SDLE

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

SHOPUP

A Local-First Application

CLASS 6

Afonso Pinto Diogo Neves Fábio Teixeira Tomás Torres

.”It is important to feel ownership of that data, because the creative expression is something so personal.”

Martin Kleppmann

Problem Definition

- **Local and Cloud Functionality -** The application will run on user devices, allowing data to be stored locally. It will also have a cloud component for data sharing and backup
- **CRUD Operations -** Users can create, edit or delete shopping lists and products alike through a user interface. Each list will have a unique ID for easy sharing and access. Lists remain active until they are deleted
- **Shared Access -** Users with access to a list's unique ID can add or delete items. This feature facilitates collaborative list management

Problem Definition

- **Concurrency and Data Integrity -** To handle concurrent modifications and ensure high availability, we will use Conflict-free Replicated Data Types (CRDTs) for better data integrity
- **Scalable Cloud-Like Architecture -** With the goal of serving millions of users, the cloud architecture must be designed to avoid bottlenecks. This includes considering independent list management and data sharding, similar to Amazon Dynamo

Server Side

1

Broker

Broker that receives client requests, forwards them to the correct server node, and returns the server's response to the client.
When a new node is added, the node requests the state of the ring from the broker, which it then uses to announce its existence

2

ServerNode

Nodes that contain the shopping lists. Communicate with each other to update the state of the ring.
"Heartbeat" between nodes: if a node stops responding, it is considered dead and information is transmitted to other nodes.
Implemented replication, rebalancing, and sharding through consistent hashing

Solution

Conflict-free Replicated Data Type

GCounters

- Each GCounter has a **HashMap** which is composed of:
  - **UUID(key)**: The user ID of every user who edited the product
  - **Counter(value)**: An integer value that increments for each operation done(addition or remotion)
- Merge method between counters

PNCounters

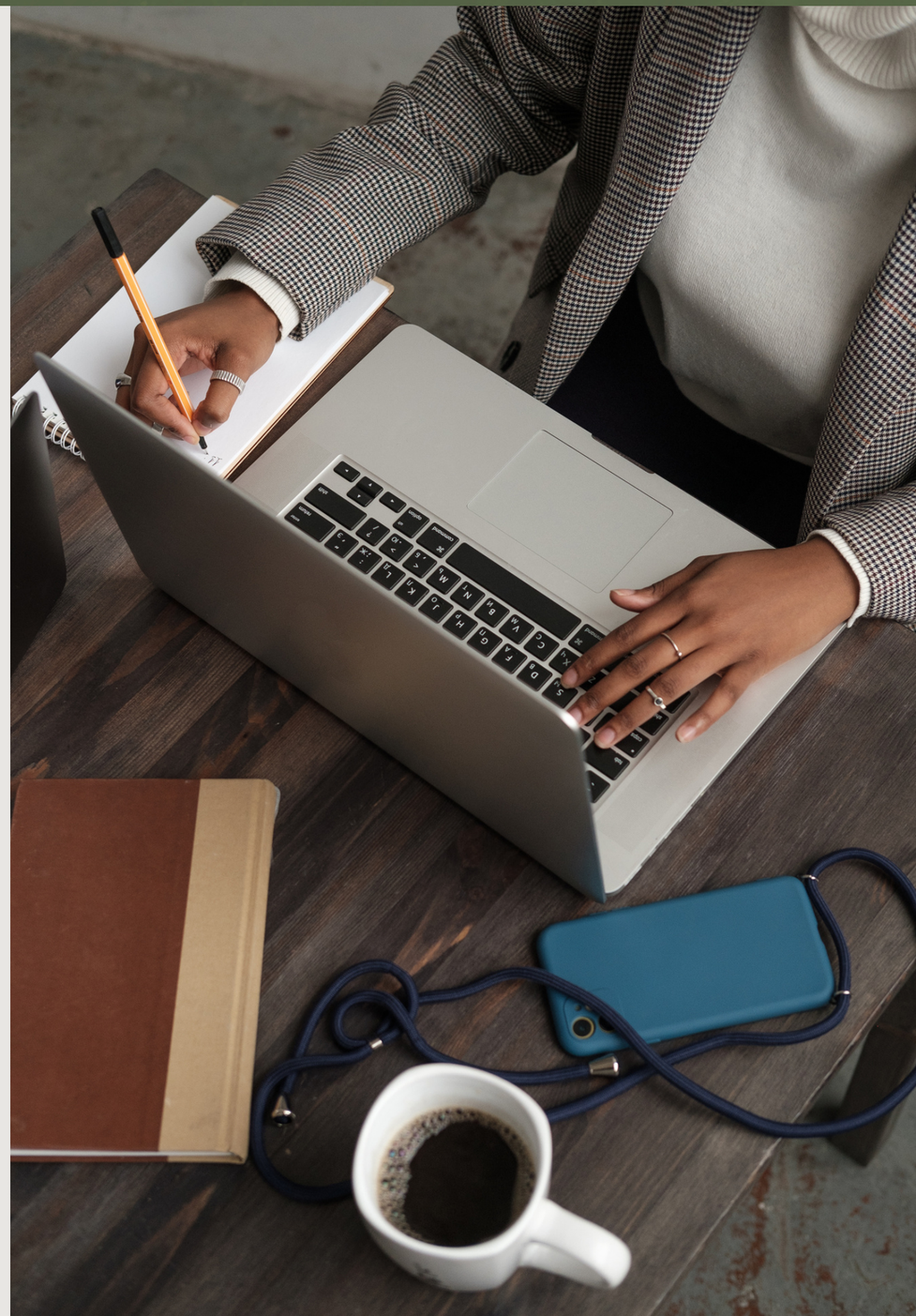
- Each product has a PNCounter associated
- Are composed of two GCounters
  - **Negative**: counts remotions
  - **Positive**: counts additions

How it works

- When a user adds or changes the quantity of a product, the counter with the key equal to the userID inside the Hashmap of the positive or negative counter will be updated
- For merge operation, the shopping list is iterated checking new products and their PNCounters
  - If there is no conflict, just add the alteration
  - If there is conflict, compare GCounters of both versions and accept the key with a higher counter

Limitations

- If the user tries to remove a product from the shopping list it only changes the counter to make the value equal to zero and when displaying the shopping list, if the quantity is zero it doesn't show on the screen
- The implementation of an AWSet could improve the CRDT and the merge function
- The broker represents a single point of failure, this could be mitigated with the use of a backup broker that takes over if the original fails
- Heartbeat for tracking nodes' health isn't ideal, as there is a small window when requests are lost. While this isn't a big problem in this project since users keep their local data and can reupload it at any time, checking if a node is alive before making a request would've solved this inconvenience



Conclusion

- Developed a local-first, distributed shopping list application that adeptly balances local data persistence with robust sharing and backup capabilities
- Achieved **consistency**, **scalability**, and **asynchronous** user collaboration with the use of CRDTs
- Although we faced many difficulties, we are proud of what we have accomplished and learned throughout this project