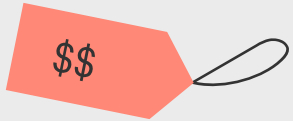
Shopping Lists On The Cloud

FEUP | MEIC | By group T8G14

João Sobral, José Isidro, Nuno Fraça. Tomás Macedo



Index

01

Context and Overview

02

Architecture

03

Design decisions

04

Demo

05

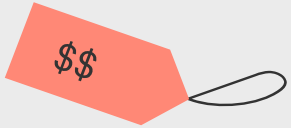
CRDT

06

Technologies

01

Context and Overview



Context and Overview

Objective : Develop a shopping list application with both local and cloud components, ensuring data persistence, sharing, and scalability.

Local-First Design

- Application operates on user devices to ensure local data persistence.
- A cloud component provides data sharing and backup capabilities.
- Each shopping list is associated with a unique identifier for easy sharing and access.
- Use of conflict-free Replicated Data Types (CRDTs) for improved consistency and conflict resolution.
- Leverage sharding techniques inspired by the Amazon Dynamo paper to ensure lists are independent and avoid bottlenecks.

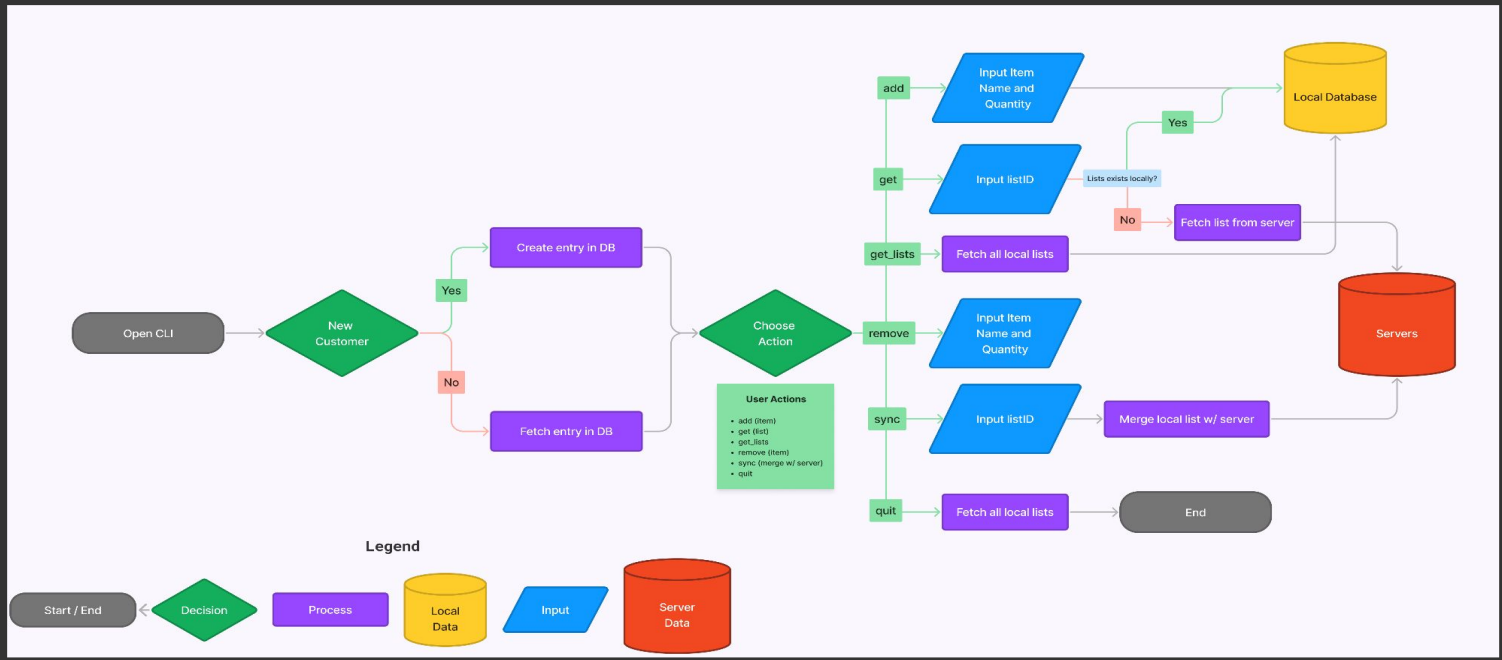
Collaborative Lists

- Users with the unique identifier of a list can collaboratively add or remove items.

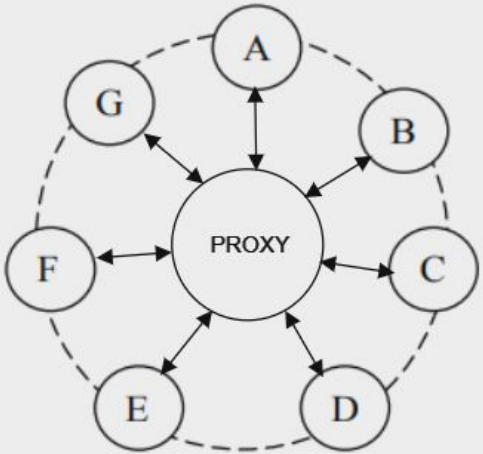
02

Architecture

Flow


Open CLI
New Customer
Yes
No
Create entry in DB
Fetch entry in DB
Choose Action
User Actions
add (item)
get (list)
get_lists
remove (item)
sync (merge w/ server)
quit
add
Input Item Name and Quantity
Local Database
Yes
get
Input listID
Lists exists locally?
No
Fetch list from server
get_lists
Fetch all local lists
Servers
remove
Input Item Name and Quantity
sync
Input listID
Merge local list w/ server
quit
Fetch all local lists
End
Legend
Start / End
Decision
Process
Local Data
Input
Server Data

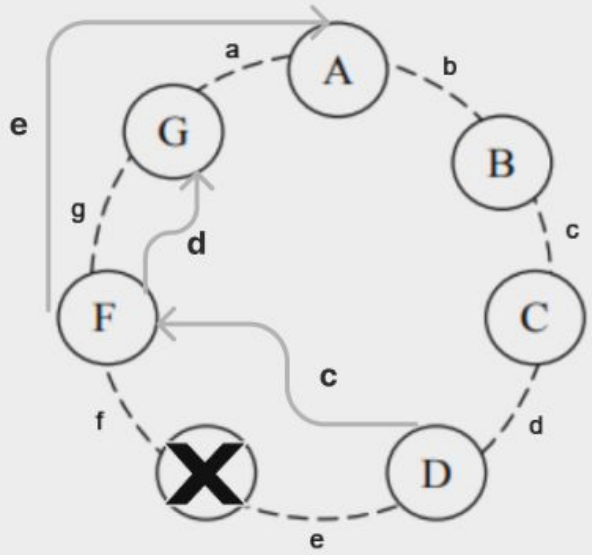
Ring



Workers

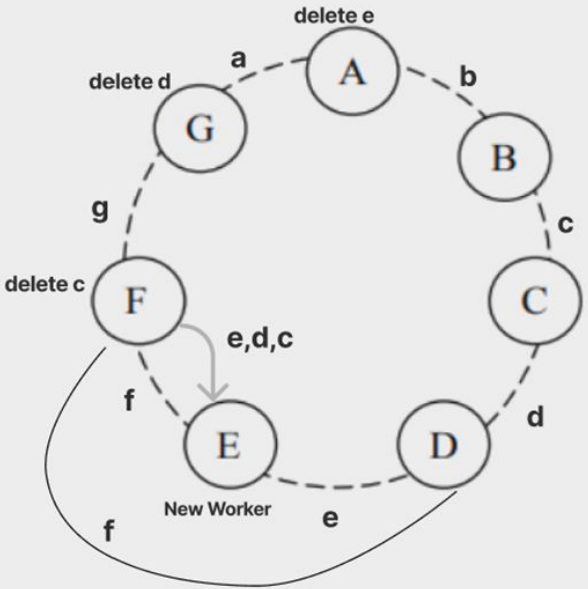
- Ring overview
- Next two neighbors (where you will replicate your lists)
- Two previous neighbors (which neighbors have their data replicated)

Removing Worker



- Node D connects to node F and passes it C's lists.
- Node F connects to node G and passes it D's lists.
- Node F connects to node A and passes it E's lists.

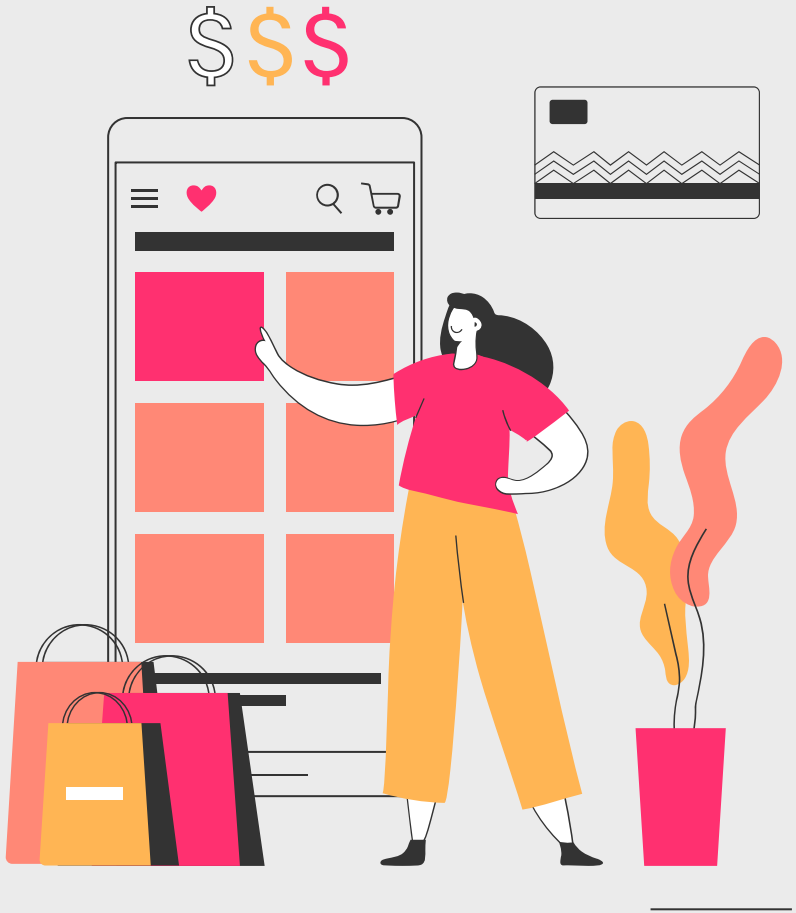
Adding Worker



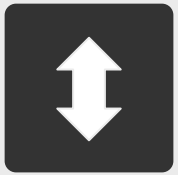
- Node F connects to node E and passes it the lists of C, E and F.
- Delete the lists from C on node F
- Delete the lists from D on node G
- Delete the lists from E on node A

03

Design decisions



Main Design Decisions



Primary Worker

- To reduce the number of hops, we communicate directly with the worker, instead of an intermediary server
- We assumed this worker is always on port 6000

Proxy

- We created an **XPUB/XSUB** for workers to communicate with one another
- Each worker has a general view of the ring, through the use of heartbeats

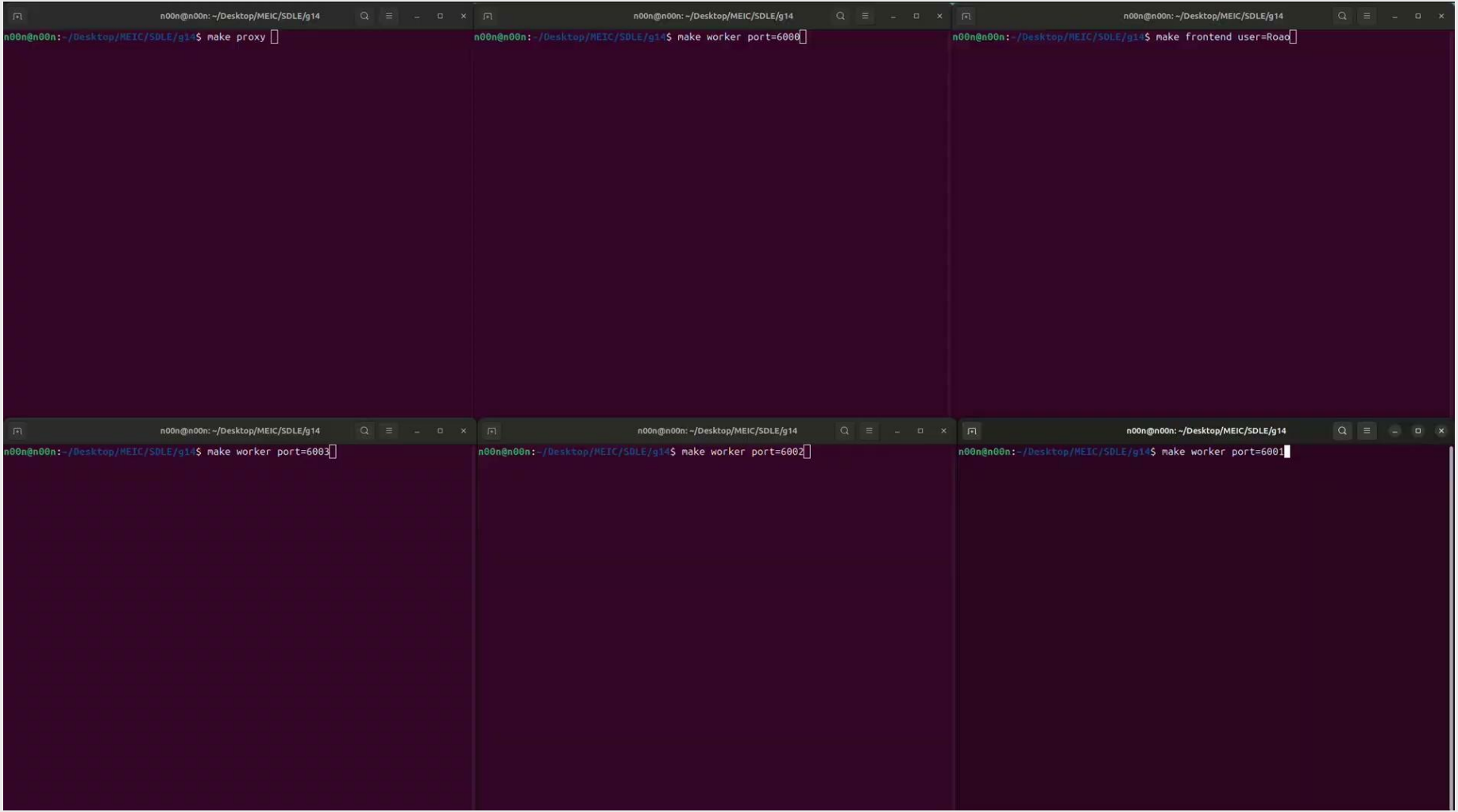
Sync

- Syncing with the server is only made at the request of the user
- This ensures that the application performs as expected locally

04

Demo!

We hope you're enjoying our presentation!


n00n@n00n: ~/Desktop/MEIC/SDLE/g14
n00n@n00n:~/Desktop/MEIC/SDLE/g14$ make proxy
n00n@n00n: ~/Desktop/MEIC/SDLE/g14
n00n@n00n:~/Desktop/MEIC/SDLE/g14$ make worker port=6000
n00n@n00n: ~/Desktop/MEIC/SDLE/g14
n00n@n00n:~/Desktop/MEIC/SDLE/g14$ make frontend user=Roac
n00n@n00n: ~/Desktop/MEIC/SDLE/g14
n00n@n00n:~/Desktop/MEIC/SDLE/g14$ make worker port=6003
n00n@n00n: ~/Desktop/MEIC/SDLE/g14
n00n@n00n:~/Desktop/MEIC/SDLE/g14$ make worker port=6002
n00n@n00n: ~/Desktop/MEIC/SDLE/g14
n00n@n00n:~/Desktop/MEIC/SDLE/g14$ make worker port=6001

05

CRDT

We choose an ORset, why?

CRDT - ORset

1. **Add-Wins Behavior**: If two replicas concurrently add and remove the same element, the "add" operation wins, and the element remains in the set.
2. **Unique Identifiers**: Elements are tracked with unique identifiers, UUIDs + timestamps, to distinguish multiple additions of the same logical value.
3. **Metadata Tracking**: The set maintains metadata for each element, recording additions and removals separately to correctly merge state across replicas.

- **Add(element, quantity)**: Adds the provided quantity of the element with a unique identifier.
- **Remove(element, quantity)**: Remove the provided quantity of the element.
- **Merge(list1, list2)**: Combines states by unioning the additions and removals from both lists.

06

Technologies

Technologies

Python

For our database and all of ours client and server side applications.

ZeroMQ

For high-performance asynchronous messaging.

Hashlib, UUID & Datetime

To generate unique identifiers across all system.

Thanks!

Feel free to ask any questions!

FEUP | MEIC | By group T8G14

João Sobral, José Isidro, Nuno Fraça, Tomás Macedo