U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# Local-First Shopping List Application

**SDLE 2023/2024**

**1MEIC03**

Alexandre Nunes - up202005358
André Sousa - up202005277
Gonçalo Pinto - up202004907
Pedro Fonseca - up202008307

# Main Design Challenges

This slide deck is supportive material for the local-first shopping list application developed. In the next few slides, we will mention the main design challenges that appeared and how we decided to solve them.

# Client

Authentication
Interface

# CLIENT AUTHENTICATION

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- In order to allow user registration and login, an **Authentication Server** was created.

- This server is nothing more than a **ZMQ Socket** listening in a certain port with a SQLite database storing all the authentication information.

- Upon a successful login, it issues **JSON Web Tokens (JWT's)** as "citizenship cards" that will be used in all future list management operations.

# CLIENT INTERFACE

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- The application was built with a **web interface**.

- Both frontend and backend were built using the **Flask** framework.

- Offline shopping list operations are supported through **local JSON file caching**.

- **Periodic attempts** are made to **synchronize** local data with the remote server, ensuring that users can seamlessly update shopping lists online and offline.

# Request Routing

# REQUEST ROUTING

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- **Fault tolerance, scalability, and availability** are achieved through a **request routing system**.

- The system employs the **REQ REP pattern** with **ZeroMQ sockets**, along with a **ZeroMQ proxy** server. This approach allows for **dynamic** addition and removal of **servers** without system reboots.

- The **ZMQ proxy** server also implements **load balancing** to prevent overloading individual servers **(Dynamic DNS Simulation)**. This setup promotes system resilience and minimizes downtime since the crash of servers is opaque to the user.

# Databases

Technology
Replication
Load Balancing

# DATABASE TECHNOLOGY

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

- Traditionally production systems store their state in relational databases. However, a **relational database** is a solution that is **far from ideal**. Our application only needs to store and **retrieve data by a unique ID** and does not require the complex querying and management functionality offered by an RDBMS.

- Inspired by Amazon's Dynamo, we used **BerkeleyDB** which has a **simple key/value interface**, is highly available with a clearly defined consistency window, is efficient in its resource usage, and has a simple scale-out scheme to address growth in data set size or request rates.

# DATABASE REPLICATION

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- In pursuit of robust **fault tolerance**, the system incorporates a **replication mechanism**, whereby **lists assigned** to a **particular database** are **replicated** in the next **X database instances**, being X a configurable parameter (in the current implementation we are using X=2).

- When a client attempts to **access a list**, the system retrieves it from the designated databases, that are **active** at the time. The resulting list is determined by the **merge** of the lists provided by the databases. This is achieved through the utilization of a **CRDT merge function**.

- Upon **consensus**, the finalized outcome will be stored in each assigned active database. This redundancy serves to promote **data integrity** and **uninterrupted operations**, even in the event of hardware failures or network disruptions.

# DATABASE LOAD BALANCING

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Inspired by **Dynamo's architectural principles**, our system implements a **hashing ring**. This ring effectively subdivides the entire spectrum of potential list identification numbers and assigns a **dedicated database** instance to **each segment**.

- The list's **identification number** corresponds to the application of the **MD5** consistent hashing algorithm (as described in Dynamo) to the concatenation of the list's name and email of the user who created it.

- To address potential **load imbalances** and further enhance **system efficiency**, **virtual nodes** are introduced into the hashing ring, thereby refining the division of responsibilities among database instances.

# Database Disruption

# Database Disruption

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- BerkeleyDB databases are essentially local files that are mapped into the process memory space, therefore, to test database failures, we had to create another process that performs logical enabling and disabling of database instances.
- This process allows 4 actions:
  - **list**: Lists the current existing DBs. Shows their IDs, current state (ONLINE, OFFLINE), number of requests, and number of shopping lists.
  - **enable/disable [id]**: Enables/disables a database. Receives an argument ID which represents the ID of the database.
  - **printlists**: Prints all the lists in the databases.

# CRDTs

# CRDTs (1/2)

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

- In the context of our shopping list application, **CRDTs** are used to **synchronize** the **shopping lists** across different databases.

- Each **operation** on a shopping list, be it adding items, buying items, removing items, or renaming items is **logged** in chronological order.

- Those logs capture every change made to the list, starting from its initial state upon creation.

- Operations done in **different timezones** are supported by converting all timestamps to UTC.

# CRDTs (2/2)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

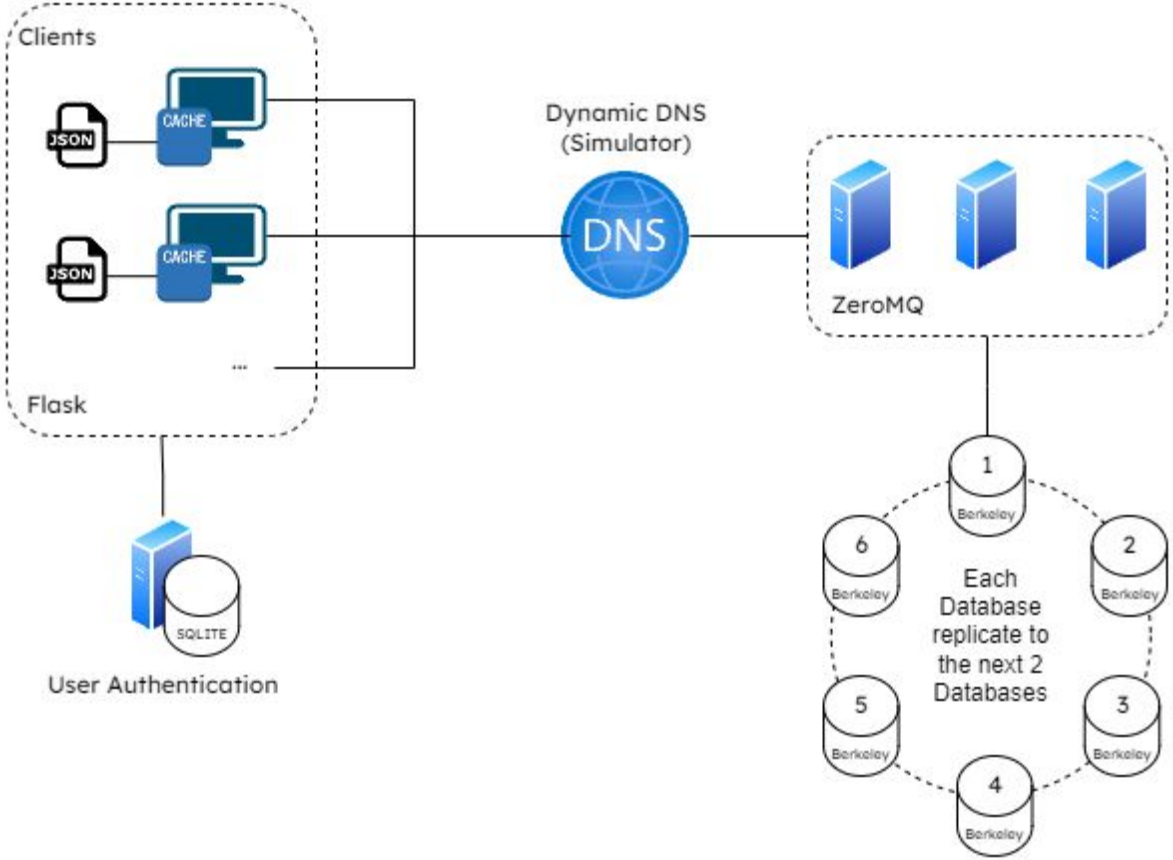
- Each **log** follows a **JSON format** with the following attributes:
  - type of change
  - change
  - timestamp

- In the face of divergent logs, the merge function will sort the logs chronologically and use them to build the final version of the list.

- The **time** factor of the logs is **only** used to support **item renaming**, having no impact on the other types of list operations.

- However, the timestamp is essential for the merge function to identify common logs in the different databases, working like a unique tag. All the log attributes together serve as an identifier.

# HIGH LEVEL ARCHITECTURE


Clients
JSON
CACHE
JSON
CACHE
...
Flask
Dynamic DNS
(Simulator)
DNS
ZeroMQ
SQLITE
User Authentication
1
Berkeley
2
Berkeley
3
Berkeley
4
Berkeley
5
Berkeley
6
Berkeley
Each
Database
replicate to
the next 2
Databases