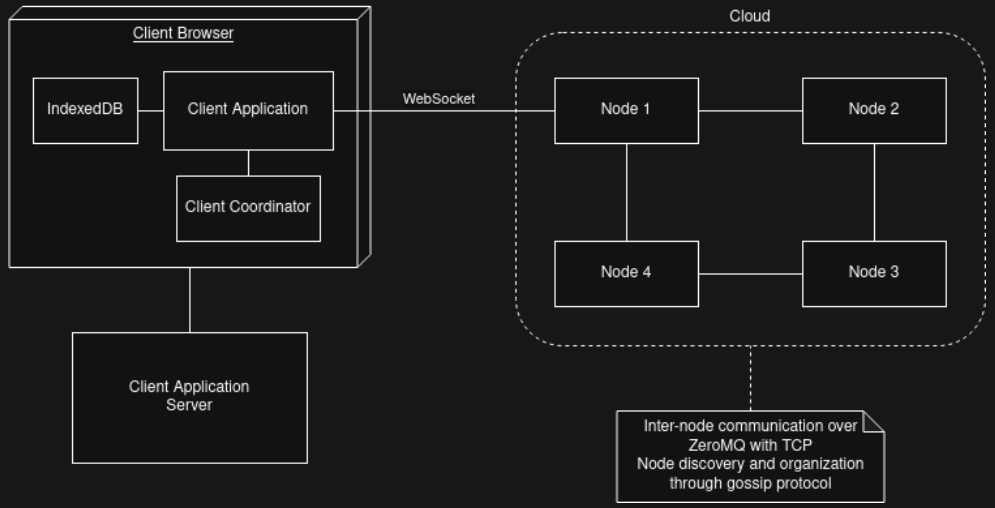
Shopping Lists on the Cloud

SDLE 2025/26

Bruno Oliveira - up202208700
Dário Guimarães - up202502543
Henrique Fernandes - up202204988

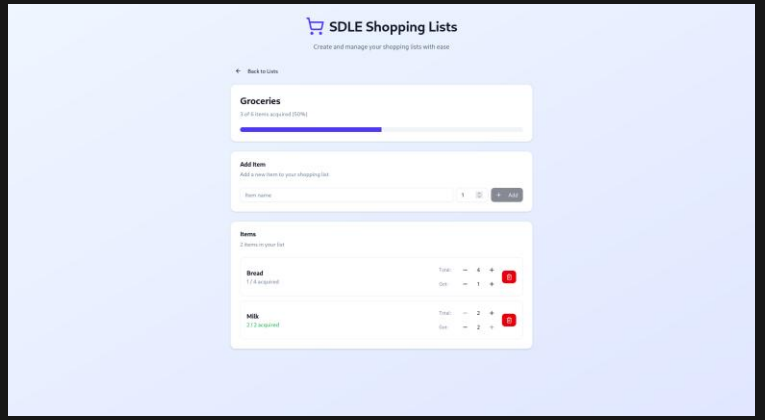
Architecture



Client Application

- Built on **TypeScript + React + Next.js**
- **Single Page Application (SPA)** - files served by an external server
- Uses **IndexedDB** for persistent storage

- Follows **local-first design principles**
  - Users can make changes locally, without confirmation from the server
  - Application syncs with the cloud only when possible, i.e. **network is optional**



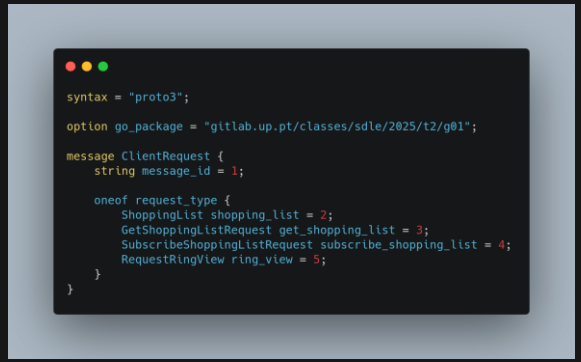
Communication

- Between cloud nodes => **ZeroMQ**
  - Simplifies common communication strategies using predefined messaging patterns
  - High performance and scalability

- Between client and cloud => **WebSocket**
  - Real-time, two way communication
  - Natively supported by most browsers (unlike ZeroMQ)

Message Serialization

- **Protocol Buffers**
  - Efficient, language-neutral, platform-neutral extensible mechanism for serializing structured data.
  - Smaller messages (binary format)
  - Faster serialization
  - Type safety (using DDL)

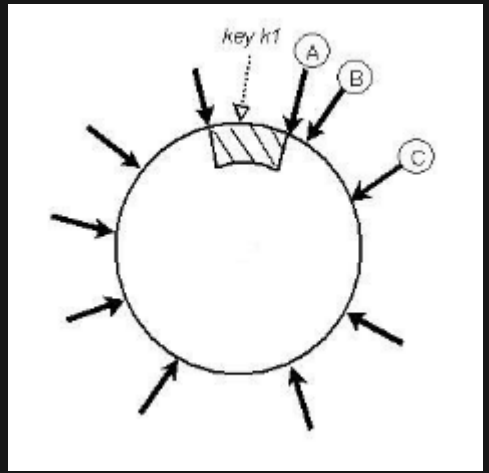


```
syntax = "proto3";

option go_package = "gitlab.up.pt/classes/sdle/2025/t2/g01";

message ClientRequest {
    string message_id = 1;

    oneof request_type {
        ShoppingList shopping_list = 2;
        GetShoppingListRequest get_shopping_list = 3;
        SubscribeShoppingListRequest subscribe_shopping_list = 4;
        RequestRingView ring_view = 5;
    }
}
```

Protobuf DDL syntax

Consistent Hashing

- Defines multiple hashes, each assigned to a node
- Node is responsible for storage and management of all the keys between its hash and its predecessor's
- Promotes **uniform load distribution** by assigning multiple hashes to each node (**virtual nodes**)
- When a node is added, key ranges in its responsibility are transferred to it



DeCandia et al. (2007)

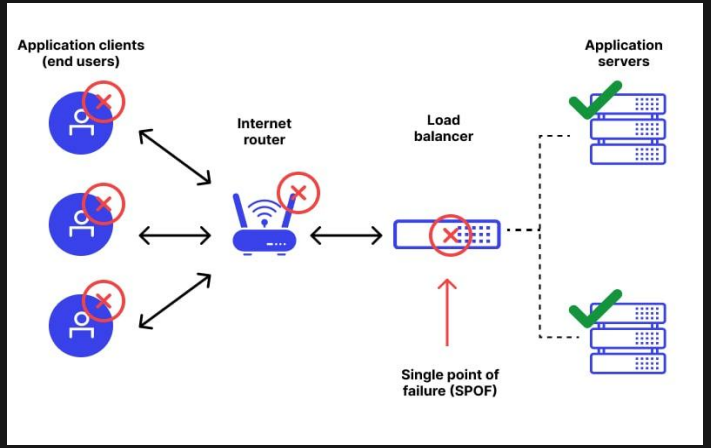
Replication

- Key Parameters
  - **N = 3**: Replication factor (data stored on 3 nodes)
  - **W = 2**: Write quorum (success after 2 writes)
  - **R = 2**: Read quorum (success after 2 reads)
- Coordinator Selection
  - Based on **preference list** (first N nodes from consistent hashing)
  - **First alive node** in preference list becomes coordinator
  - Handles client forwarding automatically (fallback mechanism in case the client coordination fails)
- Sloppy Quorum & Hinted Handoffs
  - In case a node in the preference list is unavailable, the next node after the preference list is used
  - Every 10 seconds, that node tries to transfer the data to the intended node
  - Ensures **eventual consistency** despite node failures

Why Not Use a Proxy?

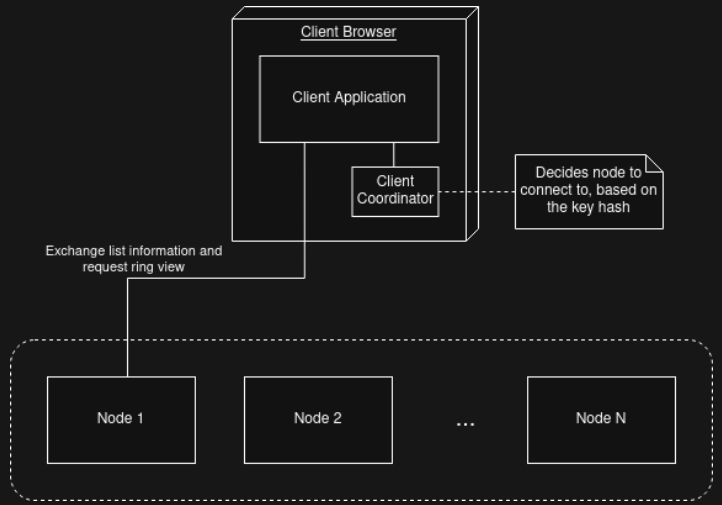
- No single point of failure (almost)
- Symmetric node architecture

- Still requires application to know the address of at least one node (we use **seed nodes**)
- No load balancer



Client-Driven Coordination

- Node periodically fetches ring view information from a known node
- Based on that information, determines the coordinator for a key and communicates directly with it



Client-Driven Coordination

- Avoids extra node hop introduced by load balancer => **Less network latency**
- **No load balancer required:** fair load distribution is implicitly guaranteed by a near uniform distribution of keys across nodes

- Requires ring view to be known by the client
- Between updates, membership information can become stale

Table 2: Performance of client-driven and server-driven coordination approaches.

	99.9th percentile read latency (ms)	99.9th percentile write latency (ms)	Average read latency (ms)	Average write latency (ms)
Server-driven	68.9	68.5	3.9	4.02
Client-driven	30.4	30.4	1.55	1.9

DeCandia et al. (2007)

Conflict-Free Replicated Data Types

DotContext

Tracks dots seen by a replica

DotKernel

Auxiliary for implementing causal based CRDTs

CCounter

Causal counter, supporting increments and decrements

MVRegister

Multi-value register, mainly for string values

DWFlag

Disable-wins flag

ORMap

Association between keys and causal CRDTs

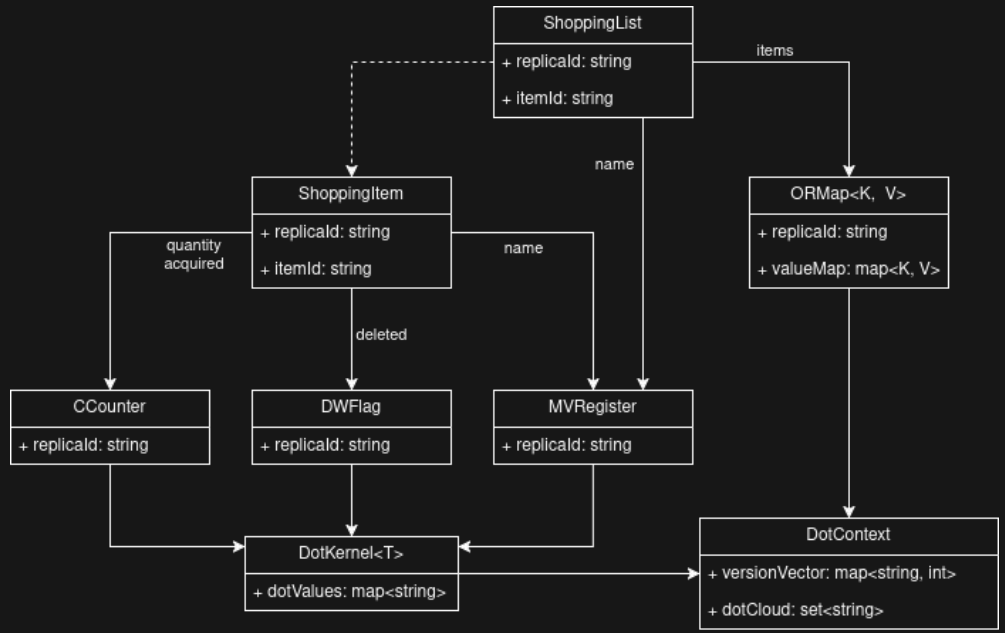
ShoppingItem

CRDT implementation of an item in a shopping list

ShoppingList

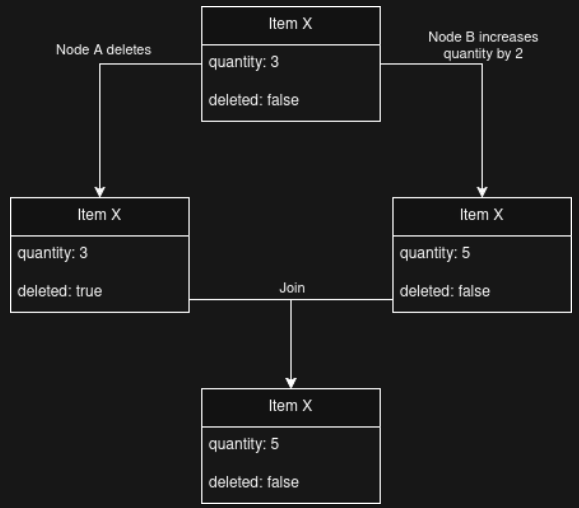
CRDT implementation of a shopping list

Conflict-Free Replicated Data Types



Conflict-Free Replicated Data Types

- Implemented CRDTs support **deltas**!
- Strategy for ensuring add-wins semantic in lists
  - Use of soft-deletes (with DWFlag)
  - Robust and simple
  - Nodes are never effectively removed



References

- De Candia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007). *Dynamo: Amazon's highly available key-value store*. Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP), 205–220.