celonis

# Project Task (C++): Efficient Key-Value-Store

**Challenge**

The challenge is to create an efficient, performance optimized key-value-store by making use of modern C++. With this application, users should be able to add/update values by key, retrieve values by key and delete values by key.

For demonstration purposes the application should be network-accessible. Please also provide a client for which you can use any library or language.

**Requirements**

The following functionalities should be implemented:

Interface:

- Keys are strings, values are strings as well. Both have variable length
- A “PUT” call which adds a pair to the store
- A “DELETE” call which deletes a pair from the store by key
- A “GET” call which retrieves a pair from the store by key

Internal functionalities:

- Multiple clients can access the KV-Store simultaneously (multi-threaded/synchronization)
- A cache eviction strategy should be implemented. The server should be constrained to a certain amount of memory (e.g. 10 MB). When this amount is reached, it should evict keys based on this strategy (e.g. FIFO, LRU, …)
- Instead of evicting the cache keys, they should be swapped out of memory onto a persisting storage (e.g. disk) and picked up when they are required again.

**Presentation**

Please prepare a demo/presentation of your implementation. This implementation will be discussed during the interview. Please consider the following points:

- Technology overview and reasoning behind choosing those technologies
- Design of data structures
- Performance characteristics of key-value-store
- Cache strategy employed
- Scaling of data structures and memory structure

Any questions? Vanessa or Eva are happy to help you -> Mail: v.mantel@celonis.de or e.matzner@celonis.de