Creature Adoption System

Project Milestone 1

Web Services and Distributed Computing (420-N45-LA)

Date: April 17, 2025



Student Name: Christian James Lee

Student ID: 2232469

Teacher: Christine Gerard

C4 Level 1 Diagram Explanation:

A diagram of a customer

AI-generated content may be incorrect.

This is the C4 Level 1 Diagram for my service called Creature Adoption. It is very simple as it only shows one interaction between the Customer looking to adopt creatures and the Creature Adoption System. This simplicity helps narrow down the system’s purpose for the <<person>> who is going to use it.

It allows the customer to have their own profile with first name, last name, email, etc. within the creature adoption system, to see what creatures are available or unavailable, and the specific training programs there are available for them creatures themselves.

C4 Level 2 Diagram Explanation:

A diagram of a customer

AI-generated content may be incorrect.

A diagram of a software application

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

This is the C4 Level 2 Diagram for my service called Creature Adoption. It starts off like the Level 1 diagram where it’s a customer who wants to interact with the adoption system, but this time it is more specific in the backend behavior of the system.

The first thing that is interacted with is the Web Application/Single-Page Application with JavaScript, React. Although this is not implemented in the system as milestone 1 does not require it, in a real-life environment it should be there because real customers would not like to interact with just an API.

Anyways, through the web, it will make a call to the API Gateway through port 8080 with a JSON format. The API Gateway acts like a middleman for all the other microservices where you will only need to send the request through the API Gateway port instead of a separate port for every individual microservice. This is why it is connected to every microservice through making API calls for them.

The aggregate microservice is set up this way because it will need to also call each individual microservice, whereas if you call the non-aggregate microservices by themselves, it will be direct and bypass the adoption-service. This will use Mongo-DB but that’s for milestone 2.

Now, all the non-aggregate microservices are all set up the same as each other where they are just connected with their own database as microservices do. The only difference is how creature-service uses PostgreSQL whereas the other microservices use MySQL.

Step 4 changelog:

I did not use generative ai for migrating my subdomains into low-level microservices because it was too much effort, and it was easier to just directly copy and paste the subdomains from the monolith into low-level microservices as their functionality stayed basically the same.

From what I’ve tried, genAI is good for questions on specific topics like how to use specific Java imports properly but is very bad at generating a lot of code all at once because of how easy it is for it to miss multiple important steps or just the mistakes it makes pretty often since it doesn’t have as much context on the project as a human working on it would.

As for my change log, for each low-level microservice, the first step is to change all the imports to be the right imports, then is to add the right build.gradle dependencies so things like Lombok could work. The next step is to write the application.yml. For my creatures application.yml, I changed the mysql to postgresql as for project requirements. I then also changed the ports for both the local hosting, and docker for each microservice to match what is configured for the api-gateway. Finally, my data was split into many different data files for each of the microservices since every microservice has its own database.

My DDD was not changed at all because everything worked almost the same way, perhaps it will be changed in milestone 2 if accommodations need to be made for the Adoption microservice.

JACOCO COVERAGE:

creatures-service:

<file:///C:/Users/oshaw/Desktop/cegep/webServices/creatureAdoption-ws/creatures-service/build/reports/jacoco/test/html/index.html>

A screenshot of a computer

AI-generated content may be incorrect.

customers-service

<file:///C:/Users/oshaw/Desktop/cegep/webServices/creatureAdoption-ws/customers-service/build/reports/jacoco/test/html/index.html>

A screenshot of a computer

AI-generated content may be incorrect.

trainings-service

<file:///C:/Users/oshaw/Desktop/cegep/webServices/creatureAdoption-ws/trainings-service/build/reports/jacoco/test/html/index.html>

A screenshot of a computer

AI-generated content may be incorrect.

Positive and negative path tests:

Creatures Microservice

Positive path (whenGetCreatureByValidId\_thenReturnCreature):

A black screen with white text

AI-generated content may be incorrect.

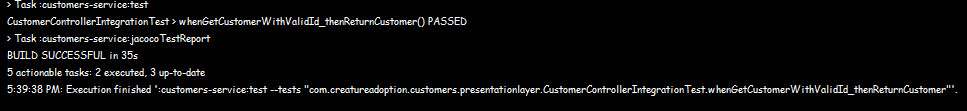
Negative path (whenGetCreatureWithInvalidIdFormat\_thenReturnUnprocessable):

A black background with white text

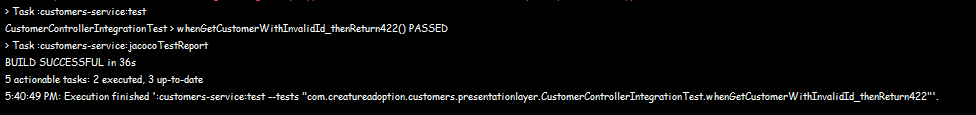
AI-generated content may be incorrect.

Customers Microservice

Positive path (whenGetCustomerWithValidId\_thenReturnCustomer):

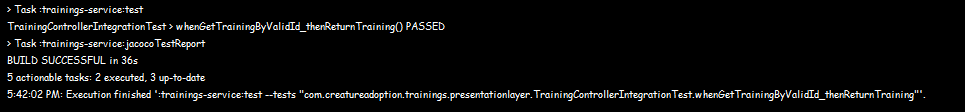


Negative path (whenGetCustomerWithInvalidId\_thenReturn422)

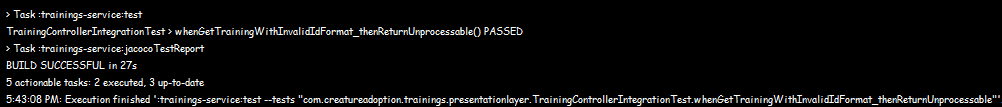


Trainings Microservice

Positive path (whenGetTrainingByValidId\_thenReturnTraining):



Negative path (whenGetTrainingWithInvalidIdFormat\_thenReturnUnprocessable):



CURL COMMANDS:

---------------------------------------------------------------------------------------------------------

trainings-controller

GET 1:

curl -X 'GET' \

'http://localhost:8080/api/v1/trainings/a9b8c7d6-e5f4-g3h2-i1j0-k9l8m7n6o5p4' \

-H 'accept: application/json'

PUT 1:

curl -X 'PUT' \

'http://localhost:8080/api/v1/trainings/a9b8c7d6-e5f4-g3h2-i1j0-k9l8m7n6o5p4' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"name": "string",

"description": "string",

"difficulty": "BEGINNER",

"duration": 0,

"status": "ACTIVE",

"category": "ATTACK",

"price": 0,

"location": "string"

}'

DELETE 1:

curl -X 'DELETE' \

'http://localhost:8080/api/v1/trainings/a9b8c7d6-e5f4-g3h2-i1j0-k9l8m7n6o5p4' \

-H 'accept: \*/\*'

GET ALL:

curl -X 'GET' \

'http://localhost:8080/api/v1/trainings' \

-H 'accept: application/json'

POST 1:

curl -X 'POST' \

'http://localhost:8080/api/v1/trainings' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"name": "string",

"description": "string",

"difficulty": "BEGINNER",

"duration": 0,

"status": "ACTIVE",

"category": "ATTACK",

"price": 0,

"location": "string"

}'

---------------------------------------------------------------------------------------------------------

customers-controller

GET 1:

curl -X 'GET' \

'http://localhost:8080/api/v1/customers/6f8d2e53-9b4c-48a7-91fe-c508dde7817a' \

-H 'accept: application/json'

PUT 1:

curl -X 'PUT' \

'http://localhost:8080/api/v1/customers/6f8d2e53-9b4c-48a7-91fe-c508dde7817a' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"firstName": "string",

"lastName": "string",

"emailAddress": "string",

"contactMethodPreference": "EMAIL",

"streetAddress": "string",

"city": "string",

"province": "string",

"country": "string",

"postalCode": "string",

"phoneNumbers": [

{

"type": "HOME",

"number": "string"

}

]

}'

DELETE 1:

curl -X 'DELETE' \

'http://localhost:8080/api/v1/customers/6f8d2e53-9b4c-48a7-91fe-c508dde7817a' \

-H 'accept: \*/\*'

GET ALL:

curl -X 'GET' \

'http://localhost:8080/api/v1/customers' \

-H 'accept: application/json'

POST 1:

curl -X 'POST' \

'http://localhost:8080/api/v1/customers' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"firstName": "string",

"lastName": "string",

"emailAddress": "string",

"contactMethodPreference": "EMAIL",

"streetAddress": "string",

"city": "string",

"province": "string",

"country": "string",

"postalCode": "string",

"phoneNumbers": [

{

"type": "HOME",

"number": "string"

}

]

}'

---------------------------------------------------------------------------------------------------------

creatures-controller

GET 1:

curl -X 'GET' \

'http://localhost:8080/api/v1/creatures/8a4b5c6d-7e8f-9a0b-1c2d-3e4f5a6b7c8d' \

-H 'accept: application/json'

PUT 1:

curl -X 'PUT' \

'http://localhost:8080/api/v1/creatures/8a4b5c6d-7e8f-9a0b-1c2d-3e4f5a6b7c8d' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"name": "string",

"species": "string",

"type": "NORMAL",

"rarity": "COMMON",

"level": 0,

"age": 0,

"health": 0,

"experience": 0,

"status": "AVAILABLE",

"strength": 0,

"intelligence": 0,

"agility": 0,

"temperament": "DOCILE"

}'

DELETE 1:

curl -X 'DELETE' \

'http://localhost:8080/api/v1/creatures/8a4b5c6d-7e8f-9a0b-1c2d-3e4f5a6b7c8d' \

-H 'accept: \*/\*'

GET ALL:

curl -X 'GET' \

'http://localhost:8080/api/v1/creatures' \

-H 'accept: application/json'

POST 1:

curl -X 'POST' \

'http://localhost:8080/api/v1/creatures' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"name": "string",

"species": "string",

"type": "NORMAL",

"rarity": "COMMON",

"level": 0,

"age": 0,

"health": 0,

"experience": 0,

"status": "AVAILABLE",

"strength": 0,

"intelligence": 0,

"agility": 0,

"temperament": "DOCILE"

}'