SDLE

CLOUDCART

DEVELOPED BY:

AMILTON KOXI, UP202300117

FILIPE FONSECA, UP202003474

MARCELO APOLINÁRIO, UP201603903

PEDRO GOMES, UP202006086

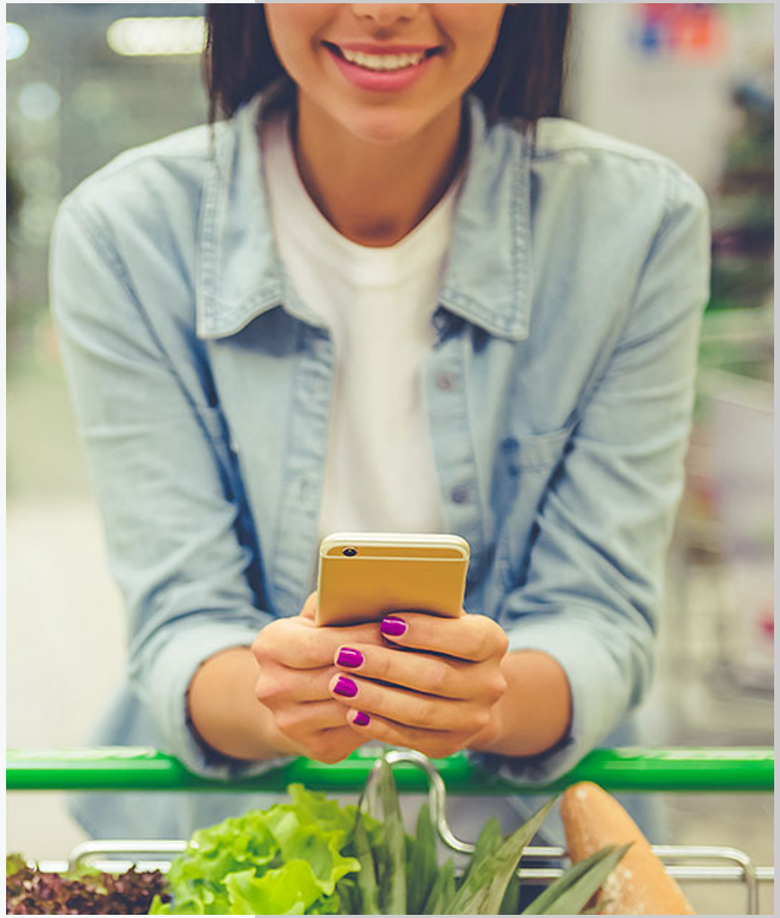
REQUIREMENTS

The Context:

- Cloud-Based Web Application for sharing and editing shopping lists collaboratively.
- Available offline and stores multiple lists per user, their names and the name and amount of each item

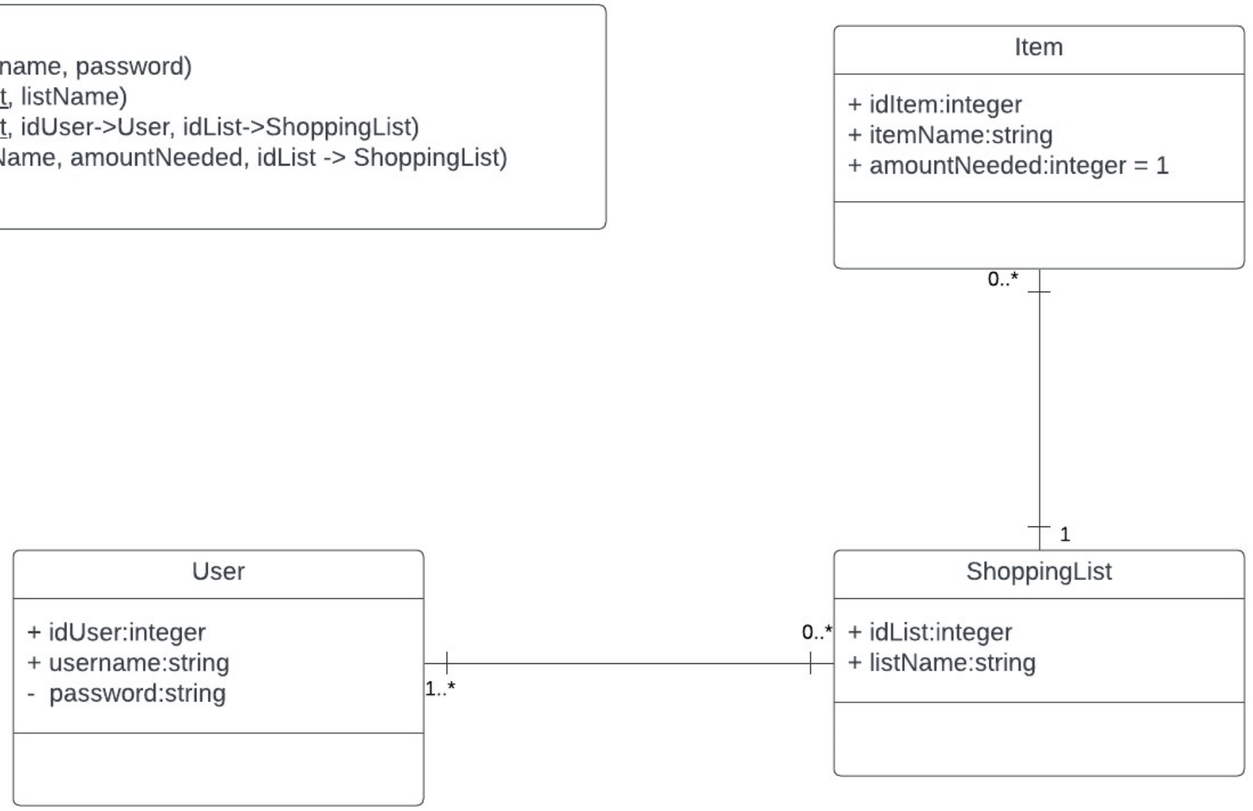
The Tools and Concepts:

- **Frontend:**
  - HTML, CSS, JavaScript, SQLite and Node.js
- **Backend:**
  - Consistent Hashing, Partitioning, Data Replication
  - Pyre (Python implementation of the Zyre framework)
- **CRDTs**



UML DIAGRAM

Relational Model:
- User (idUser, username, password)
- ShoppingList(idList, listName)
- UserList(idUserList, idUser->User, idList->ShoppingList)
- Item (idItem, itemName, amountNeeded, idList -> ShoppingList)



CONFLICT RESOLUTION DATA TYPES

Last Writer Wins

- Each update has a timestamp with time of creation (moment.js, London Time Zone)
- In case of multiple updates involving the same item, the most recent one prevails

Two-Phase Set

- Added items and removed items are registered in each replica
- Items are identified by itemName
- In case of multiple actions involving same item, the last action prevails (LWW)

CLIENT-SIDE

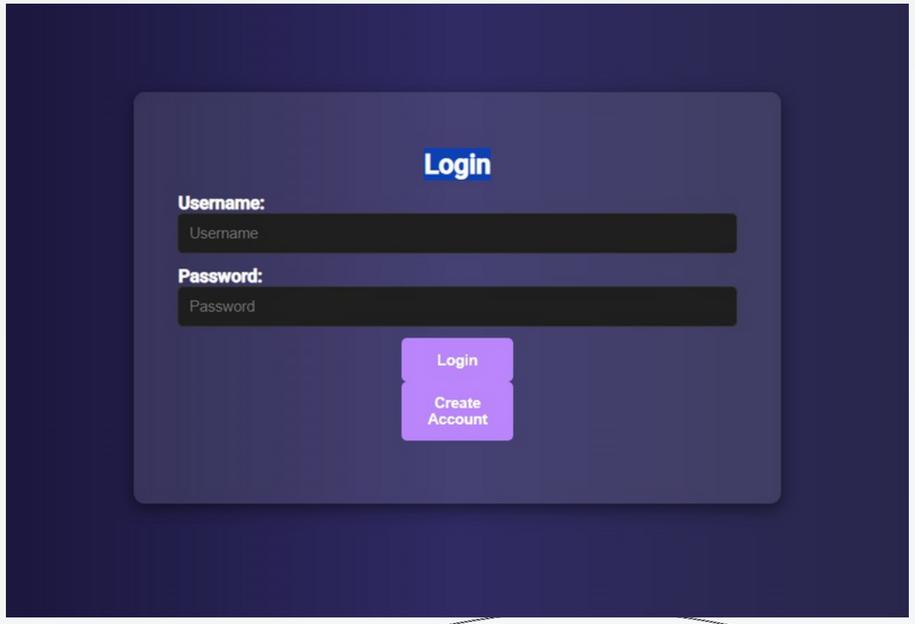
Available Operations

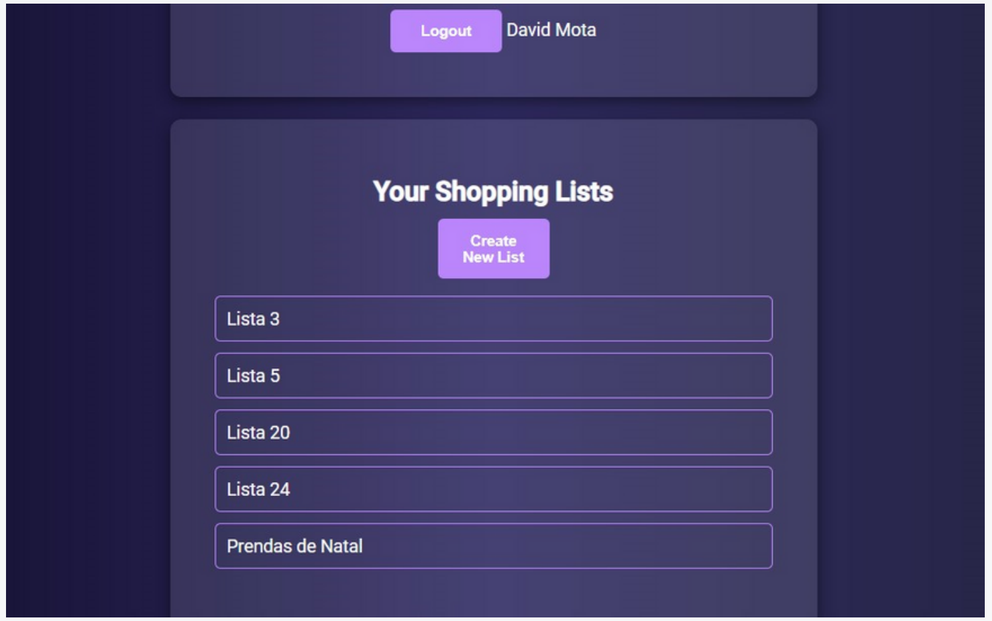
- **Authentication:** Login, Logout, Register
- **Navigation:** Display Lists and List Contents
- **List Manipulation:** Add/Remove Items, Edit Item Amounts
- **Collaboration:** Share List through Link

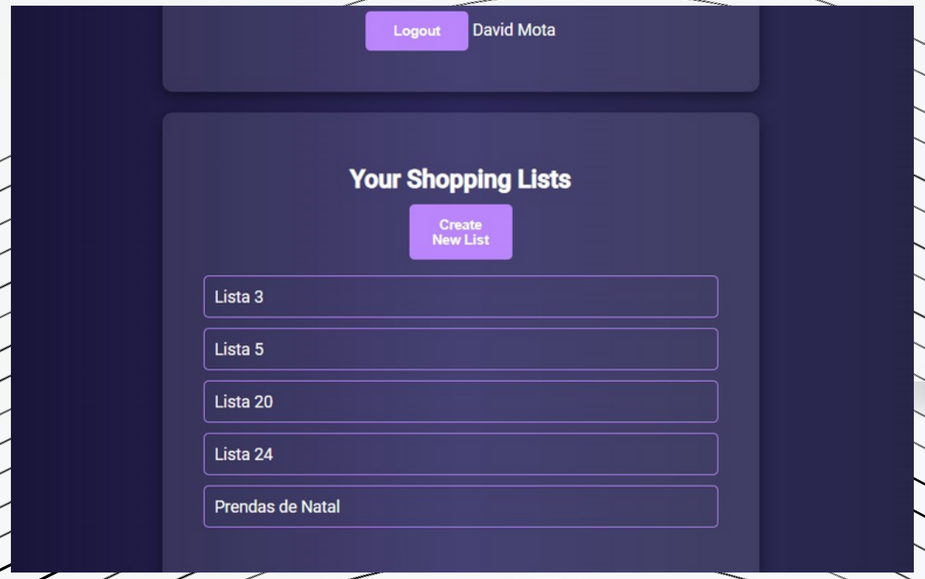
User Interface

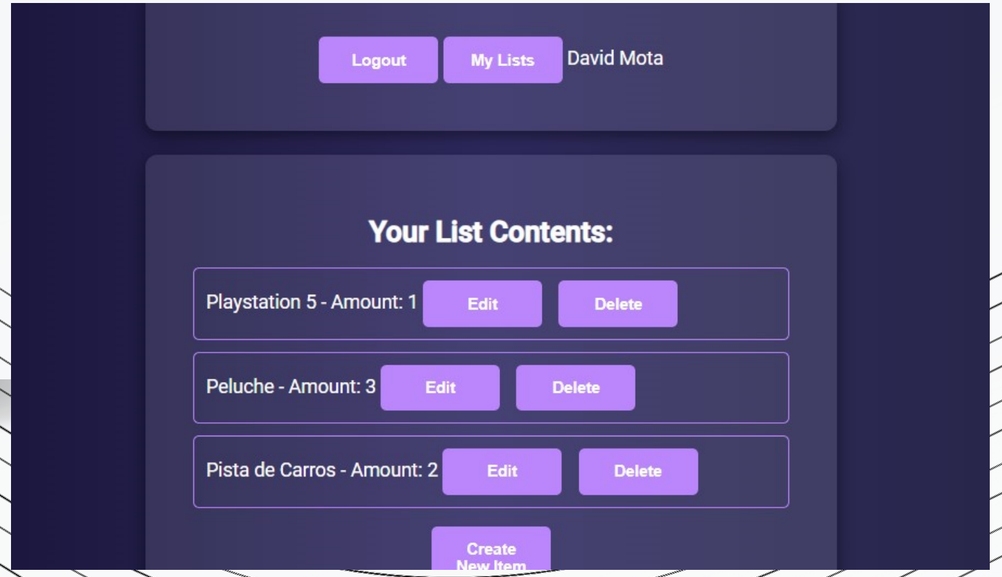
- **Dynamic Pages:** List Contents Page and Available Lists Page adjust themselves to the contents of the database
- **Layout:** Simple but intuitive dark blue theme with buttons for using the app

CLIENT - USER INTERFACE









SERVER/CLOUD

Data Partitioning

- **Consistent Hashing:** Nodes randomly placed on a ring and data items are assigned based on a hashing function
- **Equally-Sized Partitions:** Each node gets an evenly distributed set of tokens. Node leaves or joins => Its tokens are redistributed

- **Adding or Removing Nodes:** Nodes can be freely added and removed from the network
- **Membership Change:** All nodes in the network keep track of one another. Nodes can discover each other, and broadcast their departure

Ring Membership

SERVER/CLOUD

Replication

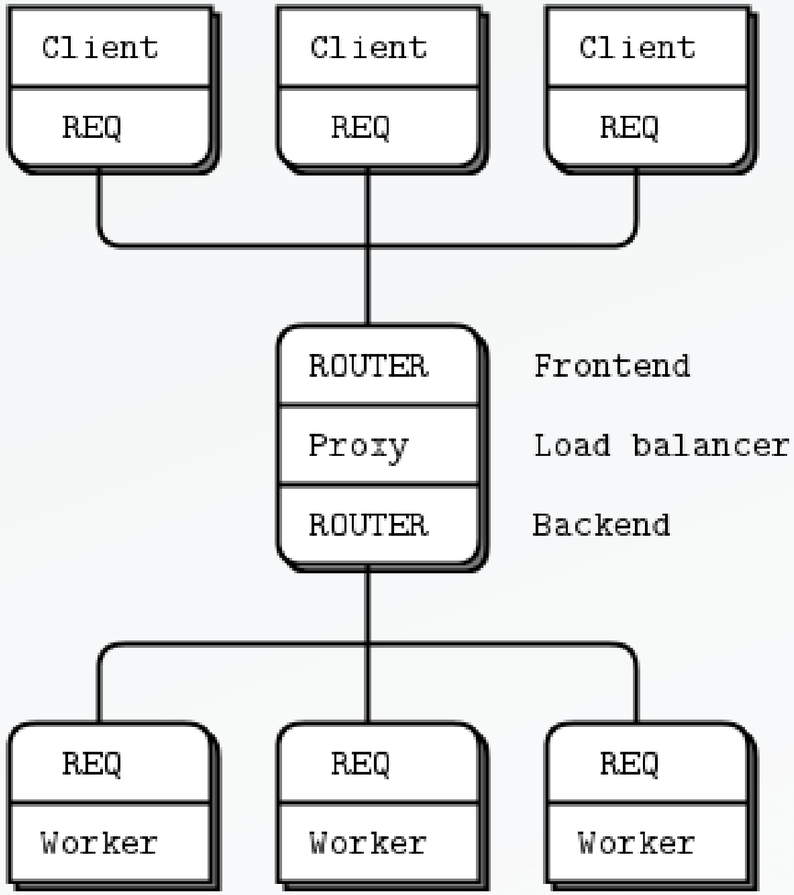
- **Coordinator Nodes:** Responsible for replicating the data items in its range
- **Preference List:** List of nodes that store a particular key. All nodes know which nodes should be in this list for any particular key

- **Node Selection:** Based on load information and done through load balancer. Has to be one of the top nodes in the preference list for the chosen key.
- **In Case of Failure:** Lower nodes in the preference list are acessed

Get and Put operations

LOAD BALANCER

- **Load Balancing Algorithm:**
  - Backend is polled constantly (activity is either a response for a client or a READY message). Worker is stored on queue and, in case of a response, it is sent to the client.
  - Frontend is polled when there are workers available and if activity is found, the last used worker is popped and the client request is sent to the backend



LIMITATIONS

- Replication system is only partly implemented. The preference lists are built for each key, and requests are routed to the first avaliable node in them, however no replicas are sent to other nodes beyond the coordinator.
  - This also means that when performing reads and writes, the data is only retrieved and saved in the coordinator node’s local storage.

- Other Dynamo features such as hinted handoffs and replica synchronization weren’t implemented.

SOURCES

- **Amazon Dynamo Paper** - https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf
- **Zyre Framework** - https://zguide.zeromq.org/docs/chapter8/
- **“Local first” resources** - https://localfirstweb.dev/