[360 min] BUILD: Project 3 / Design & Implement a Key-Value In-Memory Database

Submit Assignment

Due Friday by 11:59pm **Points** 100 **Submitting** a website url

Overview

In this practicum you will modify the database you built for previous projects to adjust them to a key-value in-memory database (Redis). You can reuse the project description, selecting one data structure that makes sense to be implemented as an in-memory store. Finally you will modify your existing codebase to make it work with Node and Redis.

Format

- This work should be completed individually

Tasks

1. (15 pts) Provide the problem requirements and the conceptual model in UML for your project. You can reuse the one made on previous projects, but describe the functionalities that you selected to be used as an in-memory key-value storage, (e.g. most viewed products, a shopping cart, current logged-in users, etc).
2. (25 *pts*) Describe the Redis data structures that you are going to use to implement the functionalities you described in the previous point. (e.g. To implement the most viewed products I will use a Redis sorted set with key "mostViewed:userId", product ids as the values and a score of the number of views of the product.). You can use/describe more than one data structure, you will need to implement at least one.
3. (60 *pts*) Create a basic Node + Express application that let's you create, display, modify and delete at least one Redis data structure from the ones describe in the previous point. No need to have a polished interface, and you can use the code created in class as a starting point, and/or the code you created for previous projects

Minimum Requirements & Grading Rubric

- At least one data structure should have all CRUD operations implemented

Submission

Submit a Github Repository containing:

A. The requirements document as a PDF.
B. Definition of Documents/Collections/Tables as example JSON objects.
C. The code of your basic application with a proper README file

P1.Su20.Rubric (1)

Criteria	Ratings						Pts
Q1 Provide the problem requirements and the conceptual model in UML for your project. You can reuse the ones made in Project 1.	**5.0 pts** **Only very minor issues**	**3.34 pts** **Very good**	**1.67 pts** **Adequate**	**0.0 pts** **No Marks**			5.0 pts
Q2 logical data model expressed as an ERD, with clear hierarchies defining your main documents. Relations can be aggregations/compositions, many to many relationships and one to many relationships	**15.0 pts** **Only very minor issues**	**12.0 pts** **Very good**	**7.5 pts** **Adequate**	**4.5 pts** **Acceptable**	**0.0 pts** **No Marks**		15.0 pts
Q3 Sample descriptions of the Documents in JSON, with examples of possible records annotated if necessary	**10.0 pts** **Excellent**	**6.67 pts** **Acceptable**	**3.33 pts** **Reasonable attempt but not adequate**	**0.0 pts** **No Marks**			10.0 pts
Q4 Test data. Initialization files and instructions on how to restore the database	**15.0 pts** **Well defined test data that tests all aspects of schema**	**12.0 pts** **Good test data but not sufficient to test all aspects of schema**	**7.5 pts** **Adequate**	**4.5 pts** **Some test data but not nearly sufficient**	**0.0 pts** **No Marks**		15.0 pts
Q5 Define and execute at least five queries that show your database. At least one query must contain and aggregation **https://docs.mongodb.com/manual/aggregation/Links** (https://docs.mongodb.com/manual/aggregation/Links) to an external site. (Links to an external site.) , one must contain a complex search criterion (more than one expression with logical connectors), one should be counting documents for an specific user, and one must be updating a document based on a query parameter (e.g. flipping on or off a boolean attribute for a document, such as enabling/disabling a song)	**30.0 pts** **Full Marks**	**24.0 pts** **Minor issues or a missing query**	**18.0 pts** **Missing some aspect**	**12.0 pts** **Missing some aspect**	**6.0 pts** **Missing some aspect**	**0.0 pts** **No Marks**	30.0 pts
Q7 Create a basic Node + Express application that let's you create, display, modify and delete at least two of the tables. One of the tables can be the users table. No need to have a polished interface, and you can use the code created in class as a starting point, and/or the code you created for Project 1	**25.0 pts** **Full Marks**	**20.0 pts** **Backend in Node works, but doesn't have front end**	**12.5 pts** **Adequate**	**7.5 pts** **Acceptable**	**0.0 pts** **No Marks**		25.0 pts
							Total Points: 100.0