Local-first Shopping Application

João Fernandes - up202108044
Igor Andrade - up202108674
João Sequeira - up202108823
Gonçalo Matias - up202108703

Index

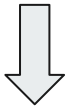
- Problem definition
- Design Choices
- Design Challenges

Problem Definition

Local-First Shopping Application

Shopping List Application with the following requirements:

- **Local-First design**-> User can shop without connection -> Sync with Server is expected!
- **Cloud** -> Share Data among others and provide backup.
- **Cart** -> There must be cart that exists under a



- **ID** -> Any User that knows a ID can add, delete items and also purchase or delete a list.

Design Choices

Technologies

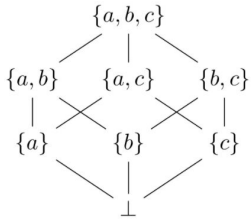
Language: Javascript (Node.JS)

Message Library: ZeroMQ -> Asynchronous and Non-Blocking Messages.

Interface Library: Inquirer -> Interface in Terminal.

Workers -> Simulate Threads in Node.JS

SQLite -> Database Management.



CRDT - Conflicted Replicated Data Types

Definition: a data structure that is replicated across multiple computers in a network.

- Used mainly for **Concurrency Control.**
- Part of the Cart Class.
- **AWORSet (Add Wins Observer Remove Set):**
  - Allows Additions and Removals
  - The removals only affect the elements that are visible locally, so that a concurrent removal and addition of the same element will result in the element still being present after joining the sets.
  - As addition takes (usually) prevalence over removal
- Each item has a **PNCounter**
  - Consists of Two **GCounter**.
  - Allows increment and Decrements
  - Used for quantity of each item.

Amazon Dynamo

Definition: DynamoDB is a distributed database system that ensures high availability and scalability by automatically spreading data across multiple servers while maintaining consistent performance.

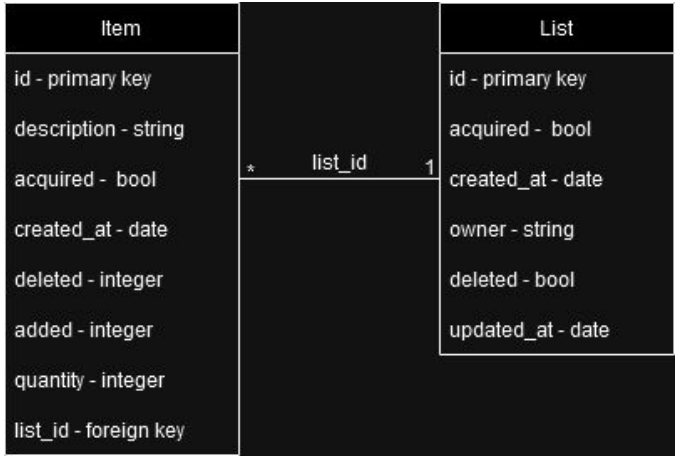
- **Consistent Hashing**
  - Ensures even data distribution
  - User virtual nodes for load balancing
- **Replication Settings**
  - N = 3 (number of replicas)
  - R = 2 (read quorum)
  - W = 2 (write quorum)
- **Node communication**
  - ZeroMQ for messaging
  - Gossip protocol for state sharing
  - Publisher/Subscriber for updates
- **Fault tolerance**
  - Automatic node recovery
  - State Maintenance
  - Message Buffering

Interface

```
? What would you like to do? (Use arrow keys)
❯ Create Shopping List
  Modify Shopping List
  Delete Shopping List
  Sync with Server
  Get all lists
  Exit
```

- Minimalist approach via terminal
- CLI-based user interface
- Allows users to manage their shopping lists with simple text-based commands
- Five core operations through the main menu: creating lists, modifying existing lists, deleting lists, synchronizing with the server, and viewing all their lists
- Supports three key actions: adding/updating items with quantities, removing items from lists, and purchasing entire lists
- Real-time synchronization is integrated into the interface

Database



- Uses SQLite as the local database system
- Each user has their own database file
- Supports offline-first architecture through local storage
- CRUD operations for shopping lists and items
- Support for concurrent list modifications
- Worker threads for database operations
- ZeroMQ integration for distributed database operations

Design Challenges

Concurrency and Data Consistency

- Multiple users modifying same list simultaneously
- Race conditions in item quantities
  - Solution: Implemented CRDT (AWORset) for conflict free merges
- Used PNCounter for reliable quantity tracking

Distributed System Coordination

- Node Failure Handling
- Data replication across nodes
- Message ordering
- Solution:
  - Implemented consistent hashing for node distribution
  - Used ZeroMP for reliable messaging
  - Created proxy pattern for message routing

References

ZeroMQ : https://zeromq.org/

Visual Studio Code : https://code.visualstudio.com/

Dynamo : https://aws.amazon.com/pt/dynamodb/

CRDT : https://crdt.tech/

Node.js : https://nodejs.org/en

SQlite : https://www.sqlite.org/

Workers: https://nodejs.org/api/worker_threads.html