Creature Adoption System

Project Milestone 2

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

Date: May 13, 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.

All the other microservices are mentioned too, adoptions, trainings, and creatures.

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 uses MongoDB, a noSQL database. Inside every adoption, it includes a customerId, a trainingId, and a creatureId. Whenever an adoption status changes, it will also change the creature status to an appropriate status. An example of this happening is with the newly implemented patch commands for the adoptions microservice: ADOPTION STATUS: APPROVED = CREATURE STATUS = RESERVED, ADOPTION STATUS: PENDING: CREATURE STATUS = ADOPTION\_PENDING, ADOPTION STATUS: COMPLETED: CREATURE STATUS: ADOPTED, ETC. I think you get the picture however.

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.

DDD AND why my code matches with the diagrams Explanation: A screenshot of a computer screen

AI-generated content may be incorrect.

I’m sorry if this is hard to read but I really don’t have that much time or energy left for this and am sort of panicking in this submission document so if you need a clearer view of the DDD, it’s in submissions.

My DDD was not changed at all because everything worked almost the same way. I did not really expect to change it because the logic was supposed to just mirror lab1 anyways.

This is the invariant note in my Adoption Management Subdomain:

[Invariant: [When a creature is adopted (adoption.status is set to PENDING), its status in the creature must be set to ADOPTION\_PENDING] Any change in adoptionstatus will then result in a change in creaturestatus too]

A screenshot of a computer program

AI-generated content may be incorrect.

This code segment can be found in com.creatureadoption.adoptions.dataaccesslayer

Within the same file there is also code for the 2 adoptions limit, the max is passed through a param.

A screenshot of a computer program

AI-generated content may be incorrect.

As seen in the response model, I do in fact use the enums listed in the DDD alongside the adoption identifier (adoptionid):

A screenshot of a computer

AI-generated content may be incorrect.

(This will get repetitive as you requested that I prove that I’ve followed my DDD)

Now for the creature response model, the enums are followed:

A screenshot of a computer program

AI-generated content may be incorrect.

Now for Customer response model, still followed generally but this comes directly from car dealership:

A screen shot of a computer

AI-generated content may be incorrect.

The way adoption accesses the other microservices is through the api-gateway port 8080:

A screenshot of a computer

AI-generated content may be incorrect.

Like the other microservices, it is only internally exposed to a different port than 8080, but can be externally accessed through that because of such api gateway. In the docker configuration, the external facing ports have been commented out:

A screen shot of a computer

AI-generated content may be incorrect.

If you need EVEN MORE code proof, then here is my getAdoptions implementation in adoptionsserviceimpl, as you can see, it gets all the adoptions from the repository (mongo) and puts them in a list, it then checks for query params and does a bit of filtering to that same adoptions list and then once done, it will put it through the responseModel, (as seen above) and it will have all the correct fields.

A screenshot of a computer program

AI-generated content may be incorrect.

The repository uses MongoRepository instead of JPA, but same principle for this! So this is how the data is retrieved for adoptions!!!!

A screen shot of a computer

AI-generated content may be incorrect.

I forgot to include this but to wrap it all off, here is my gets for my customer Controller:

A screen shot of a computer program

AI-generated content may be incorrect.

Finally, just for good measure, here is the api-gateway application.yml docker section, as you can see, it is being ran on 8080, connecting to the other hosts internally of the microservices.

A screen shot of a computer

AI-generated content may be incorrect.

And that wraps this section of the code! I will continue this next morning.

Request and response models in adoptions service

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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.

adoptions-service:

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

A screenshot of a computer

AI-generated content may be incorrect.

api-gateway:

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

A screenshot of a computer

AI-generated content may be incorrect.

./gradlew clean build test results (VERY LONG BEWARE) :

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot 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.

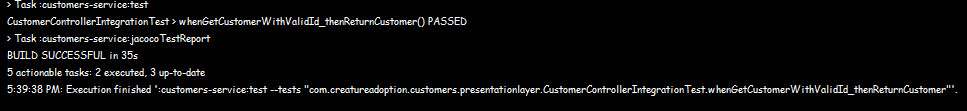
Negative path (whenGetCreatureWithInvalidIdFormat\_thenReturnUnprocessable):

A black background with white text

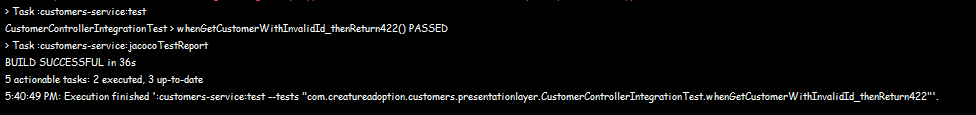
AI-generated content may be incorrect.

Customers Microservice

Positive path (whenGetCustomerWithValidId\_thenReturnCustomer):

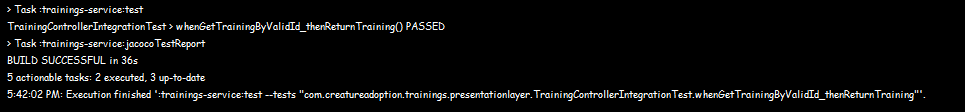


Negative path (whenGetCustomerWithInvalidId\_thenReturn422)

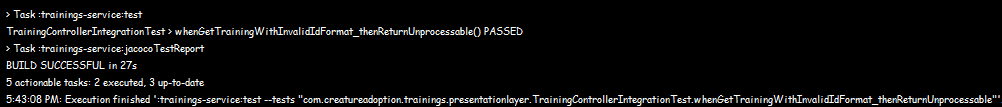


Trainings Microservice

Positive path (whenGetTrainingByValidId\_thenReturnTraining):



Negative path (whenGetTrainingWithInvalidIdFormat\_thenReturnUnprocessable):



Adoptions Microservice (NEW!!!! 😍)

Positive path (getAdoptionByAdoptionId\_WithValidId\_ShouldReturnAdoption):

A screenshot of a computer program

AI-generated content may be incorrect.

Negative path (getAdoptionByAdoptionId\_WithNonExistentId\_ShouldReturn404):

A screen shot of a computer program

AI-generated content may be incorrect.

Test table (INV means invalid so I might’ve had a test that fit that category but idk I got to 90% in each at least):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Type | API-Gateway | Aggregator | LLMS1 | LLMS1 | LLMS3 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Repository Integration | INV | INV | 9 | 3 | 7 | 2 | 12 | 1 | 10 | 1 |
| Controller Integration | 23 | 28 | 8 | 16 | 10 | 12 | 6 | 17 | 10 | 17 |
| Controller Unit | 21 | 12 | 8 | 15 | INV | INV | INV | INV | INV | INV |
| Service Unit | INV | INV | 13 | 12 | INV | INV | INV | INV | INV | INV |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | adoptions | creatures | customers | trainings |

(some of the test classes might’ve been named differently from what you wanted exactly, but I’ve spent a few hours matching them the best I can to whatever category they fit the most to my abilities but hopefully my entire project warrants a passing grade at least.)

You might also notice that I have included way more files than requested for submission, I only had those included just in case anything in my word document broke