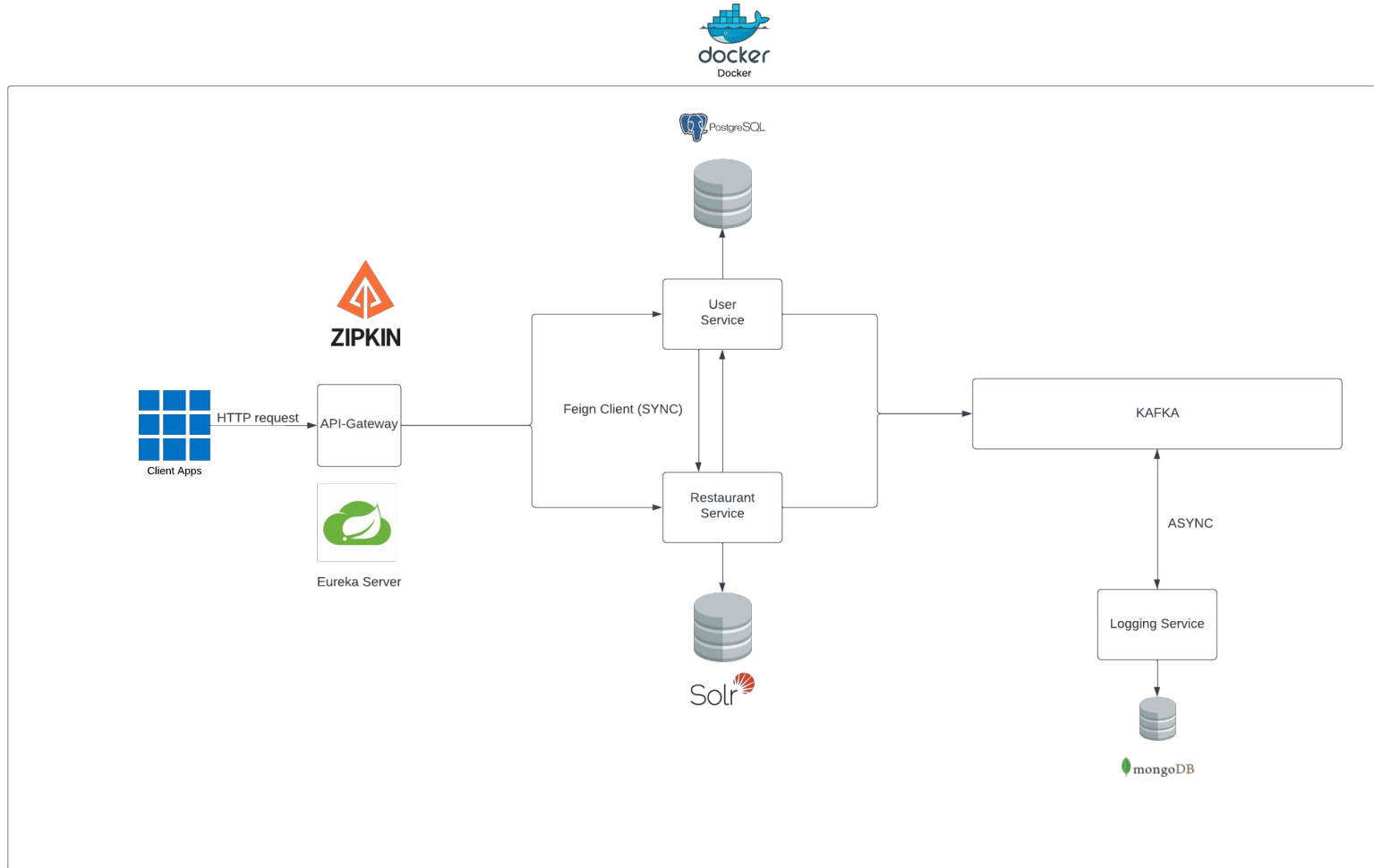



n11 TalentHub Bootcamp Final Project

- Project Owner : Serhat Acar
- Project Name : Dine Experince Applicaton
- Project Description :Utilizing a microservice architecture to manage users, restaurant, user reviews based on location and ratings. It offers personalized restaurant recommendations, handles user and restaurant data, and performs queries on Apache Solr, ensuring efficient data management and a seamless dining experience.

Project architecture and System Design



User Service API

 **Swagger**
powered by SMARTBEAR

/v3/api-docs

Explore

Dine Experience App - User Service API ^{1.0} OAS 3.0

[/v3/api-docs](#)

OpenAPI documentation for Dine Experience App, a Spring Boot REST API as a graduation project for n11 TalentHub Bootcamp

[Contact Serhat Acar](#)

[Source code on GitHub](#)

Servers

http://localhost:8082 - user-service

Authorize

user-controller

PUT

/api/v1/users/{debugId}

PUT request to update a user

GET

/api/v1/users

GET request for all users

POST

/api/v1/users

POST request to save a user

GET

/api/v1/users/{id}

GET request for a user by id

DELETE

/api/v1/users/{id}

DELETE request to delete a user

user-review-controller

PUT

/api/v1/users/reviews

PUT request for user review

POST

/api/v1/users/reviews

POST request to create user review

GET

/api/v1/users/reviews/{id}

GET request for user review by id

DELETE

/api/v1/users/reviews/{id}


DELETE request to delete a user review

GET

/api/v1/users/reviews/user/{userId}

GET request for user reviews by user id

Restaurant Service API

 **Swagger**
OpenAPI

Explore

Dine Experience App - Restaurant Service API 1.0 OAS3


[/v3/api-docs](#)

OpenAPI documentation for Dine Experience App, a Spring Boot REST API as a graduation project for n11 TalentHub Bootcamp

[Contact Serhat Acar](#)

[Source code on GitHub](#)

Servers

Authorize 

restaurant-controller

GET

/api/v1/restaurants

GET request for all restaurants

⌵

PUT

/api/v1/restaurants

PUT request to update a restaurant

⌵

POST

/api/v1/restaurants

POST request to save a restaurant

⌵

GET

/api/v1/restaurants/{id}

GET request for restaurant by id

⌵

DELETE

/api/v1/restaurants/{id}

DELETE request to delete a restaurant

⌵

recommendation-controller

GET

/api/v1/restaurants/recommendations/{userId}

GET request for restaurant recommendations by user id

⌵

Schemas

RestaurantUpdateRequest

>

Swagger Schemas

POST	/api/v1/users/reviews	POST request to create user review	▼
GET	/api/v1/users/reviews/{id}	GET request for user review by id	▼
DELETE	/api/v1/users/reviews/{id}	DELETE request to delete a user review	▼
GET	/api/v1/users/reviews/user/{userId}	GET request for user reviews by user id	▼
GET	/api/v1/users/reviews/restaurant/{restaurantId}	GET request for user reviews by restaurant id	▼
GET	/api/v1/users/reviews/all	GET request for all user reviews	▼

Schemas			^
<div>UserUpdateRequest ▼ { id > [...] name > [...] surname > [...] birthDate > [...] email > [...] gender > [...] userStatus > [...] latitude > [...] longitude > [...] }</div>			
<div>RestResponseUserDTO ▼ { data UserDTO > {...} responseDate > [...] message > [...] success > [...] }</div>			
<div>UserDTO ▼ { id > [...] name > [...] surname > [...] birthDate > [...] email > [...] ... > [...] }</div>			

User and User Reviews Tables

The screenshot displays a database management interface with two tables. The top table, 'users', contains 10 rows of user information. The bottom table, 'user_reviews', contains 10 rows of reviews. The interface includes a search bar, a table selector, and a status bar at the bottom.

users

	id	name	surname	gender	user_status	birth_date	email	latitude	longitude	creator_id	created_at	updated_id	updated_at
1	1	John	Doe	MALE	ACTIVE	1998-01-15	john.doe@example.com	40.74	-74.1	1	2022-01-01 00:00:00.000000	1	2022-01-01 00:00:00.000000
2	2	Jane	Smith	FEMALE	ACTIVE	1985-05-20	jane.smith@example.com	40.7	-73.93	2	2022-01-02 00:00:00.000000	2	2022-01-02 00:00:00.000000
3	3	Bob	Jones	MALE	ACTIVE	1998-08-10	bob.jones@example.com	40.69	-74.07	1	2022-01-03 00:00:00.000000	1	2022-01-03 00:00:00.000000
4	4	Alice	Miller	FEMALE	INACTIVE	1980-03-25	alice.miller@example.com	40.69	-73.99	3	2022-01-04 00:00:00.000000	3	2022-01-04 00:00:00.000000
5	5	David	Wilson	MALE	ACTIVE	1995-12-03	david.wilson@example.com	40.72	-74.05	2	2022-01-05 00:00:00.000000	2	2022-01-05 00:00:00.000000
6	6	Emily	Brown	FEMALE	INACTIVE	1987-09-18	emily.brown@example.com	40.72	-73.95	3	2022-01-06 00:00:00.000000	3	2022-01-06 00:00:00.000000
7	7	Charlie	Anderson	MALE	ACTIVE	1993-07-12	charlie.anderson@example.com	40.68	-73.92	1	2022-01-07 00:00:00.000000	1	2022-01-07 00:00:00.000000
8	8	Olivia	White	FEMALE	INACTIVE	1983-11-30	olivia.white@example.com	40.75	-73.97	2	2022-01-08 00:00:00.000000	2	2022-01-08 00:00:00.000000
9	9	Samuel	Martin	MALE	ACTIVE	1991-04-22	samuel.martin@example.com	40.72	-73.96	3	2022-01-09 00:00:00.000000	3	2022-01-09 00:00:00.000000
10	10	Ava	Thompson	FEMALE	ACTIVE	1989-06-05	ava.thompson@example.com	40.73	-74.05	1	2022-01-10 00:00:00.000000	1	2022-01-10 00:00:00.000000


user_reviews

	id	restaurant_id	user_id	comment	user_rate	creator_id	created_at	updated_at	updated_id
1	1	1		1 Great food!	1.00	3	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	1
2	2	2		2 Good service	2.00	2	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	2
3	3			3 Excellent atmosphere	3.00	4	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	3
4	4			4 Could be better	4.00	1	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	4
5	5			5 Nice place	5.00	3	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	5
6	6			6 Average food	1.00	2	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	6
7	7			7 Loved it!	2.00	4	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	7
8	8			8 Not my favorite	3.00	1	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	8
9	9			9 Good overall	4.00	3	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	9
10	10	10		10 Okay experience	5.00	2	2024-03-11 17:57:10.354737	2024-03-11 17:57:10.354737	10

Database > docker_db > user_service_db > public > tables > user_reviews

SUM: Not enough values

Solr Restaurant Documents



Dashboard

Logging

Security

Core Admin

Java Properties

Thread Dump

restaurants

Overview

Analysis

Documents

Params

Files

Ping

Plugins / Stats

Query

Replication

Schema

Segments info

Request-Handler (qt)

/select

common

q

:

q.op

OR

fq

sort

id asc

start.rows

010

fl

df

paramset(s)

Select paramset(s)...

wt

☒ indent on

☐ debugQuery

defType

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Raw Query Parameters

JSON Query

Execute Query

http://localhost:8983/solr/restaurants/select?indent=true&q.op=OR&q=*&rows=10&sort=id%20asc&useParams=

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 13,
    "params": {
      "q": ":*:*",
      "indent": "true",
      "q.op": "OR",
      "sort": "id asc",
      "rows": "10",
      "useParams": "",
      "_": "1710650858659"
    }
  },
  "response": {
    "numFound": 11,
    "start": 0,
    "numFoundExact": true,
    "docs": [
      {
        "id": "1",
        "name": ["Test Restaurant 1"],
        "address": ["Test Address 1"],
        "phone": "1234567890",
        "email": ["test1@test.com"],
        "description": ["Test Description1"],
        "website": ["www.testrestaurant1.com"],
        "workingHours": ["09:00-18:00"],
        "Latitude": [40.71],
        "Longitude": [-74.05],
        "restaurantRate": [1],
        "status": ["ACTIVE"],
        "version": "179363969979158016"
      },
      {
        "id": "10",
        "name": ["Test Restaurant 10"],
        "address": ["Test Address 10"],
        "phone": "1234567890",
        "email": ["test10@test.com"],
        "description": ["Test Description"],
        "website": ["www.testrestaurant10.com"],
        "workingHours": ["09:00-18:00"],
        "Latitude": [41.71],
        "Longitude": [-71.2],
        "restaurantRate": [5],
        "status": ["ACTIVE"],
        "version": "1793639700303446016"
      },
      {
        "id": "2",
        "name": ["Test Restaurant 2"],
        "address": ["Test Address 2"],
        "phone": "1234567890",
        "email": ["test2@test.com"]
      }
    ]
  }
}
```

Docker-Compose File

```
1 version: '3'
2
3 services:
4   db:
5     image: postgres
6     container_name: postgres_db
7     environment:
8       POSTGRES_USER: postgres
9       POSTGRES_PASSWORD: 123
10      POSTGRES_DB: user_service_db
11      ports:
12        - "5432:5432"
13      networks:
14        - back-tier
15      volumes:
16        - db_data:/var/lib/postgresql/data
17
18   solr:
19     image: solr
20     container_name: solr
21     ports:
22       - "8983:8983"
23     volumes:
24       - data:/var/solr
25     networks:
26       - back-tier
27     command:
28       - solr-precreate
29       - restaurants
30
31   zipkin:
32     image: openzipkin/zipkin
33     container_name: zipkin
34     networks:
35       - back-tier
36     ports:
37       - "9411:9411"
38
39   discovery-server:
40     build:
41       context: ./discovery-server
42       dockerfile: Dockerfile
43     container_name: discovery-server
```

Document 1/1 | services: db: volumes:

Git Commits

The screenshot displays the IntelliJ IDEA interface with the Git Log window open. The window shows a list of commits for the README.md file, sorted by date. The commits are listed in a table with columns for the commit message, the author (Serhat Acar), and the date. The commit messages include various updates to documentation, tests, and code refactoring. The date column shows the commit time in UTC+8.

Commit Message	Author	Date
docs: update documentation	Serhat Acar	A minute ago
chore: add banner for all services	Serhat Acar	Today 01:00
test(user-service): check integration test	Serhat Acar	Today 00:26
refactor(user-service & restaurant-service): enhanced controller advice	Serhat Acar	Today 00:26
feat(frontend): add pages	Serhat Acar	Yesterday 21:31
feat(frontend) : add services and util	Serhat Acar	Yesterday 21:30
feat(frontend): add context	Serhat Acar	Yesterday 21:29
feat(frontend): initialize frontend	Serhat Acar	Yesterday 21:28
chore : change frontend structure	Serhat Acar	Yesterday 21:00
feat : add frontend directory	Serhat Acar	Yesterday 20:53
docs: add project documentations to direct service documentations	Serhat Acar	Yesterday 17:33
docs(restaurant-service): update swagger open api config	Serhat Acar	Yesterday 15:53
chore(api-gateway): add cors config	Serhat Acar	Yesterday 15:52
docs(user-service & restaurant-service): update swagger api documentations	Serhat Acar	Yesterday 00:39
docs(user-service): add documentations	Serhat Acar	Yesterday 00:29
docs(restaurant-service): update documentations	Serhat Acar	Yesterday 00:14
docs(restaurant-service): add documentations	Serhat Acar	15.03.2024 23:51
fix(user-service): dump data	Serhat Acar	15.03.2024 23:04
fix(user-service & restaurant-service): fix mapper bug & double parse bug	Serhat Acar	15.03.2024 23:03
fix(user-service): fix validation bugs	Serhat Acar	15.03.2024 21:20
chore(user-service): configuration for test classes	Serhat Acar	15.03.2024 20:53
refactor(restaurant-service): move package for curlRestaurants	Serhat Acar	15.03.2024 19:32
docs(discovery-server): add README.md	Serhat Acar	15.03.2024 19:28
docs(api-gateway): add README.md	Serhat Acar	15.03.2024 19:11
chore: configuration add mvn files	Serhat Acar	15.03.2024 15:40
chore: configuration add .mvn directories	Serhat Acar	15.03.2024 15:36
chore: configuration (work successfully in dev mode)	Serhat Acar	15.03.2024 15:09
chore: configuration for docker	Serhat Acar	15.03.2024 02:58
chore: delete unused Dockerfile	Serhat Acar	15.03.2024 01:03
feat(user-service): add validations	Serhat Acar	14.03.2024 23:46
refactor(restaurant-service): enhance recommendation algorithm	Serhat Acar	14.03.2024 23:25
feat(logging-service): add repository and service	Serhat Acar	14.03.2024 21:30
feat(logging-service): add ErrorLog entity	Serhat Acar	14.03.2024 21:28
feat(logging-service): add logging-service module	Serhat Acar	14.03.2024 21:27
chore(user-service): update dumb data	Serhat Acar	14.03.2024 20:52
refactor(user-service): add constructor in RestResponse for test	Serhat Acar	14.03.2024 18:37

Spring Eureka Server (Discovery)

The screenshot displays the Spring Eureka Server web interface in a browser window. The page has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing environment details and system metrics.
- DS Replicas:** A section with a link to 'localhost'.
- Instances currently registered with Eureka:** A table listing registered applications and their status.
- General Info:** A table showing system resources like memory and CPU usage.

System Status Table:

Environment	test	Current time	2024-03-17T09:09:34 +0300
Data center	default	Uptime	01:50
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	12

DS Replicas: [localhost](#)

Instances currently registered with Eureka:

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - victus.hgw.local:api-gateway:8080
RESTAURANT-SERVICE	n/a (1)	(1)	UP (1) - victus.hgw.local:restaurant-service:8081
USER-SERVICE	n/a (1)	(1)	UP (1) - victus.hgw.local:user-service:8082


General Info:

Name	Value
total-avail-memory	94mb
num-of-cpus	12
current-memory-usage	45mb (47%)

localhost:8081/actuator/info

Docker Containers

Upgrade to Business Edition

 **portainer.io**
COMMUNITY EDITION

Home

local

Dashboard

App Templates

Stacks

Containers

Images

Networks

Volumes

Events

Host

Settings

Users

Environments

Registries

Authentication logs

Notifications

Settings

Containers







































































Container list

admin

Containers

Search...

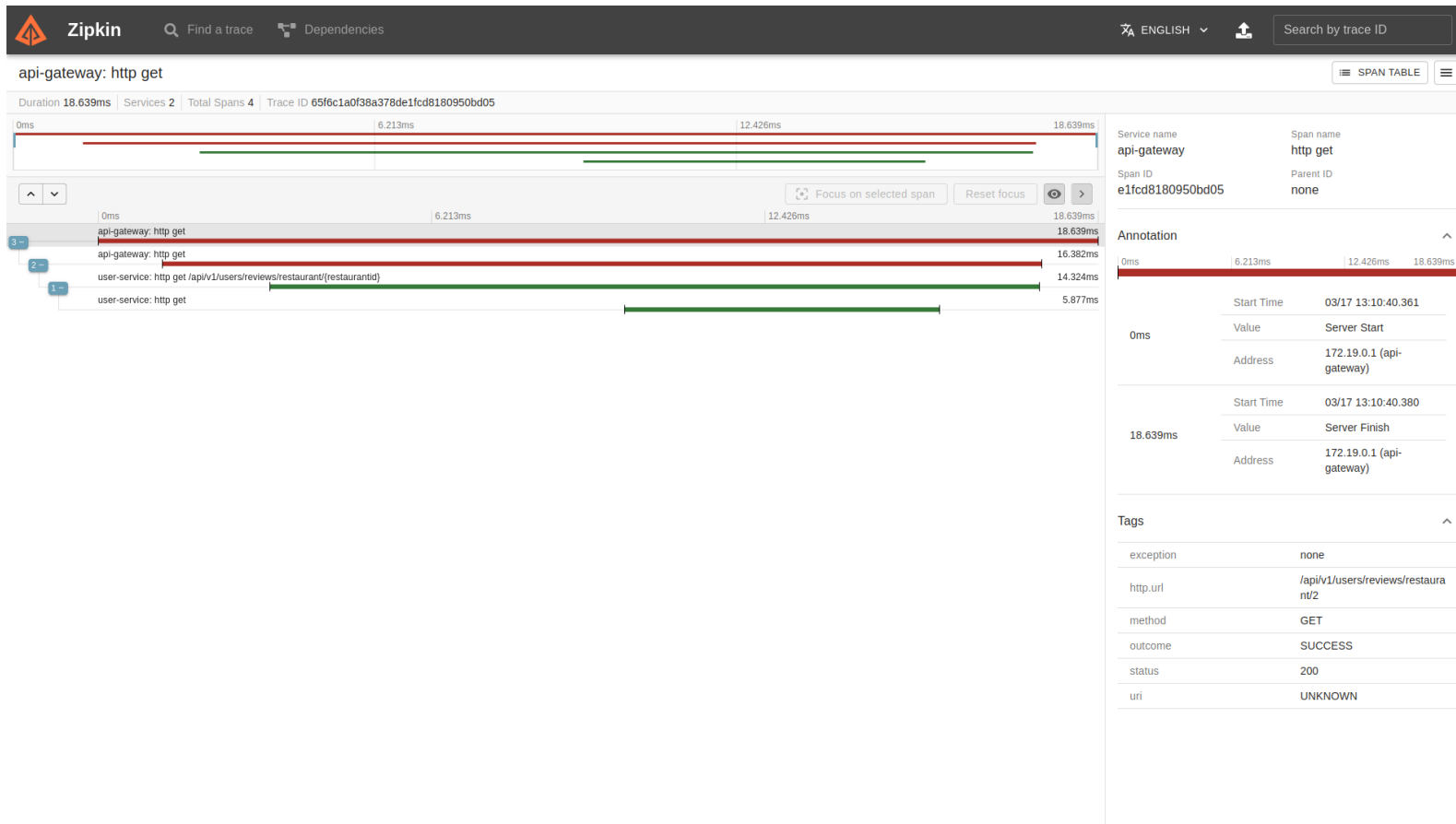
Start Stop Kill Restart Pause Resume Remove Add container

Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
api-gateway	running	      	dinexperienceapp	dinexperienceapp_api-gateway	2024-03-17 12:43:52	172.19.0.2	8080:8080	administrators
discovery-server	running	      	dinexperienceapp	dinexperienceapp_discovery-server	2024-03-17 12:43:52	172.19.0.7	8761:8761	administrators
kafka	running	      	dinexperienceapp	wurstmeister/kafka	2024-03-17 12:43:52	172.19.0.4	9092:9092	administrators
logging-service	running	      	dinexperienceapp	dinexperienceapp_logging-service	2024-03-17 12:43:52	172.19.0.9	8083:8083	administrators
mongo_db	running	      	dinexperienceapp	mongo	2024-03-17 12:51:53	172.19.0.12	27017:27017	administrators
portainer	running	      	-	portainer/portainer-ce	2024-03-16 00:52:50	172.17.0.2	9443:9000	administrators
postgres_db	running	      	dinexperienceapp	postgres	2024-03-17 12:43:52	172.19.0.5	5432:5432	administrators
restaurant-service	running	      	dinexperienceapp	dinexperienceapp_restaurant-service	2024-03-17 12:43:52	172.19.0.10	8081:8081	administrators
solr	running	      	dinexperienceapp	solr	2024-03-17 12:43:52	172.19.0.11	8983:8983	administrators
user-service	running	      	dinexperienceapp	dinexperienceapp_user-service	2024-03-17 12:43:52	172.19.0.3	8082:8082	administrators

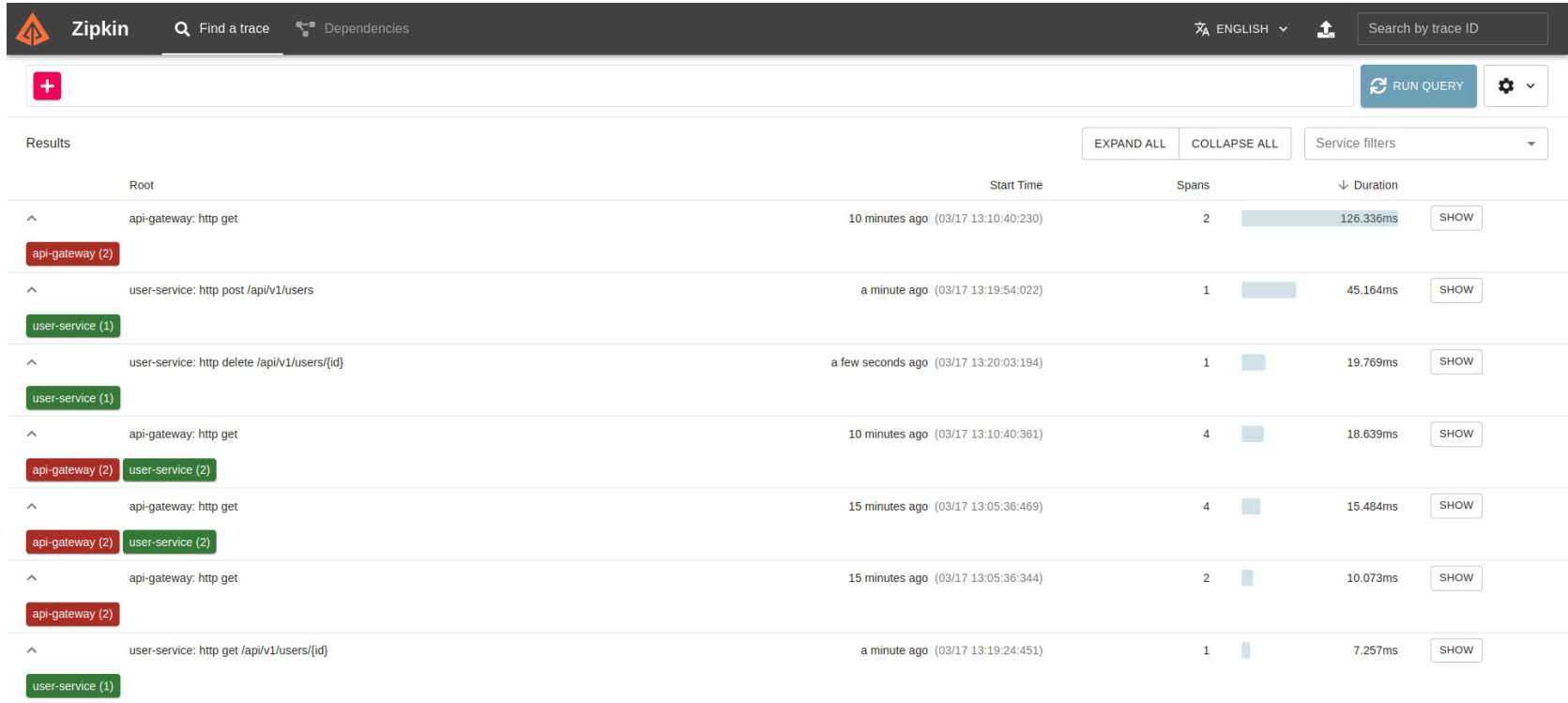
Items per page 10 1 2

portainer.io Community Edition 2.19.4

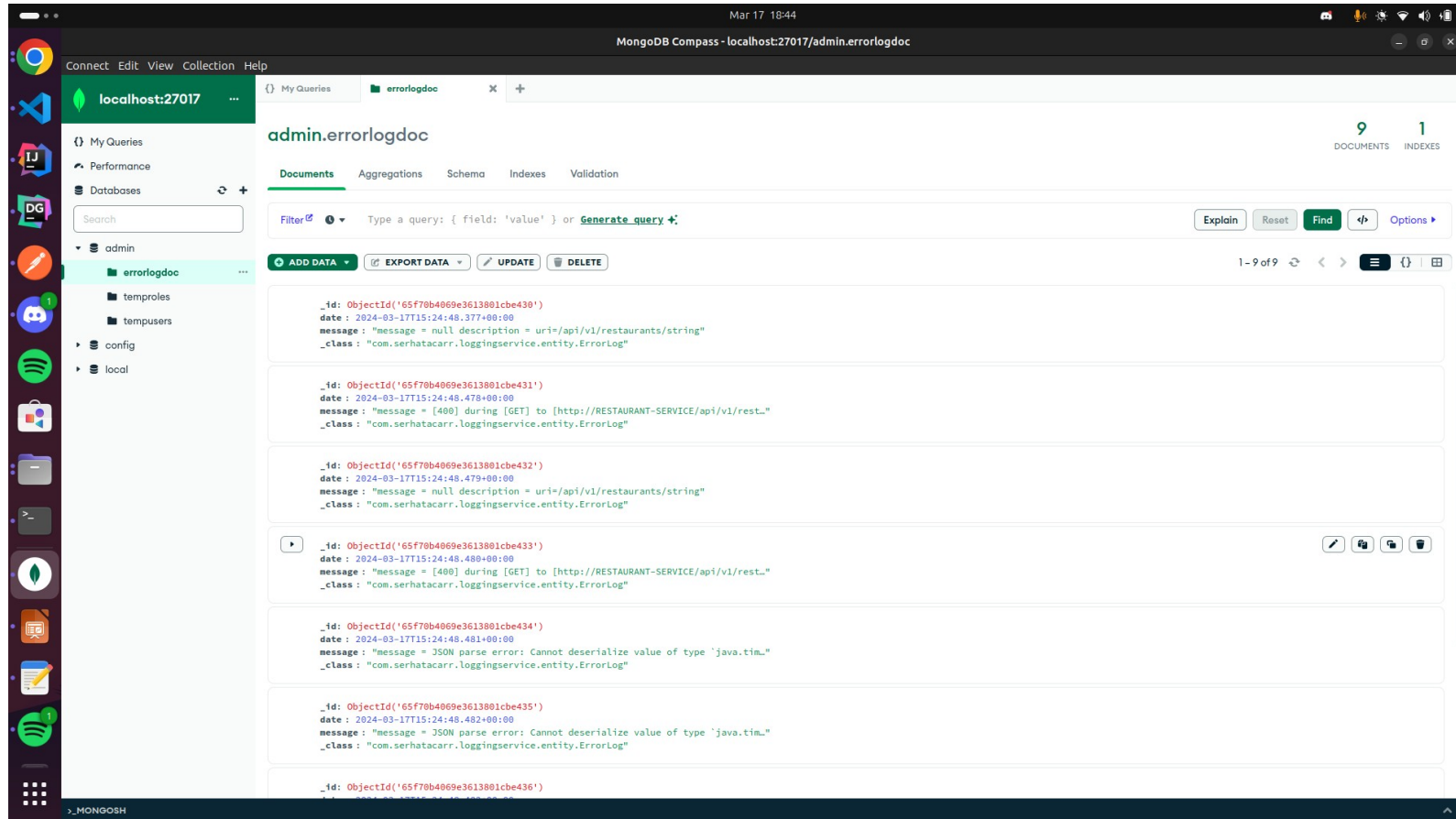
Zipkin Request Monitoring



Zipkin Time Monitoring



Logs in MongoDB with Kafka



The screenshot displays the MongoDB Compass interface for a local instance at localhost:27017. The left sidebar shows the database structure with 'admin' selected, containing collections like 'errorlogdoc', 'temproles', 'tempusers', 'config', and 'local'. The main panel shows the 'admin.errorlogdoc' collection with 9 documents and 1 index. The 'Documents' tab is active, displaying a list of log entries. Each entry is a JSON object with fields: '_id' (ObjectId), 'date' (timestamp), 'message' (log message), and '_class' (com.serhatacarr.loggingservice.entity.ErrorLog). The messages describe HTTP GET requests to the RESTAURANT-SERVICE API, some with null descriptions and others with JSON parse errors.

MongoDB Compass - localhost:27017/admin.errorlogdoc

My Queries | errorlogdoc

9 DOCUMENTS | 1 INDEXES

admin.errorlogdoc

Documents | Aggregations | Schema | Indexes | Validation

Filter | Type a query: { field: 'value' } or [Generate query](#)

EXPLAIN | RESET | FIND | OPTIONS

ADD DATA | EXPORT DATA | UPDATE | DELETE

1 - 9 of 9

Log entries (JSON format):

- `{ "_id": "ObjectId('65f70b4069e3613801cbe430')", "date": "2024-03-17T15:24:48.377+00:00", "message": "message = null description = uri=/api/v1/restaurants/string", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`
- `{ "_id": "ObjectId('65f70b4069e3613801cbe431')", "date": "2024-03-17T15:24:48.478+00:00", "message": "message = [400] during [GET] to [http://RESTAURANT-SERVICE/api/v1/rest-]", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`
- `{ "_id": "ObjectId('65f70b4069e3613801cbe432')", "date": "2024-03-17T15:24:48.479+00:00", "message": "message = null description = uri=/api/v1/restaurants/string", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`
- `{ "_id": "ObjectId('65f70b4069e3613801cbe433')", "date": "2024-03-17T15:24:48.480+00:00", "message": "message = [400] during [GET] to [http://RESTAURANT-SERVICE/api/v1/rest-]", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`
- `{ "_id": "ObjectId('65f70b4069e3613801cbe434')", "date": "2024-03-17T15:24:48.481+00:00", "message": "message = JSON parse error: Cannot deserialize value of type 'java.tim-", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`
- `{ "_id": "ObjectId('65f70b4069e3613801cbe435')", "date": "2024-03-17T15:24:48.482+00:00", "message": "message = JSON parse error: Cannot deserialize value of type 'java.tim-", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`
- `{ "_id": "ObjectId('65f70b4069e3613801cbe436')", "date": "2024-03-17T15:24:48.483+00:00", "message": "message = JSON parse error: Cannot deserialize value of type 'java.tim-", "_class": "com.serhatacarr.loggingservice.entity.ErrorLog" }`

> MONGOSH

User Service Unit Test

The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar shows the 'Run' button (a green play icon) next to the file name 'UserServiceImplTest.java'. The left sidebar shows the project structure with the 'test' directory selected. The main editor window shows the code for 'UserServiceImplTest.java', which includes package declarations, imports, and a public class definition with several private fields and annotations. The bottom panel shows the 'Run' output, indicating that all 5 tests passed successfully within 645 ms. The test results list includes 'testSaveUser()', 'testGetAllUsers()', 'testUpdateUser()', 'testDeleteUser()', and 'testGetUserById()'. The status bar at the bottom indicates the current file path and encoding settings.

```
package com.serhatacar.userservice.service;

import ...

public class UserServiceImplTest {

    5 usages
    @InjectMocks
    private UserServiceImpl userService;

    13 usages
    @Mock
    private UserEntityService userEntityService;

    11 usages
    @Mock
    private UserMapper userMapper;
```

Run UserServiceImplTest x

✓ UserServiceImplTest (com.serhatacar.userservice) 645 ms

✓ Tests passed: 5 of 5 tests - 645 ms

✓ testSaveUser() 638 ms

✓ testGetAllUsers() 3 ms

✓ testUpdateUser() 2 ms

✓ testDeleteUser() 2 ms

✓ testGetUserById() 2 ms

/home/ss/.jdk/openjdk-21.0.2/bin/java ...

Process finished with exit code 0

Loaded classes are up to date. Nothing to reload.

user-service > src > test > java > com > serhatacar > userservice > service > UserServiceImplTest 25:14 LF UTF-8 4 spaces

User Review Service Unit Test

The screenshot displays an IDE interface with the following components:

- Project Explorer:** Shows the project structure with a package hierarchy: `com.serhatacar.userservice` > `resources` > `test` > `java` > `com.serhatacar.userservice` > `common.base` > `controller` > `service`. The `UserReviewServiceImplTest` class is selected.
- Code Editor:** Displays the implementation of `UserReviewServiceImplTest`. The code includes imports, a package declaration, and a class definition with annotations like `@author Serhat Acar`, `@InjectMocks`, and `@Mock`.
- Run Console:** Shows the execution results of the tests. The output indicates that all tests passed successfully.
- Test Results:** A table summarizing the test results, showing the test name, duration, and status.

Test Name	Duration (ms)	Status
<code>testDeleteUserReview()</code>	686	Passed
<code>testGetUserReviewsByRestaurantId()</code>	22	Passed
<code>testGetUserReviewsById()</code>	2	Passed
<code>testGetUserReviewsByUserId()</code>	2	Passed
<code>testGetAllUserReviews()</code>	16	Passed
<code>testEditUserReview()</code>	2	Passed

Overall Test Summary: **Tests passed: 6 of 6 tests - 730 ms**

Process finished with exit code 0

Loaded classes are up to date. Nothing to reload.

User Controller Unit Test

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows the project structure with the following packages:
 - com.serhatacar.userservice
 - resources
 - test
 - java
 - com.serhatacar.userservice
 - common.base
 - controller
 - UserControllerIntegrationTest
 - UserControllerTest
 - service
 - UserServiceApplicationTests
- Editor:** Displays the code for `UserControllerTest.java`. The code is as follows:

```
@Test
void testGetAllUsers() {
    // Arrange
    UserDTO user1 = new UserDTO(1L, "name1", "surname1", LocalDate.now(), "email1", Gender.MALE, Status.ACTIVE, 4.0, 5.0);
    UserDTO user2 = new UserDTO(2L, "name2", "surname2", LocalDate.now(), "email2", Gender.FEMALE, Status.ACTIVE, 4.0, 4.0);
    List<UserDTO> users = Arrays.asList(user1, user2);

    when(userService.getAllUsers()).thenReturn(users);

    // Act
    ResponseEntity<RestResponse<List<UserDTO>>> response = userController.getAllUsers();

    // Assert
    assertEquals("expected: 200, response.getStatusCodeValue()", response.getStatusCodeValue(), 200);
    assertEquals(users.size(), response.getBody().getData().size());
}
```
- Run Panel:** Shows the test results for `UserControllerTest`. The test passed, and the output is as follows:

```
Tests passed: 5 of 5 tests - 692 ms
Process finished with exit code 0
```
- Bottom Panel:** Shows the file path: `user-service > src > test > java > com > serhatacar > userservice > controller > UserControllerTest`. The status bar indicates the file is encoded in UTF-8 with 4 spaces.

User Controller Integration Test

The screenshot displays an IDE interface with the following components:

- Project Explorer:** Shows the project structure for `dineXperienceApp`. The `test` directory is expanded, showing the `java` package containing `com.serhatacar.userservice`, `common.base`, `controller`, and `service`. The `UserControllerIntegrationTest` file is selected.
- Code Editor:** Displays the source code for `UserControllerIntegrationTest.java`. The code includes a `private Long userid = 10L;` field, a `@BeforeEach` `setUp()` method that initializes `MockMvc` and `ObjectMapper`, and a `@Test` `shouldGetUsers()` method that performs a GET request to `/api/v1/users` and asserts the status is OK.
- Run Window:** Shows the execution results of the `UserControllerIntegrationTest`. It indicates that 5 tests passed in 535 ms. The test results are as follows:

Test Method	Duration
<code>shouldGetUserById()</code>	405 ms
<code>shouldUpdateUser()</code>	44 ms
<code>shouldDeleteUser()</code>	16 ms
<code>shouldGetUsers()</code>	51 ms
<code>shouldSaveUser()</code>	19 ms
- Output Console:** Displays the standard output of the test run, including log messages from the Spring Boot application and the IDE's restart listener.

Frontend Users

Restaurants

Users

<input type="checkbox"/>	ID	First name	Last name	Email	Status	Actions
<input type="checkbox"/>	1	John	D****e	j*****@example.com	ACTIVE	Delete
<input type="checkbox"/>	2	Jane	S****h	j*****@example.com	ACTIVE	Delete
<input type="checkbox"/>	3	Bob	J****s	b*****@example.com	ACTIVE	Delete
<input type="checkbox"/>	4	Alice	M****r	a*****@example.com	INACTIVE	Delete
<input type="checkbox"/>	5	David	W****n	d*****@example.com	ACTIVE	Delete

Rows per page: 100 1-10 of 10

Name:

Surname:

Birth Date:

mm/dd/yyyy

Email:

Gender:

Male

Status:

Active

Submit

Frontend Restaurants

