U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Shopping Cart

SDLE Project - 2024/25 Grupo 11

Hugo Castro up202403131 | Marcos Costa up202108869 | Rodrigo Moucho up202108855

Overview

- **Local-First Design**: The app executes code directly on user devices and ensures data persistence by saving changes locally. This allows users to continue using the application seamlessly, even without an internet connection or during server disruptions.
- **Server Side:** Broker receives requests from the client and sends that data to the respective worker. The respective worker is calculated by hashing the list url.
- The app was developed in Python using ZeroMQ for connections and JSON for data storing

Local-First

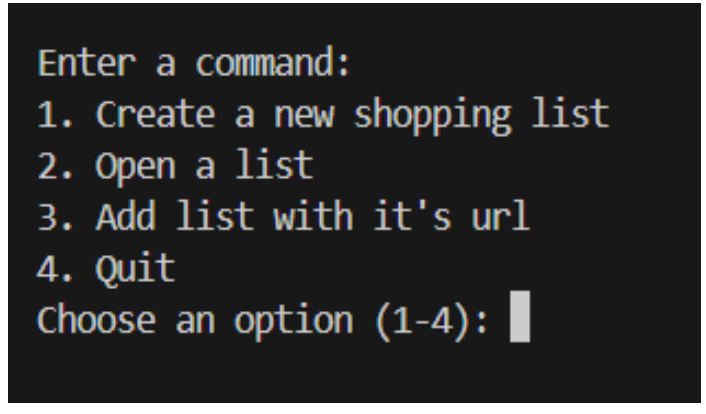
Each list is represented by:

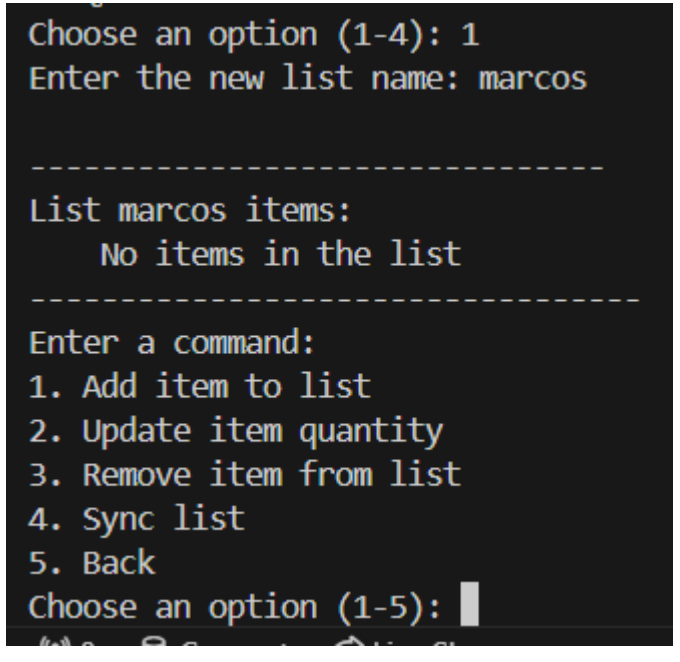
- list_id, list_name, list_url (unique), quantities

GUI and Cloud Connection:

A simple graphical interface was developed to enable seamless interaction with the application. Users can create new lists, manage existing lists, or retrieve lists from the cloud.

Each user has the option to sync their current list with the cloud, merging it with the most up-to-date version stored online. This ensures data consistency, fault tolerance, and facilitates data recovery in case of any disruptions.





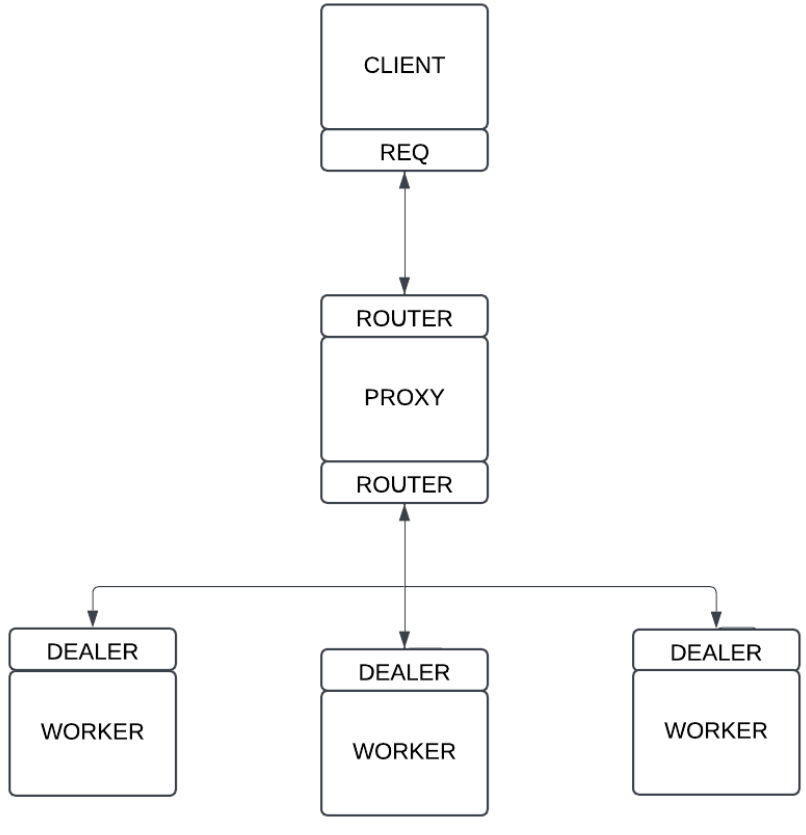
Architecture

Fault tolerance: every change is stored in the local database, allowing for data recovery in case of any type of failure

Consistent Hashing: in order to balance the load between all the available workers, each worker has a similar range of positions in the ring that they are responsible for.

Additionally, each worker has multiple nodes in the ring (virtual nodes) to help balance the load.

Ring Replication: when the client does a write operation, the responsible worker will propagate it to other workers (2 in our policy)

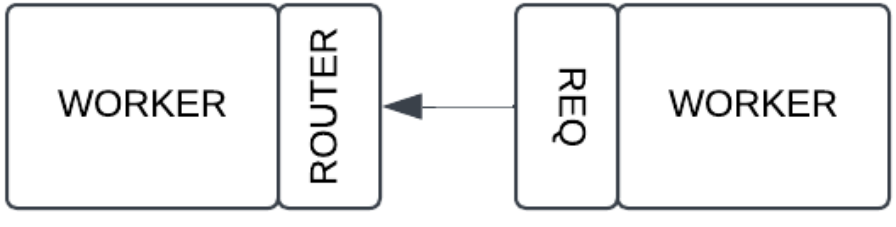


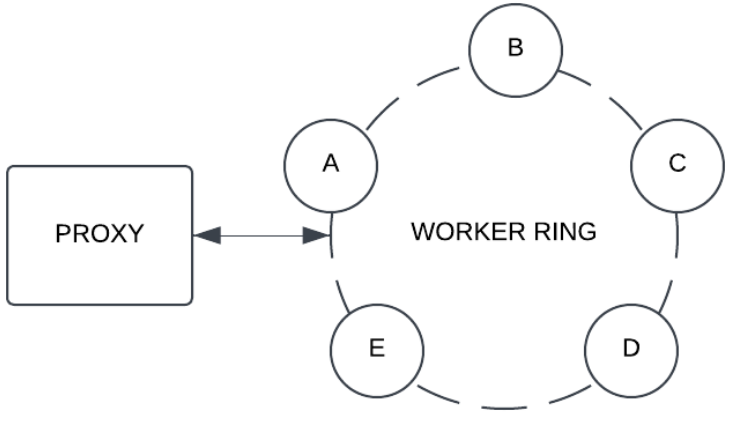
Workers

Each worker periodically sends a “heartbeat” to the broker to signal they are still functional.

They handle the write and read operations, store it asynchronously, and send the result state to the proxy that will then send it to the client.

Data Replication: in every write operation the responsible worker will attempt to propagate and sync the list with two other workers from a set of up to 5 workers that follow the initial node in the hash ring (to account for possible failures of nodes)





CRDTs

In order to avoid merge conflicts, CRDTs were implemented to avoid loss of information

- **AWORSet:** Manages set of items with conflit resolution
  - Assigns unique IDs to each item addition/removal
  - Tracks "tombstones" to remember deleted items
  - Ensures no item is accidentally lost during synchronization
- **PNCounter:** Used for incrementing/decrementing item quantities from lists
  - Stores separate increment/decrement counts per device
  - Each device tracks its own quantity changes

Desing Implementation and Challenges

- Deciding sockets for the proxy (ROUTER-DEALER vs ROUTER-ROUTER)
- Implementation of the worker hashring
- CRDT choice and integration
- Communication between workers, worker and broker, and broker and client

Conclusions

- Successfully implemented a shopping list application that can persist data locally
- Cloud component to share data among users and provide backup storage.
- Fault tolerance mechanisms to improve robustness of the system.
- CRDT’s to synchronize clients' concurrent changes