FAF.PAD16.2 Autumn 2024
Lab 2: Logically Linked DBs

Handed out:	October 26, 2024
Due:	December 13, 2024

Checkpoints Terms

As with the previous laboratory, the final grade will be calculated based on three marks, with the previously mentioned House Rules in regards to the presentation process (and its modifications) still being applied.

Checkpoint 1

Update your architecture with features you want to implement. Add in READ.ME a mini documentation about all your endpoints, how to run/deploy your project and steps to run/test docker images. Make clear what endpoint should be accessed first (if any), bodies/parameters for all of them. Also export Postman collection for all endpoints as json and upload it to Github.

Checkpoint 2

For Checkpoint 2, student is required to implement a part of the laboratory work, see Requirements section for tasks and marking system. Interpret it as being a Minimum Viable Product (MVP), and defend to the professor a functional or partially functional implementation.

The grading for this checkpoint will be specific. Students who miss the deadline will receive a grade of 1, which will be automatically recorded in the grade book. Those who present will be given a grade according to the number of requirements that have been implemented.

Checkpoint 3

Further details on the process of advance submission of the checkpoint and testing conditions will be provided by the professor a few days before the presentation day.

Requirements

Mark	Team size: 1	Team size: 2
1	just be	(just be) x2
2	• Mark 1 • trip **Circuit Breaker** if multiple re-routes happen	• Mark 1 • trip **Circuit Breaker** if multiple re-routes happen
3	• Mark 2 • **Service High Availability**	• Mark 2 • **Service High Availability**
4	• Mark 3 • implement **ELK stack**[3] or **Prometheus[4]** + **Grafana[5]** for logging. Aggregate data from ALL services	• Mark 3 •implement **ELK stack**[3] or **Prometheus[4]** + **Grafana[5]** for logging. Aggregate data from ALL
5	• Mark 4 • implement microservice-based **2 Phase Commits** for an endpoint that create changes more than in one database (create new endpoint if needed).[1]	• Mark 4 • implement microservice-based **2 Phase Commits** for an endpoint that create changes more than in two databases (create new endpoint if needed).[1]
6	• Mark 5 • **Consistent Hashing for Cache** [7]	• Mark 5 • **Consistent Hashing for Cache** [7]
7	• Mark 6 • implement **Cache High Availability**	• Mark 6 • implement **Cache High Availability**
8	• Mark 7 • instead of **2 Phase Commits** implement **Long-running saga transactions** with coordinator [1]	• Mark 7 • instead of **2 Phase Commits** implement **Long-running saga transactions** with coordinator [1]
9	• Mark 8 • **Database redundancy/replication + failover** - implement any kind of replication for at least one database, minimum 3 replicas [2]. In case of trying to reach mark 9, mark 7 condition may be skipped.	• Mark 8 • **Database redundancy/replication + failover** - implement any kind of replication for at least two databases, minimum 4 replicas [2]
10	• Mark 9 • create a **Data Warehouse** that will be periodically updated with all data from your databases. Use any ETL you want, it can be a job or separated service [9]	• Mark 9 • create a **Data Warehouse** that will be periodically updated with all data from your databases. Implement your own ETL as separated service [9]

Readings

[1]Keyang Xiang. “Patterns for distributed transactions within a microservices architec- ture”. https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed- transactions-within-a-microservices-architecture.

[2]Ben Lutkevich. “database replication”. https://searchdatamanagement.techtarget.com/definition/database-replication.

[3] Elasticsearch. “What is the ELK Stack?”. https://www.elastic.co/what-is/elk-stack.

[4] Cloud Native Computing Foundation. “Prometheus”. https://prometheus.io.

[5] Grafana Labs. “Grafana”. https://grafana.com.

[6] Redis. “Replication”. https://redis.io/topics/replication.

[7]Juan Pablo Carzolio. “The Ultimate Guide to Consistent Hashing”. https://www.toptal.com/big-data/consistent-hashing.

[8]l3a0. “Distributing a Cache”. https://blog.baowebdev.com/2019/04/distributing-a- cache.

[9] Redis. “Sharding”. https://redis.io/docs/latest/operate/oss_and_stack/management/scaling/

[10]“What is a data warehouse”. https://www.ibm.com/topics/data-warehouse.

Good Luck!