Assessment Cover Page

Module Title:	Data Storage Solutions (5 ECTS) Commercial Solutions Design (10 ECTS)
Assessment Title:	CA1
Lecturer Name:	Muhammad Iqbal David McQuaid
Student Full Name:	Student NAME: Bekezhan Abdykarimov STUDENT NUMBER: 2020297 Student NAME: Daniel Bezerra Martellini STUDENT NUMBER: 2020356 Student NAME: Jefferson De Oliveira Lima STUDENT NUMBER: 2020373
Student Number:	2020297, 2020356, 2020373
Assessment Due Date:	07/05/23
Date of Submission:	07/05/23

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

PROJECT LINK

INTRODUCTION

Nowadays, the Internet has become an indispensable part of our life. According to the latest estimates, 328.77 million terabytes of data are created every day. Many people use it on a daily basis, without realizing how crucial the Internet to be precise the web apps and BigData that go along with them have become, how deeply they were integrated in the Economy field, Health Care, Logistics, Education, basically in all main parts of our modern society. If it all fails, for let's say an hour, our world will face a huge chain of collapses, billions of money will be lost, and millions of people's lives will be affected. That's why it is important to create safe, fast, modern and manageable web applications that will simplify our lives and allow progress to move forward. To learn how to build those applications that can handle the storing and manipulation of big amounts of data, we decided to use publicly available "Spotify tracks" dataset, to build a Music Management web application.

This report provides an analysis and a comprehensive overview of a Management Music System application planned and built by our group, focusing on the code implementation, including HTML, CSS, and Java components using Spring boot framework. The application is a web-based platform that provides authentication functionality that allows users to access the application, manipulate the dataset and perform full CRUD functionality on it. The report will also cover the source datasets, the objective of the data storage framework, possible database solutions, the architecture of the data storage, and the difficulties that our group faced during the development. Finally, the report will provide a detailed description and rationalization of all the functionality for the Spring Application and each group member's reflection and contribution. We separated the project tasks so each of us were responsible for some particular part of the project and we wrote the report the way that each of us will write about the part that he was responsible for.

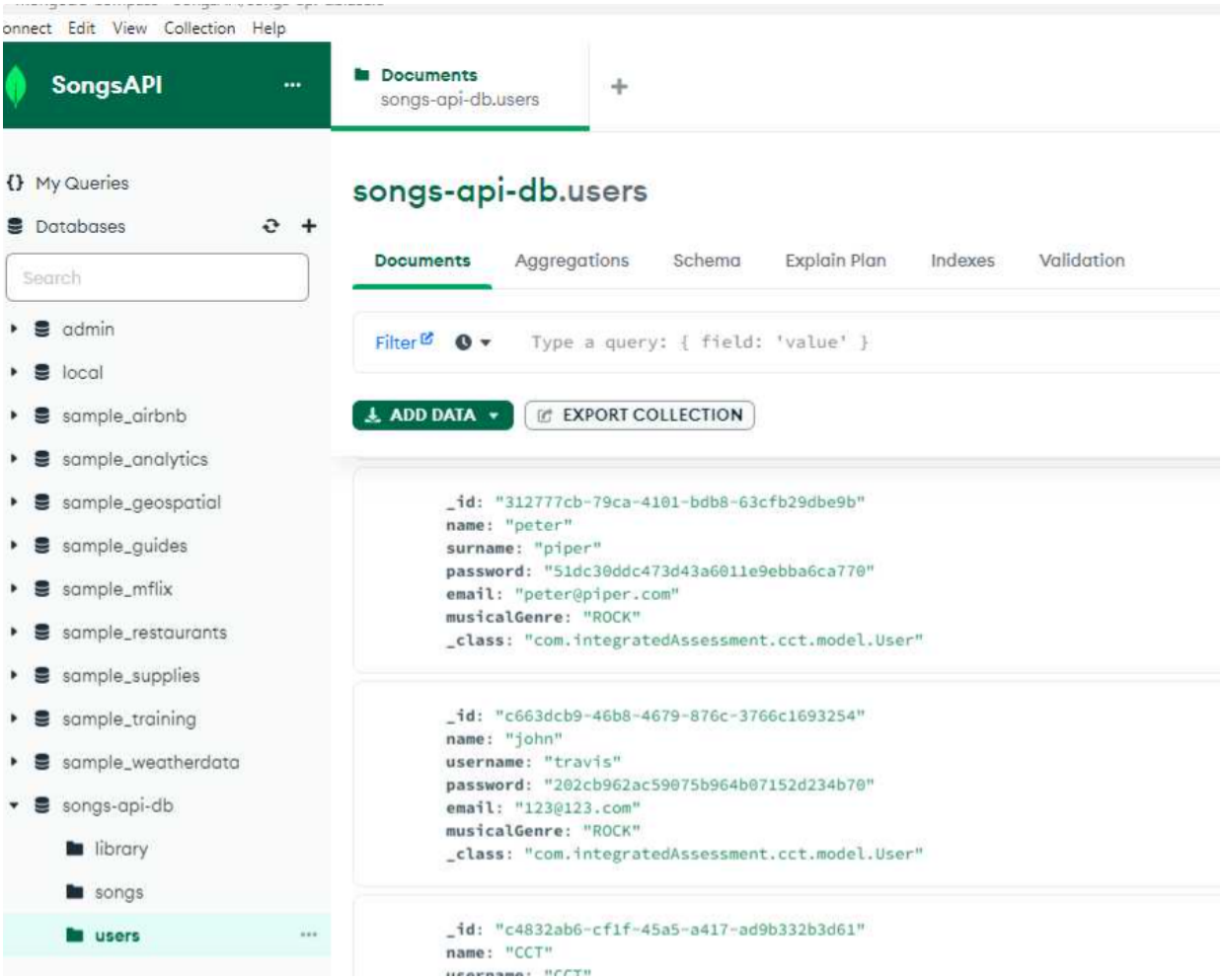
DATABASE AND DESIGN

The chapter deals with the database and ITs Design

For this project we decided to use MongoDb Atlas which is a global cloud database service built and run by the team of Mongo Database, which simplifies working on the application development.

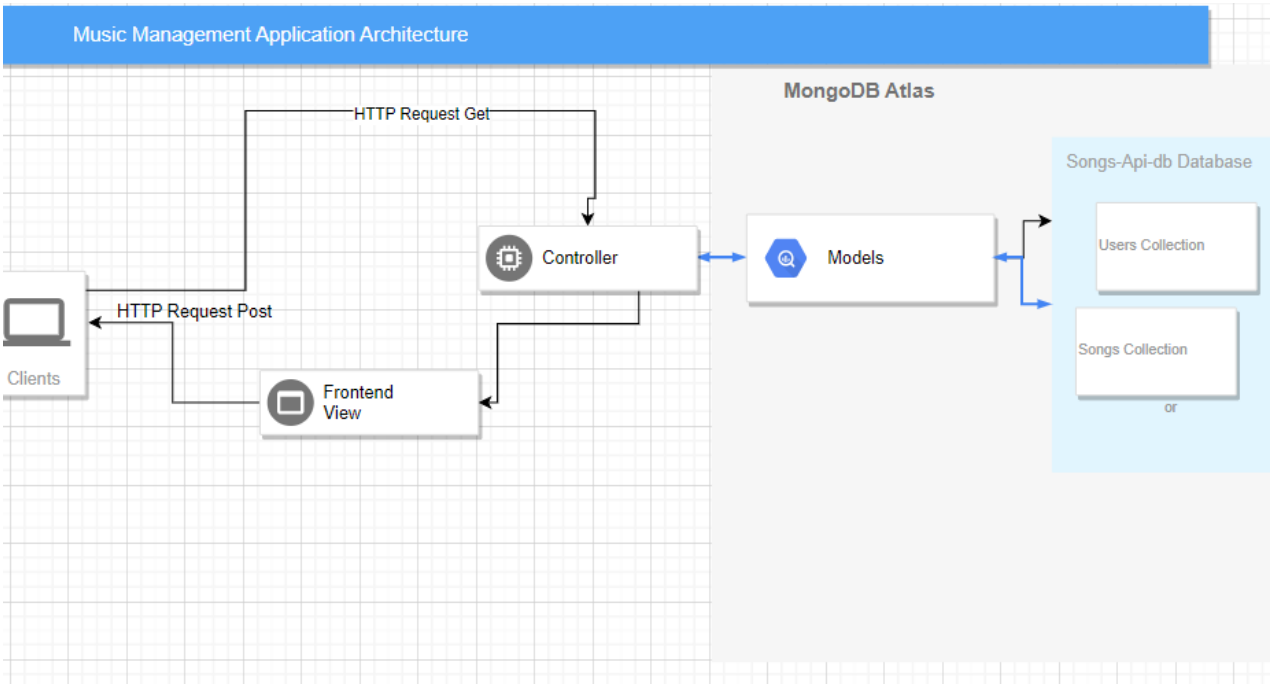
The users collection was used to store and save all the data from the user that he will enter in the userLogin.html page. Each user has an unique id, as a primary key.

Figure 1. Songs-api-db database with collections:



The users collection was used to store and save all the data from the user that he will enter in the userLogin.html page. Each user has an unique id, as a primary key.

Figure 2. Project architecture Diagram



DESCRIPTION OF THE SOURCE DATASET

DANIEL BEZERRA MARTELLINI (2020356):

We chose to use this dataset found on Kaggle which contains data of 114000 Spotify Tracks and has 21 different columns with different information such as Spotify Track id, track Popularity, Artist, Album, etc. The dataset source is Spotify and it was obtained and cleaned using Spotify's WEB API and Python and published the Open Data Commons Open Database License and it was published in December 2021 on Keggle by user maharshipandya.

https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset

In my group I was the one to do the Hadoop part of the project, I, Daniel Bezerra Martellini 2020356, started by creating a new virtual machine and proceeded to install the hadoop file system on it, inside hadoop I created a directory named user1 where I uploaded my dataset which was in csv format.

```
hduser@integratedCA:~$ hadoop version
Hadoop 3.2.4
Source code repository Unknown -r 7e5d9983b388e372fe640f21f048f2f2ae6e9eba
Compiled by ubuntu on 2022-07-12T11:58Z
Compiled with protoc 2.5.0
From source with checksum ee031c16fe785bbb35252c749418712
This command was run using /usr/local/hadoop-3.2.4/share/hadoop/common/hadoop-common
-3.2.4.jar
hduser@integratedCA:~$
```

1.1 Hadoop Installed

```
hduser@integratedCA:~$ hadoop fs -put ./dataset.csv /user1
```

1.2 Sending File from Ubuntu storage to be stored in Hadoop

After having the dataset in hadoop i proceeded to do the map-reduce tasks and to upload my database to mongodb so after we could use it in our spring boot application. In order to do the map reduce I had to adapt code that we had in a previous class that would map a txt file and then reduce it, but inside of doing it by word I did by column, so then I was able to choose from which column I would like to extract information, I also modified the code from the reducer later so I could only see items that had 10 or more appearances to find out what was more popular.

Open *mymapper.py ~/Desktop/Hadoop Save

*mymapper.py myreducer.py reducer.py

```
#!/usr/bin/env python3
import sys

# which column want to map and reduce
column_index = 21
delimiter = ","


for line in sys.stdin:

    # Split the line
    data = line.strip().split(delimiter)

    # If the column has data
    if len(data) > column_index:


        value = data[column_index].strip()


        print(f"{value}\t1")
```

Open myreducer.py ~/Desktop/Hadoop Save

mymapper.py myreducer.py reducer.py

```
#!/usr/bin/env python3

import sys

current_value = None
current_count = 0

for line in sys.stdin:
    line = line.strip()
    if "\t" in line:
        value, count = line.split("\t")
    elif "," in line:
        value, count = line.split(",")
    else:
        continue

    if current_value != value:
        if current_value is not None:
            print("%s\t%d" % (current_value, current_count))
        current_value = value
        current_count = 0

    current_count += int(count)

if current_value is not None:
    print("%s\t%d" % (current_value, current_count))

```

1.3 My Mapper & 1.4 My Reducer

The mapper and reducer are both small pieces of code written in python built from different exemples I found online and the one used in class to count words, it was a bit tricky because of the precise indentation and also because I don't know much python, I faced an issue when trying to run mapreduce tasks when trying to use the original dataset.csv because the original dataset had a row with the title (header) for each column but also had one item less than the rest of the rows causing the file to be misaligned creating problems when running my map reduce function, after removing the title of the columns on the datasetupdate.csv file I had no more problems.

```
hduser@integratedCA:~/Desktop/Hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/tools
/lib/hadoop-streaming-3.2.4.jar  -mapper ./mymapper.py  -reducer ./myreducer.py
-input /user1/datasetupdated.csv -output /out111
2023-05-06 01:16:28,458 INFO impl.MetricsConfig: Loaded properties from hadoop-m
etrics2.properties
2023-05-06 01:16:28,543 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot p
eriod at 10 second(s).
```

1.5 Executing MapReduce

```
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=114000
                Map output records=114000
                Map output bytes=2224227
                Map output materialized bytes=2452403
                Input split bytes=98
                Combine input records=0
                Combine output records=0
                Reduce input groups=31438
                Reduce shuffle bytes=2452403
                Reduce input records=114000
                Reduce output records=31437
                Spilled Records=228000
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=75
                Total committed heap usage (bytes)=557842432
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=20139901
        File Output Format Counters
                Bytes Written=774055
2023-05-06 01:16:33,099 INFO streaming.StreamJob: Output directory: /out111
hduser@integratedCA:~/Desktop/Hadoop$
```

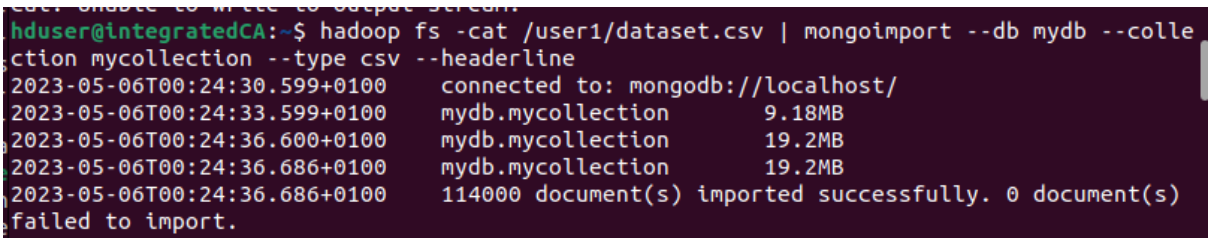
1.6 Successful MapReduce

```
hduser@integratedCA:~/Desktop/Hadoop$ hadoop fs -cat /out111/part-00000
```

1.7 Outputting MapReduce Result

1.8 Part of MapReduce output

After being able to run map reduce on all different columns of my dataset I proceeded to upload my csv file, which this time was the original one with the header, only one line was enough to generate a database in mongodb with 11400 Documents.



1.9 Importing CSV file from Hadoop to Mongodb collection

1.10 Checking if the collection is in the right format

I tested the database with a few Queries to make sure that everything was right, and it had all fields correctly filled and also the correct number of items so after checking that the database was correctly populated I started testing the Workloads with YCSB.

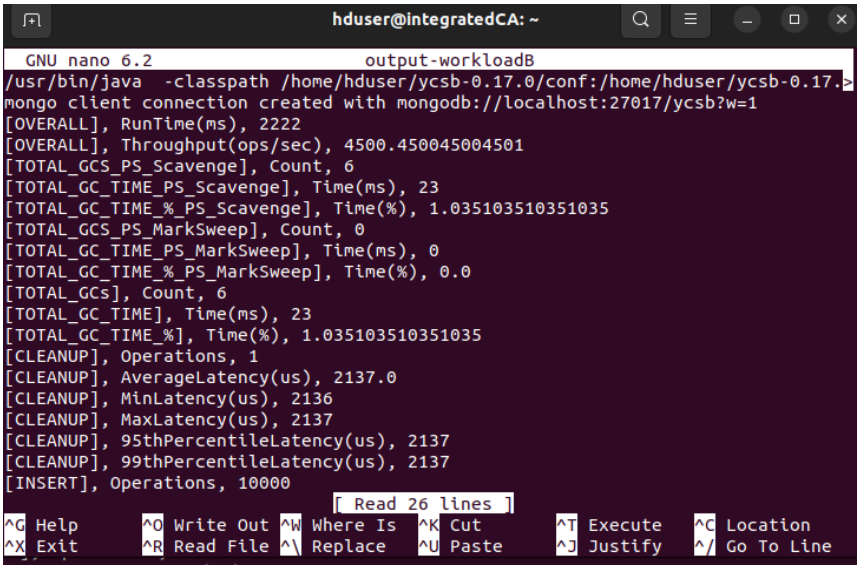
First I tested the preload workloada with the default values with noSQL database Mongodb and then took a screenshot of the results, after I tested the preloaded workloadb with 10,000 rows instead of 1000 and then tested the same workload with 100,000 rows and saved them both on txt. files in order to compare results.

```
Command line: -load -db site.ycsb.db.MongoDbClient -s -P workloads/workloada
YCSB Client 0.17.0

Loading workload...
Starting test.
2023-05-06 01:55:48:078 0 sec: 0 operations; est completion in 0 second
mongo client connection created with mongodb://localhost:27017/ycsb?w=1
DBWrapper: report latency for each error is false and specific error codes to track for latency
are: []
2023-05-06 01:55:50:126 2 sec: 10000 operations; 4816.96 current ops/sec; [CLEANUP: Count=1, Max
=2211, Min=2210, Avg=2211, 90=2211, 99=2211, 99.9=2211, 99.99=2211] [INSERT: Count=10000, Max=10
4063, Min=85, Avg=166.88, 90=255, 99=608, 99.9=4071, 99.99=14135]
[OVERALL], RunTime(ms), 2077
[OVERALL], Throughput(ops/sec), 4814.6364949446315
[TOTAL_GCS_PS_Scavenge], Count, 5
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 20
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 0.9629272989889264
[TOTAL_GCS_PS_MarkSweep], Count, 0
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
[TOTAL_GCs], Count, 5
[TOTAL_GC_TIME], Time(ms), 20
[TOTAL_GC_TIME_%], Time(%), 0.9629272989889264
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 2211.0
[CLEANUP], MinLatency(us), 2210
[CLEANUP], MaxLatency(us), 2211
[CLEANUP], 95thPercentileLatency(us), 2211
[CLEANUP], 99thPercentileLatency(us), 2211
[INSERT], Operations, 10000
[INSERT], AverageLatency(us), 166.8767
[INSERT], MinLatency(us), 85
[INSERT], MaxLatency(us), 104063
[INSERT], 95thPercentileLatency(us), 338
[INSERT], 99thPercentileLatency(us), 608
[INSERT], Return=OK, 10000
hduser@integratedCA:~/ycsb-0.17.0$
```

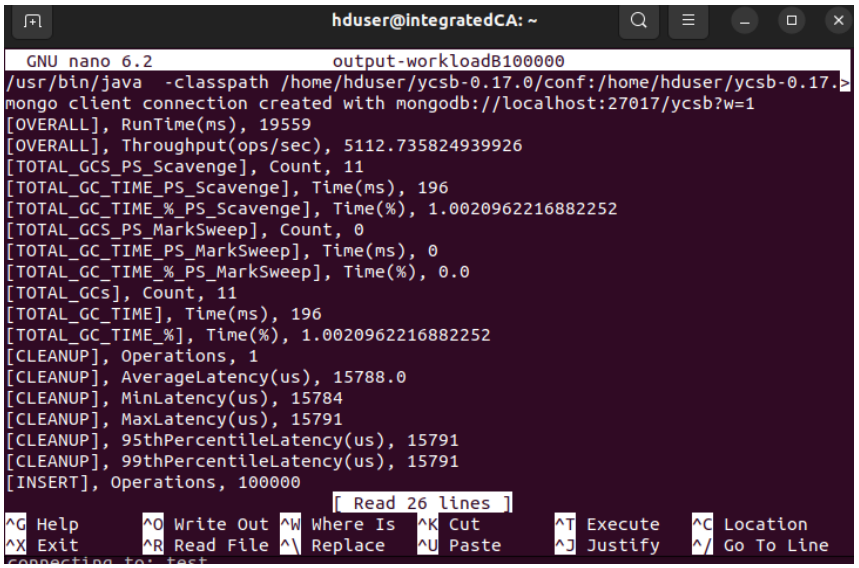
1.11 Running workloada

```
# Copyright (c) 2010 Yahoo! Inc. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"); you
# may not use this file except in compliance with the License. You
# may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied. See the License for the specific language governing
# permissions and limitations under the License. See accompanying
# LICENSE file.


# Yahoo! Cloud System Benchmark
# Workload A: Update heavy workload
#   Application example: Session store recording recent actions
#
#   Read/update ratio: 50/50
#   Default data size: 1 KB records (10 fields, 100 bytes each, plus key)
#   Request distribution: zipfian

recordcount=10000
operationcount=1000
workload=site.ycsb.workloads.CoreWorkload

readallfields=true

readproportion=0.5
updateproportion=0.5
```

1.12 Editing recordcount for workloadb

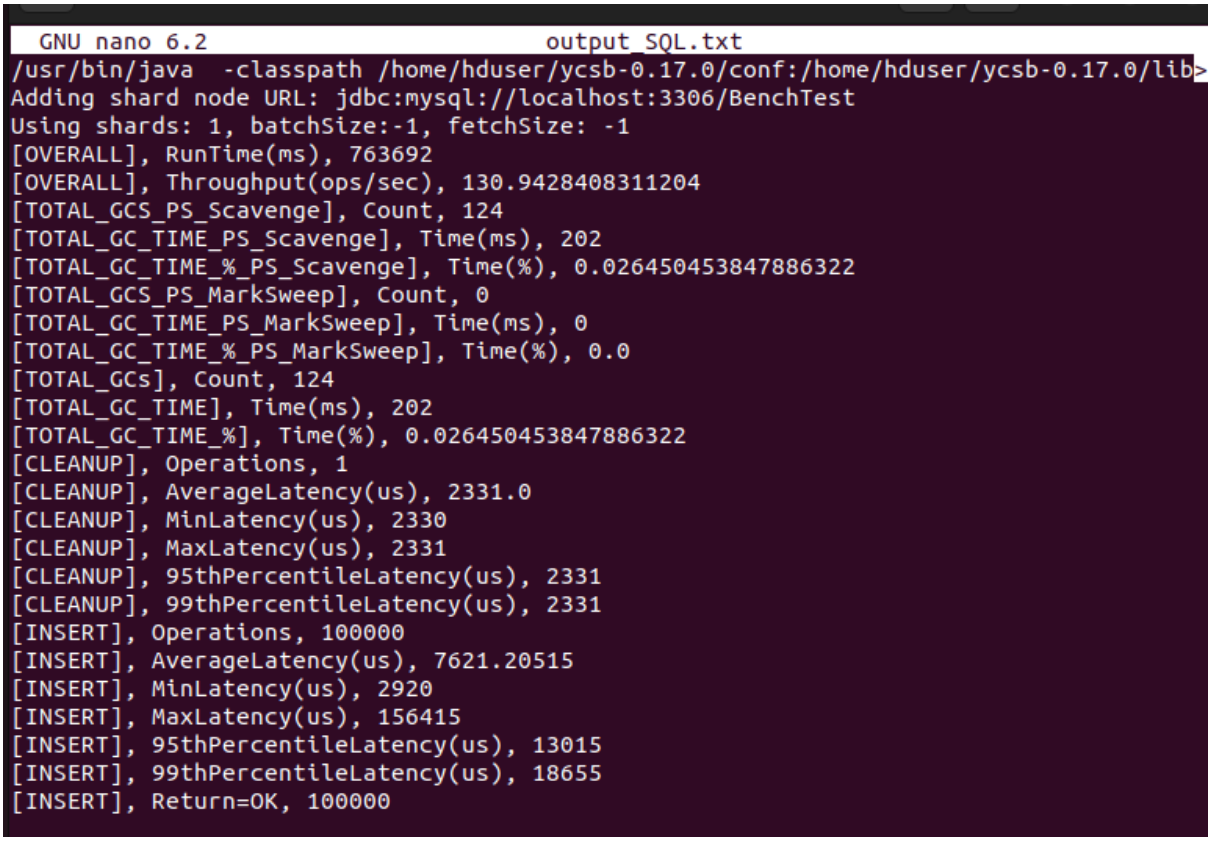


1.13 Output workloadb with 10,000 rows mongoDB



1.14 Output workloadb with 100,000 rows mongoDB

After running those workloads in mongoDB I decided to run the 100,000 rows workloadb in mySQL to compare the benchmark with mongoDB, mongo was way faster, during the mySQL execution I even thought that the machine had crashed due to the time it took, much longer than MongoDB.



1.15 Output workloadb with 100,000 rows mySQL

APPLICATION FUNCTIONALITY DESCRIPTION AND IMPLEMENTATION

Login System

Created an HTML file for the login page, left the fields ready to be read and checked with thymeleaf and implemented all login functionality

To login the user must enter their username and password, then when the user hits the login button we verify if the parameters passed match an existent user in the database, if the credentials are correct the user may login which will create a user session, the other pages on the website won't open if user is not logged in, I (Daniel) did that by passing a parameter to the page that only is instantiated if the user logs in, otherwise the webapp will take you to a formatted error page, which will tell the user to log-in.

I also implemented the MD5 algorithm which diggests the user password and generates a same size hash every time, this way even in plain sight someone wouldn't be able to guess someone else's password.

I also created a method that checks if an email or username has been already registered in the database, this way the app won't allow anyone to create a new user if the username or email are already registered.

JEFFERSON'S CONTRIBUTION

As part of the integrated continuous assessment for our course in commercial solutions development and data storage solutions, we were tasked with creating a fully functional web application. This web application needed to meet specific requirements, including obtaining a dataset from any repository, storing the data on the Hadoop platform, loading the dataset using NoSQL, and processing queries based on the chosen dataset. The web application had to be developed using Java Spring and In this report, I will discuss my experiences, the challenges I faced, and the lessons I learned throughout the project.

Initial Setup and Configuration

The first step in the project was setting up the development environment and configuring the necessary tools and libraries. As part of the requirement, we were told to use the Spring Boot framework. Initially, I was not very comfortable with this choice because I had never used it before, and it seemed quite complex. However, as the project progressed, my understanding of the framework improved, and the development process became smoother. In addition, I utilized Thymeleaf as a template engine for rendering dynamic HTML content.

One of the primary challenges I faced during the project was the limited resources and tutorials available for integrating MongoDB with Java Spring compared to the more widely documented MySQL and Java Spring integration. In this report, I will describe the process of connecting Java Spring to MongoDB, creating the HTML templates with Thymeleaf, handling user input and validation, and the lessons learned from overcoming these challenges.

Connecting Java Spring with MongoDB

While there are numerous tutorials and resources available for integrating Java Spring with MySQL, finding adequate resources for MongoDB proved to be more challenging. Most examples and tutorials I found were specific to MySQL or other relational databases, while MongoDB is a NoSQL database. This difference in database types required me to adapt my approach to data modeling and persistence.

To overcome this challenge, I referred to the official MongoDB and Spring Data MongoDB documentation, which provided valuable information on setting up and configuring the project to work with MongoDB. By studying these resources, I gained a better understanding of the differences between SQL and NoSQL databases, and how to properly model and interact with the data stored in MongoDB.

Creating the HTML Templates with Thymeleaf

After successfully connecting the Java Spring application to the MongoDB database, the next step was to create the HTML templates for the user interface. Thymeleaf, a Java-based template engine, was employed to generate dynamic HTML content. While working with Thymeleaf, I encountered some difficulties in understanding its syntax and how it integrates with Java Spring.

To address these challenges, I studied various tutorials and resources on Thymeleaf and its integration with Java Spring. By doing so, I was able to comprehend how to create dynamic HTML templates using Thymeleaf and how to pass data between the Java Spring controllers and the templates.

Handling User Input and Validation

Another challenge I faced during the development of the project was handling user input and validation. Ensuring that users input valid data and providing appropriate feedback in case of invalid data is crucial for a user management system. To address this issue, I researched and implemented input validation techniques using Java Spring and Thymeleaf.

By understanding how to use Java annotations, such as @Valid and @NotNull, I could validate user input on both the server and client sides, ensuring that only valid data is persisted in the database.

Conclusion

Developing a Java Spring application with MongoDB presented numerous challenges, mainly due to the limited resources and tutorials available for this specific combination of technologies. By referring to the official documentation, studying various tutorials, and persistently experimenting with different approaches, I successfully overcome these challenges and gained valuable experience in working with Java Spring and MongoDB.

Throughout this project, I have learned the importance of adapting to new technologies and the value of perseverance when faced with difficulties. Furthermore, I learned that working with unfamiliar technologies can be a daunting task at first, but with determination and a willingness to learn, it can be a fulfilling experience. Additionally, I discovered that having a solid foundation in the fundamentals of programming and databases is critical for working with new technologies and efficiently troubleshooting issues.

Moreover, I realized the significance of documentation in the development process. Having accurate and thorough documentation can save a lot of time and effort, especially when working with complex technologies such as MongoDB and Java Spring. Documentation also provides a valuable resource for troubleshooting and learning new concepts.

In conclusion, the project was a challenging yet rewarding experience. It provided me with valuable insights into working with new technologies, overcoming challenges, and the importance of documentation. The project also honed my skills in data modeling, database management, and web development using Java Spring and MongoDB. Overall, I believe that this project has prepared me for future endeavors in software development and has equipped me with a new skillset that I can leverage in my career.

BEKEZHAN ABDYKARIMOV 2020297

As part of the team, I was responsible for working with the Spotify tracks dataset and implementing CRUD functionality on this dataset to allow users of the Music Management System application to manage their music collections. In this personal contribution report, I will discuss the process of completing my part of the project, the challenges I faced during the project, my thoughts on the project, what I learned, and what I would improve.

At the outset, I didn't have an idea of how we could even build the app since we had only worked on small console-based applications before, and the lectures did not provide me with enough confidence about the tools we had to use. Additionally, the project included many different subjects and technologies that we had covered previously, but never in combination. As a result, it was challenging for me to focus on so many things at the same time. To address this, I decided to gain knowledge and build confidence by enrolling in a few courses on Udemy about the SpringBoot framework and how to work with it. This was the right decision since these courses provided concentrated and modern information about the tool, with exercises and examples of some design principles. Though I didn't finish all the courses, covering the basics allowed me to understand the fundamentals, making it easier to go through the code and the project in general, saving me a lot of time.

Connecting Java Spring with MongoDB

After gaining knowledge about the basics, I started creating the basic app functionality, which involved adding the dataset to the database, which was not an issue. The main challenge was displaying the database content since I had never worked with such a large dataset before, and I did not know all the issues such as load time, etc. Initially, I thought that my IDE was not working correctly, since it was loading the data for 10 minutes, or sometimes even refusing to work. However, I found the issue, which was the large size of the dataset. The problems were perfectly explained in the MongoDB documentation, which suggested using pagination. Pagination is a method that splits the large dataset into smaller pieces and loads them as a specified group of documents. As Charles Kettring (Head of the General Motors company) said, "A well-stated problem is half solved." I quickly found a way to use pagination and fix the issue.

Handling the HTTP Requests

Another challenge that I faced was working with HTTP requests and how to send a certain information from page to page. I still do not fully manage this flow of the HTTP requests, but it was enough to complete the task I wanted to do, which was to keep the user on the same page after deleting or updating the track with the same page configuration (in my case, it was sortField since it affected the position and the page where the track was originally, so it was important to be able to keep it). As I said above, this still confuses me a bit, but after watching some tutorials on YouTube and understanding that we need to have key-value pairs that we can send from page to page, it was a relief. The only challenging part is to manage that flow and not get lost on that. I have not found a

good way of how to keep track of that flow yet, except writing and drawing maps manually, but I am pretty sure there is a tool or method that helps to simplify the work, unfortunately, I have not found it yet.

Time and Mistakes

The last challenge was time since this type of project was new to me. The majority of the time, I spent reading, Googling, and debugging errors, and I couldn't completely finish all the functionality that I wanted to add. Originally, I wanted each user to be able to create their playlists, add tracks, and write comments on the Hot100 chart page, but unfortunately, I also learned a lot about the importance of seeking out additional resources and learning opportunities outside of the classroom. The Udemy courses I took were extremely helpful in filling in the gaps in my knowledge and giving me the confidence I needed to tackle this project. I now realize that self-directed learning is a valuable skill to have, especially in the fast-paced field of technology where new tools and frameworks are constantly emerging.

Looking back on this project, there are a few things I would improve. First, I would like to spend more time exploring different approaches to managing the flow of HTTP requests between pages. While I was able to accomplish my goals with my current understanding, I believe there are more efficient and effective methods out there that could streamline the process. Second, I would like to further optimize the app's load time, perhaps by exploring different pagination techniques or implementing caching. Finally, I would like to complete the remaining functionality that I was not able to finish due to time constraints, as I believe it would enhance the overall user experience and make the app more useful.

Conclusion

In conclusion, working on the Spring MVC web application for the music management system app was a challenging and rewarding experience. It allowed me to apply my knowledge and skills in a practical setting and gave me the opportunity to learn new technologies and techniques. While I faced several challenges along the way, I was able to overcome them with persistence and the help of additional resources. Overall, I am proud of what I have accomplished and excited to continue learning and growing in the field of software development.

References :

MongoDB. (n.d.). *MongoDB Atlas Tutorial*. [online] Available at: https://www.mongodb.com/basics/mongodb-atlas-tutorial#:~:text=MongoDB%20Atlas%20is%20a%20fully.

www.youtube.com. (n.d.). *Curso Spring Boot - YouTube*. [online] Available at: https://www.youtube.com/playlist?list=PL0j7juv7l4HgSY7gNDzNQjgwEA5s4hzjx [Accessed 7 May 2023].

https://www.baeldung.com/spring-redirect-and-forward

https://www.w3docs.com/snippets/css/how-to-set-the-width-of-the-table-column.html#:~:text=If%20you%20want%20to%20set,col%3E%20elements%20having%20span%20attributes.

http://jtuts.com/2016/05/24/validating-double-values-spring/

https://www.javatpoint.com/spring-mvc-number-validation

https://www.youtube.com/watch?v=5PdEmeopJVQ&t=5395s

https://www.udemy.com/course/spring-hibernate-tutorial/learn/lecture/36837720?start=345#overview