },
10     "interestRate": 2.59
11   },
12   "email": "cem2@gmail.com",
13   "bankName": "Garanti"
14 }
```

Application Save Data JSON Ex.



Kredibizde Service Test Package