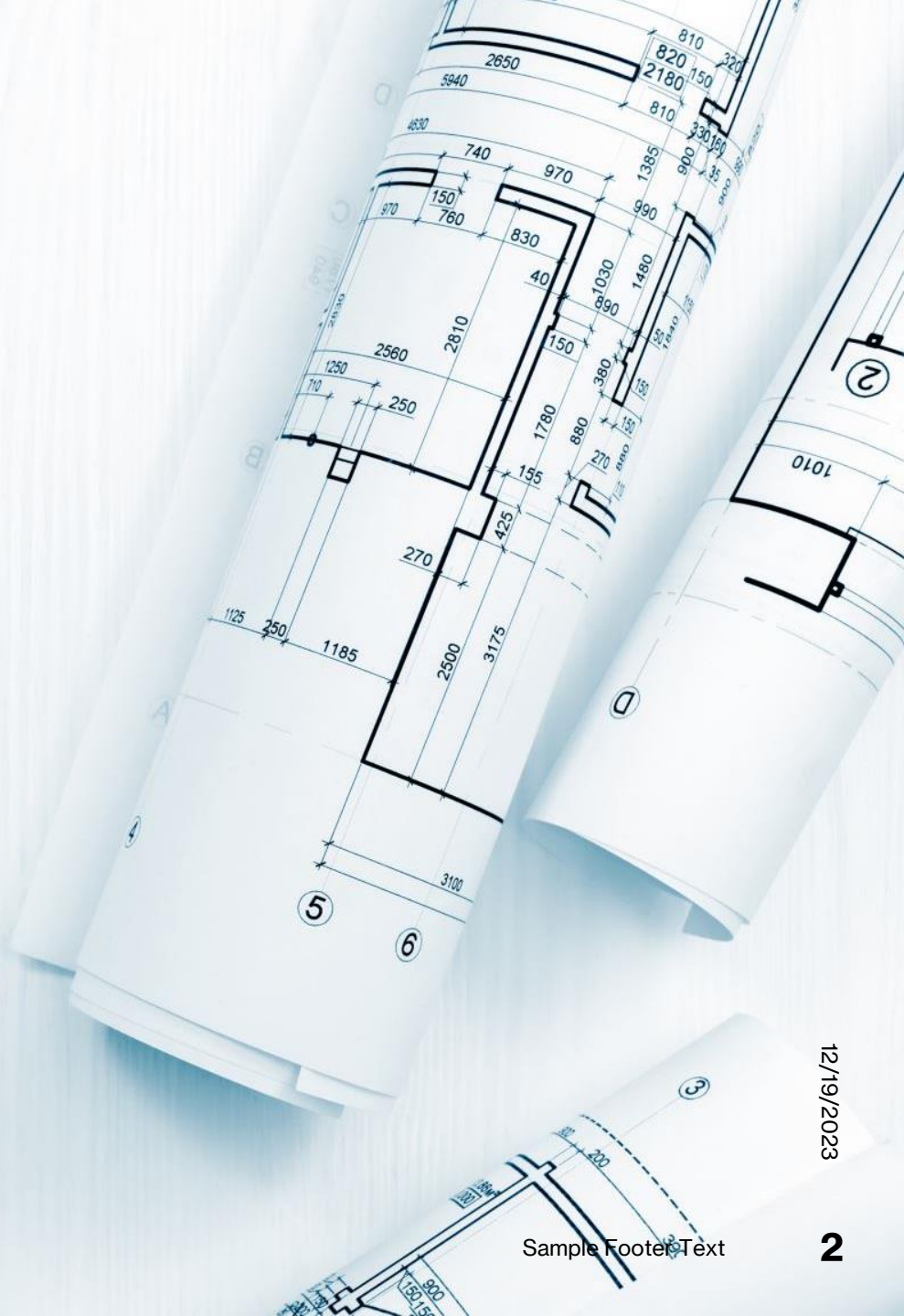
RushCart: Cloud-Based Shopping Lists under Pressure 2023/2024

Large Scale Distributed Systems (SDLE) - M.EIC004

Pedro Balazeiro
up202005097
Rúben Viana
up202005108

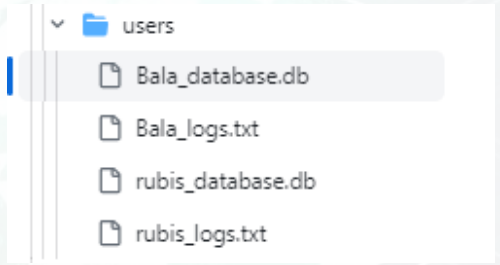
Requirements

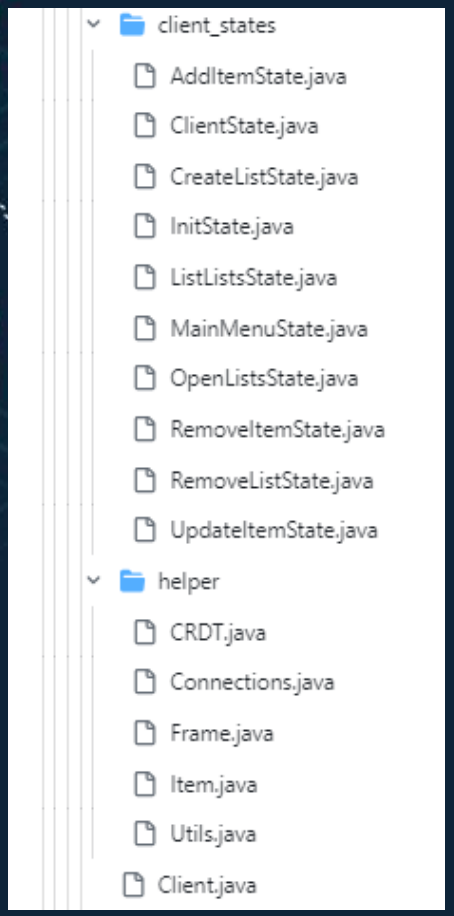
- Creation of a local-first shopping list application
- Unique ID lists shared among users
- Managing Lists: Creating, Reading, Updating, and Deleting Entries
- CRDT based lists
- Architecting for Massive Scale: Serving Millions of Users
- Data sharding



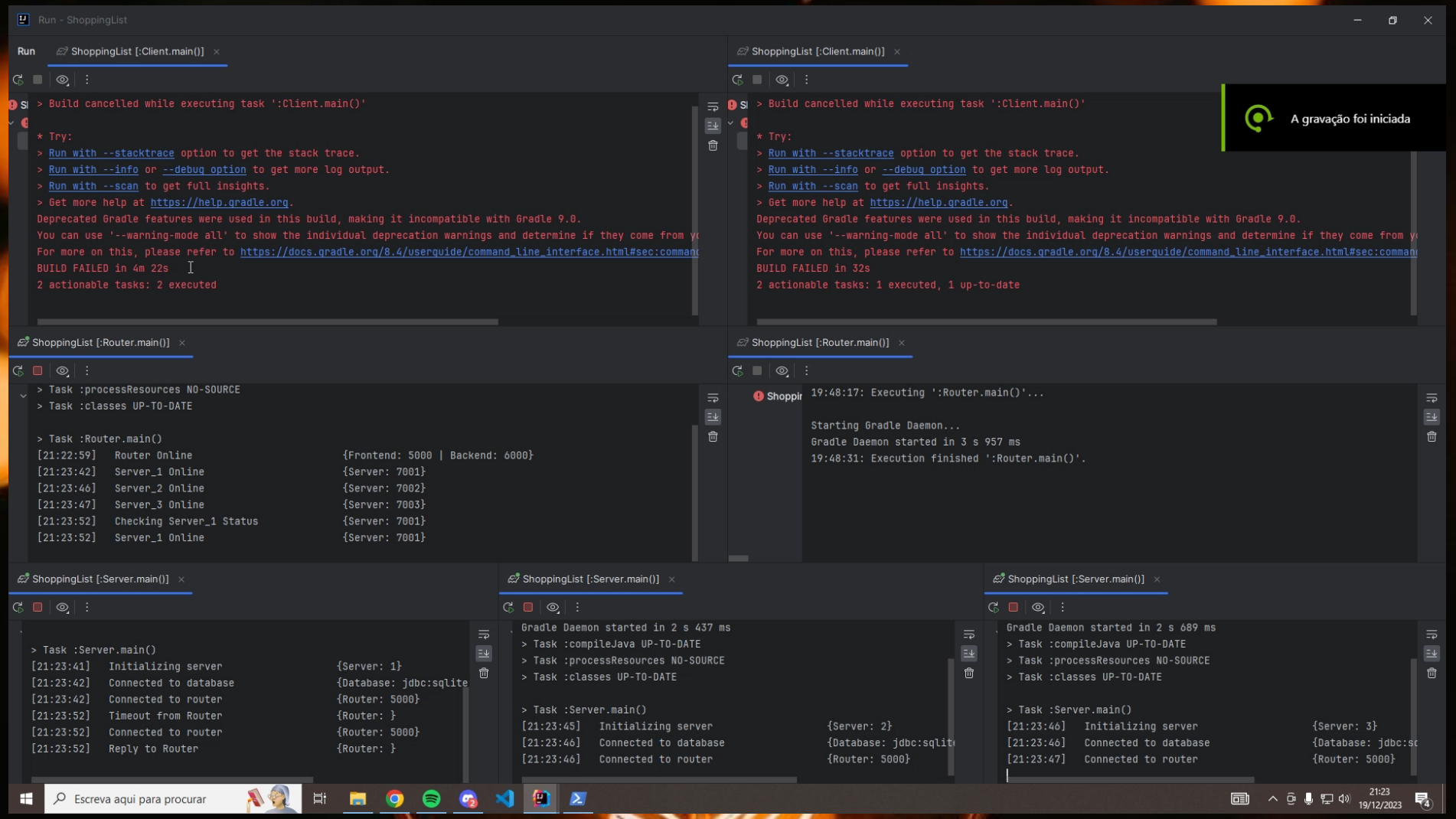
Architectural componentes - client

- State based program
- Sqlite database for each user
- Local first approach
- CLI
- The synchronization with the cloud occurs at regular intervals (every 5 seconds).
- CRDT – LWW



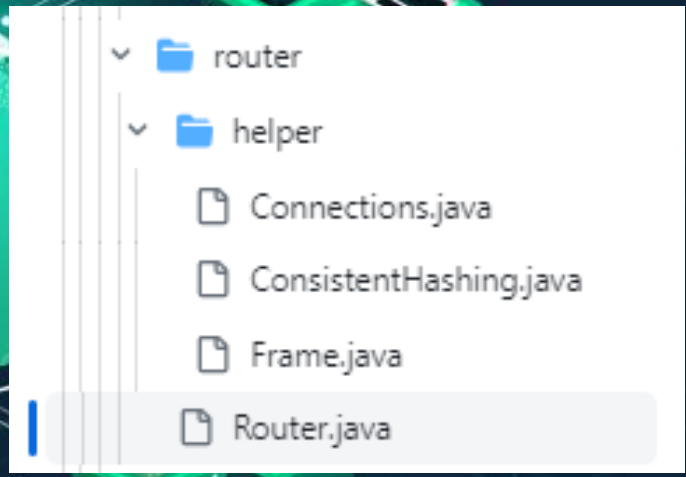


Video Demo – sync with cloud

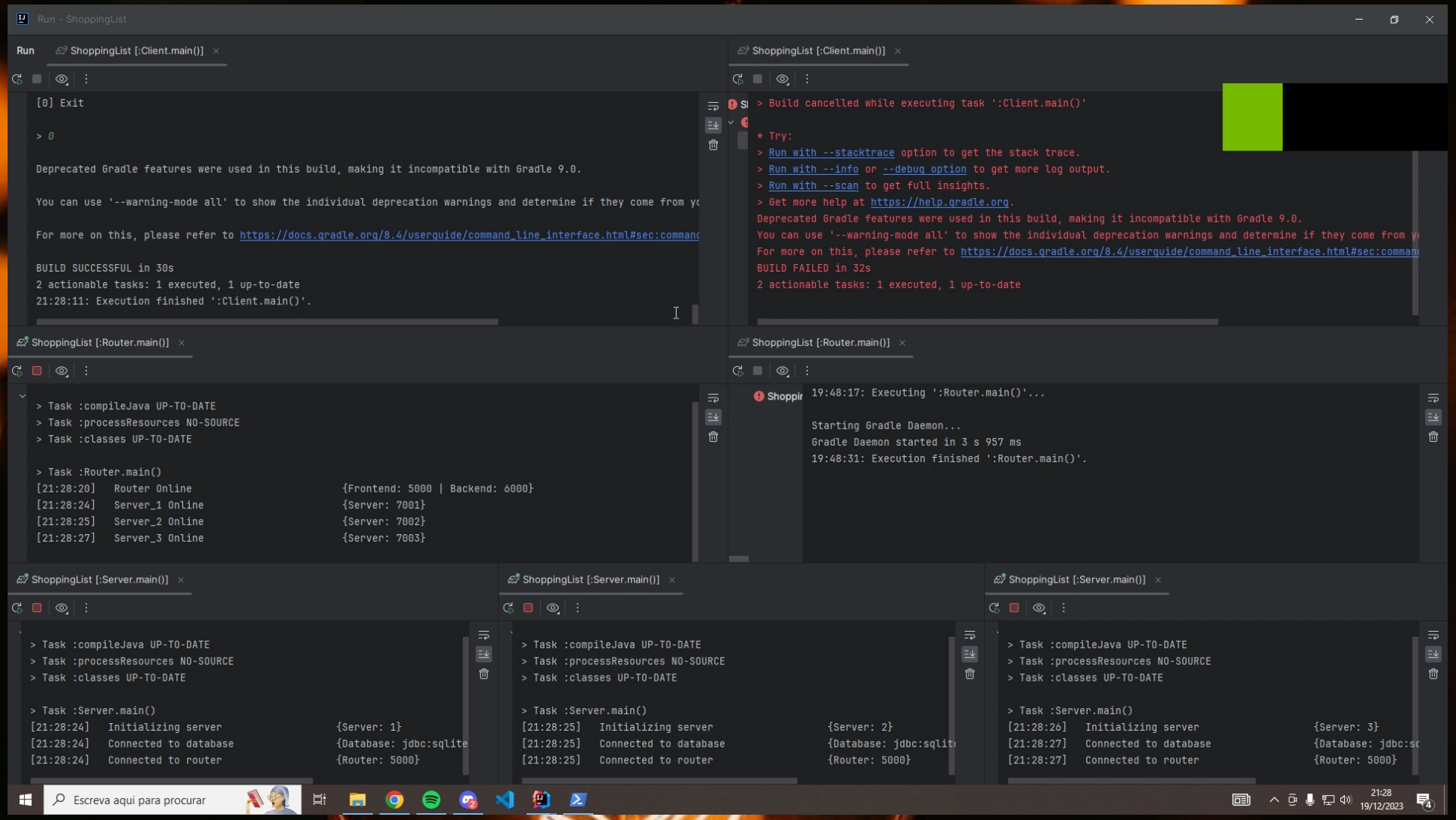


Architectural componentes - router

- Developed utilizing ZMQ
- A mediator connecting users' devices to server systems
- Number of routers 2
- Sharding is done using the hash ring
- Router sends heart beats to servers to check if they are online

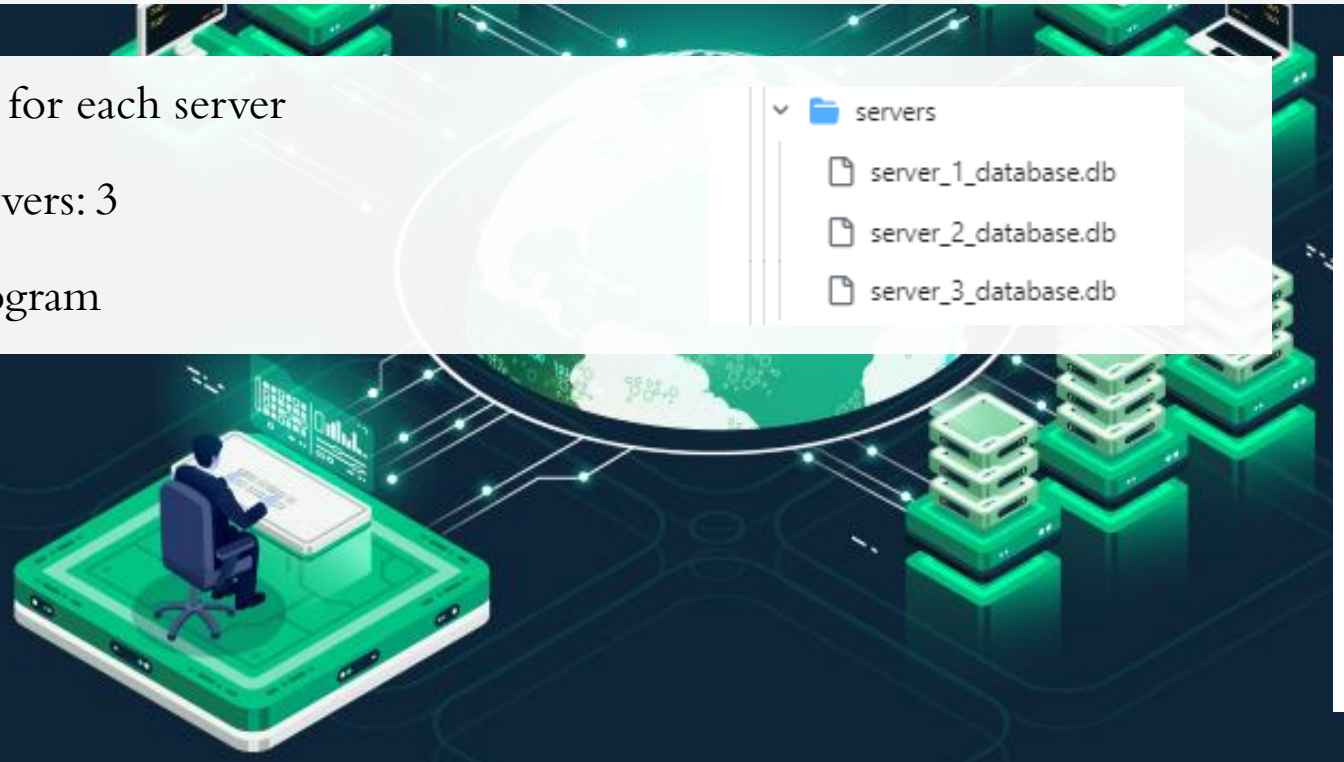


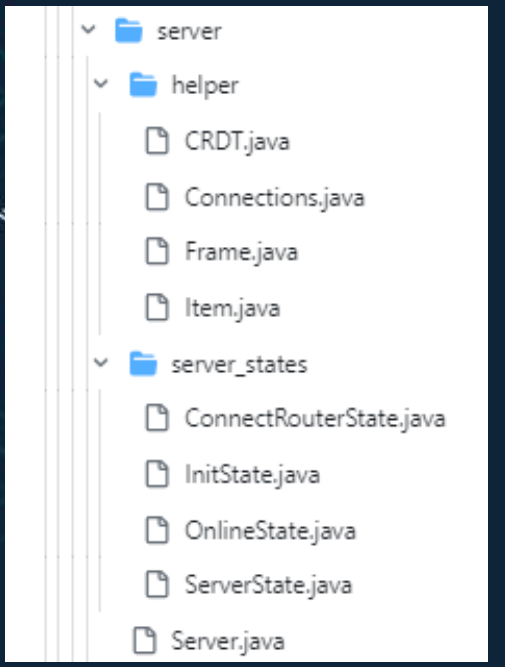
Video Demo- Sharding

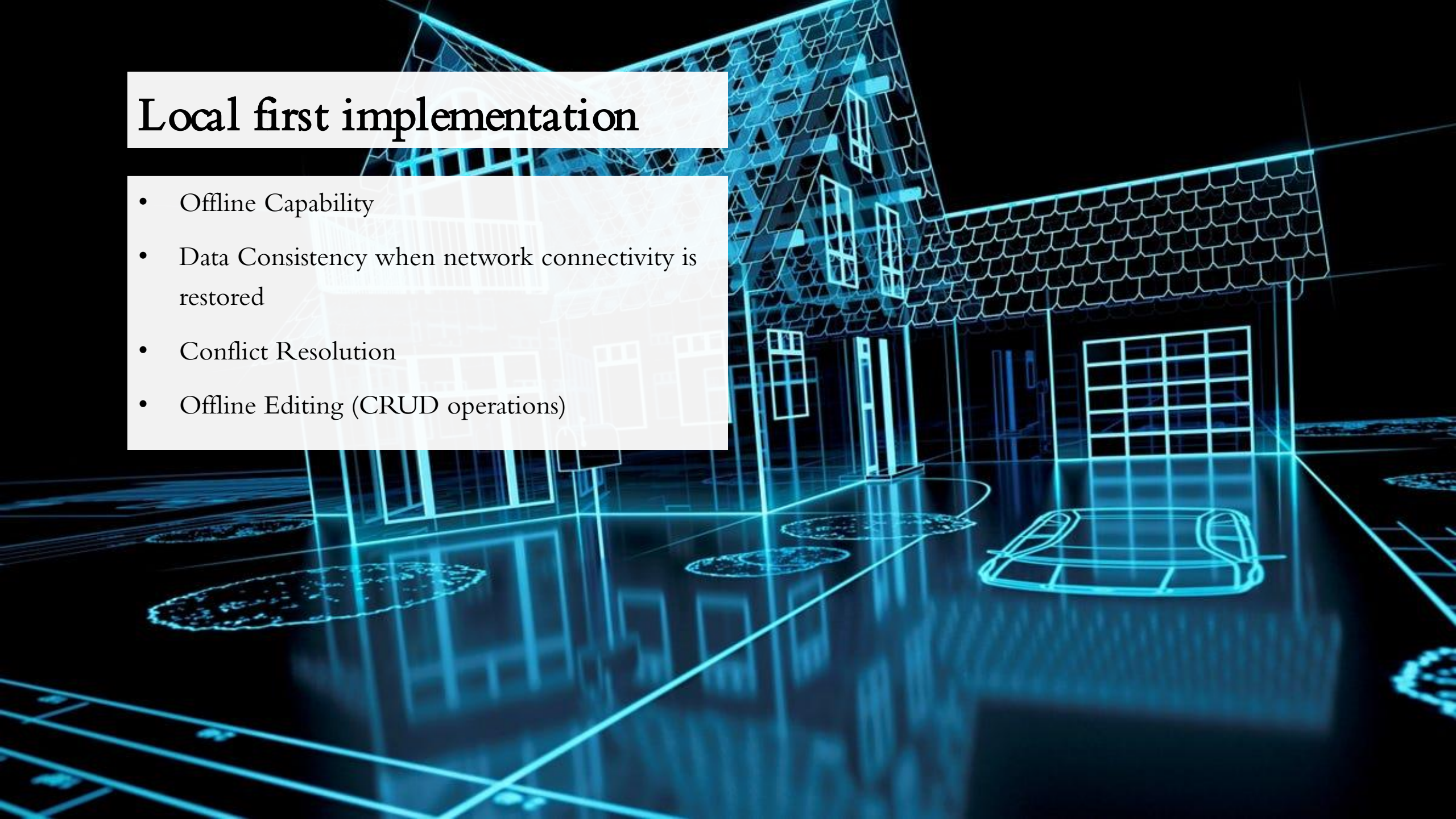


Architectural componentes - server

- Sqlite database for each server
- Number of servers: 3
- State based program



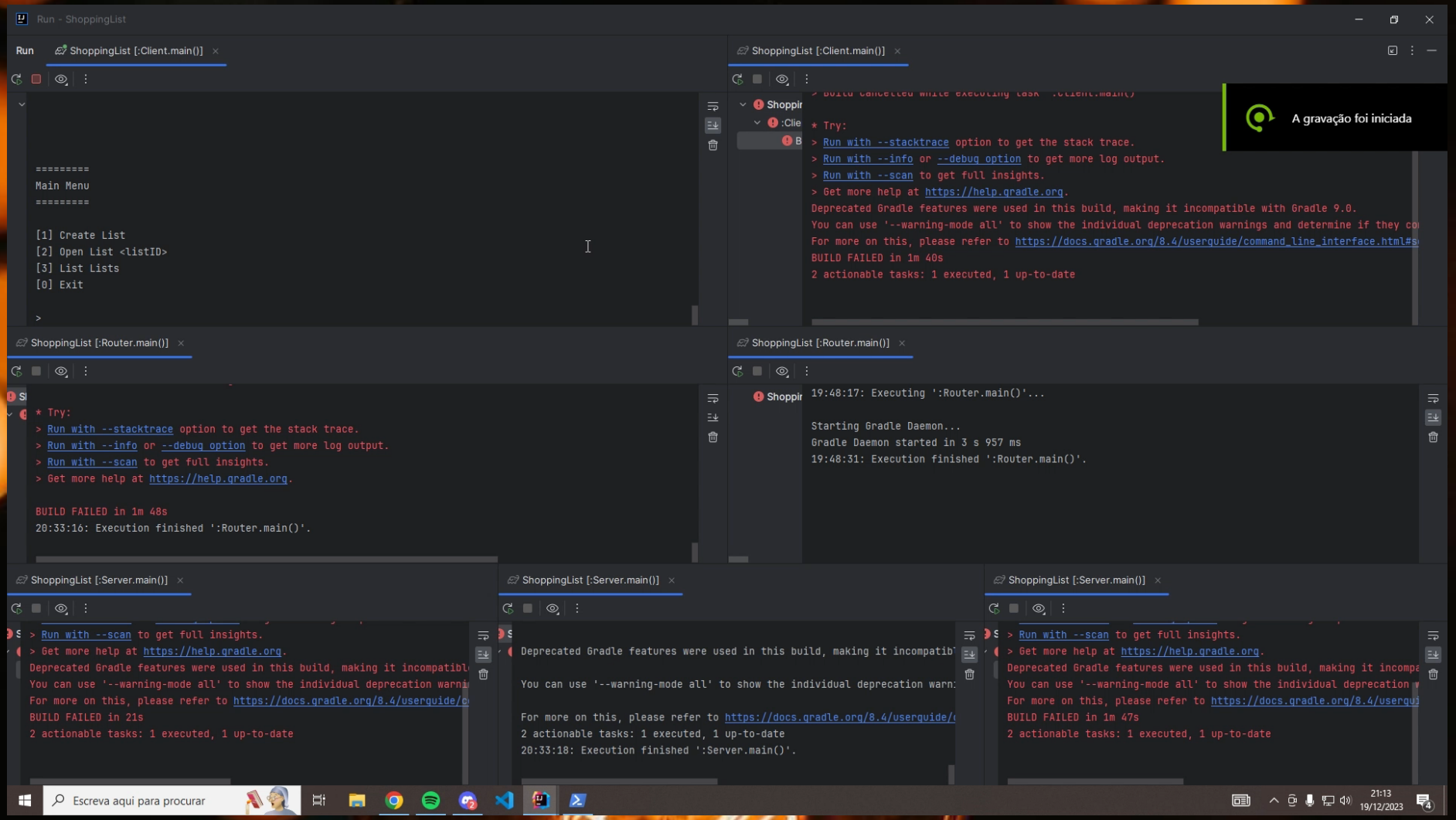




Local first implementation

- Offline Capability
- Data Consistency when network connectivity is restored
- Conflict Resolution
- Offline Editing (CRUD operations)

Video Demo – Local first

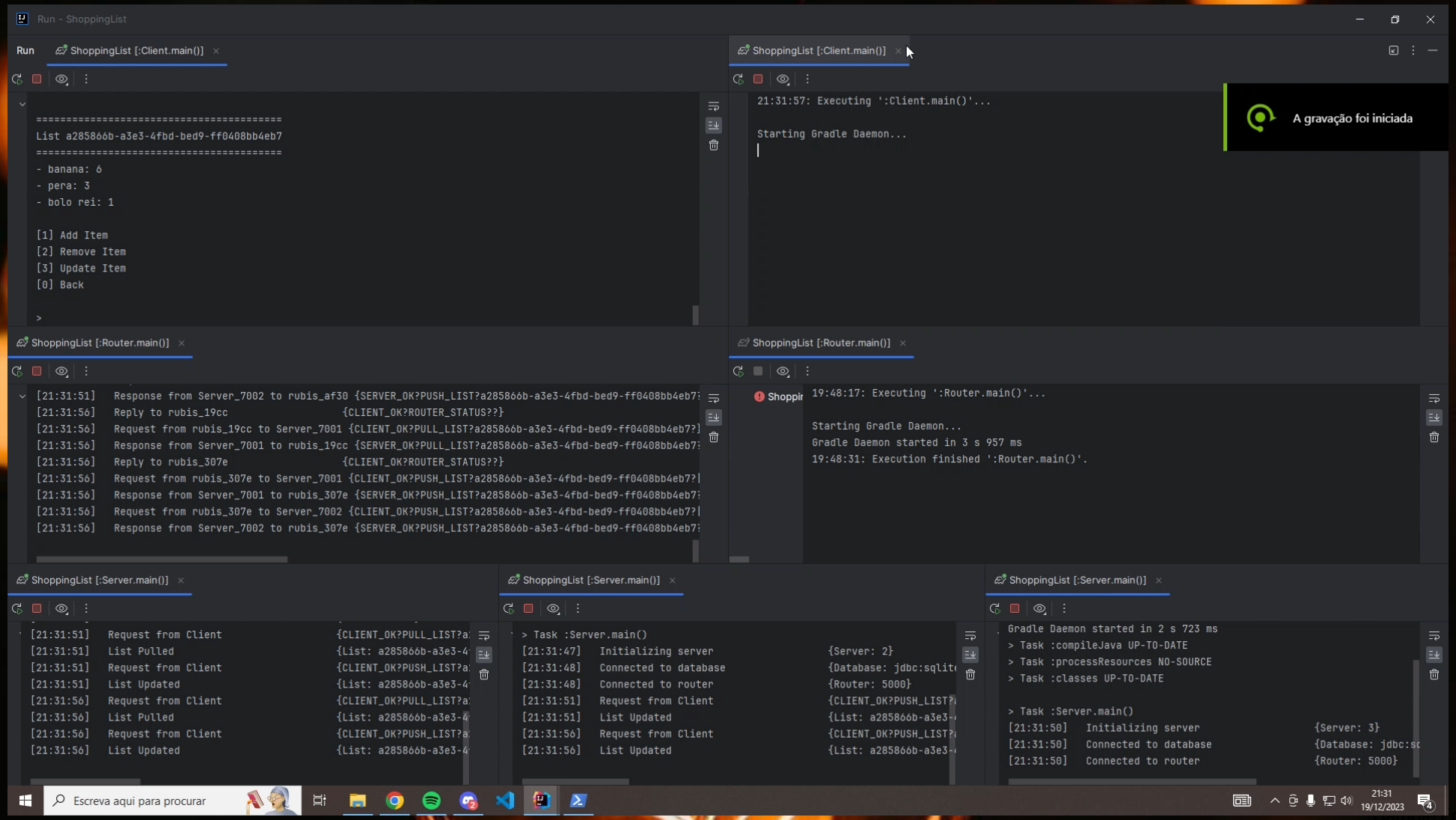


CRDT

- Simple version
- Uses timestamp for merge
- Add, remove, update and merge operations

```java
public class CRDT {
    private Map<String, Item> items = new HashMap<>();
    public CRDT(String itemsList) {
        JsonArray jsonArray = new Gson().fromJson(itemsList, JsonArray.class);
        for (JsonElement item : jsonArray) {
            Item newItem = new Gson().fromJson(item, Item.class);
            items.put(newItem.name, newItem);
        }
    }

    public void addOrUpdateItem(Item newItem) {
        items.put(newItem.name, newItem);
    }

    // Removing an Item is equivalent to setting its quantity to 0
    public void removeItem(Item removedItem) {
        Item newItem = new Item(removedItem.name, 0, removedItem.timestamp);
        items.put(newItem.name, newItem);
    }

    // Method to merge two lists based on timestamps
    public void merge(String otherItemList) {
        JsonArray otherJsonArray = new Gson().fromJson(otherItemList, JsonArray.class);
        for (JsonElement otherItem : otherJsonArray) {
            Item item = new Gson().fromJson(otherItem, Item.class);
            items.merge(item.name, item, (existingItem, newItem) ->
                    (existingItem.timestamp >= newItem.timestamp) ? existingItem : newItem
            );
        }
    }

    // Method to get the current state as a JSON string
    public String getItemsList() {
        return new Gson().toJson(items.values());
    }

    public Collection<Item> getItems() {
        return items.values();
    }

    public Item getItem(String name) {
        return items.get(name);
    }
}
```

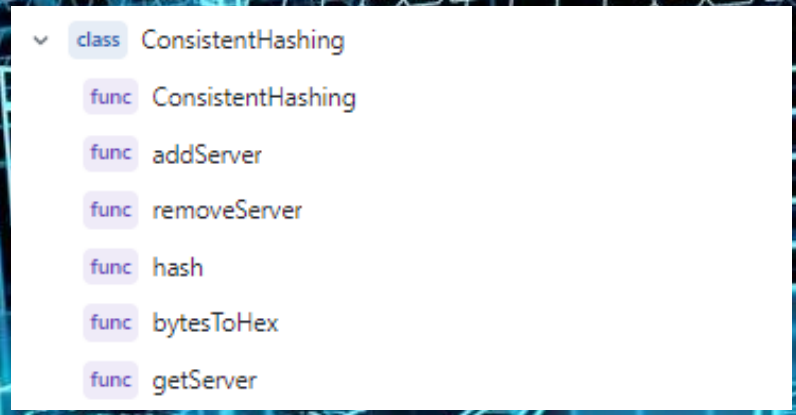
Video Demo- Conflict resolution



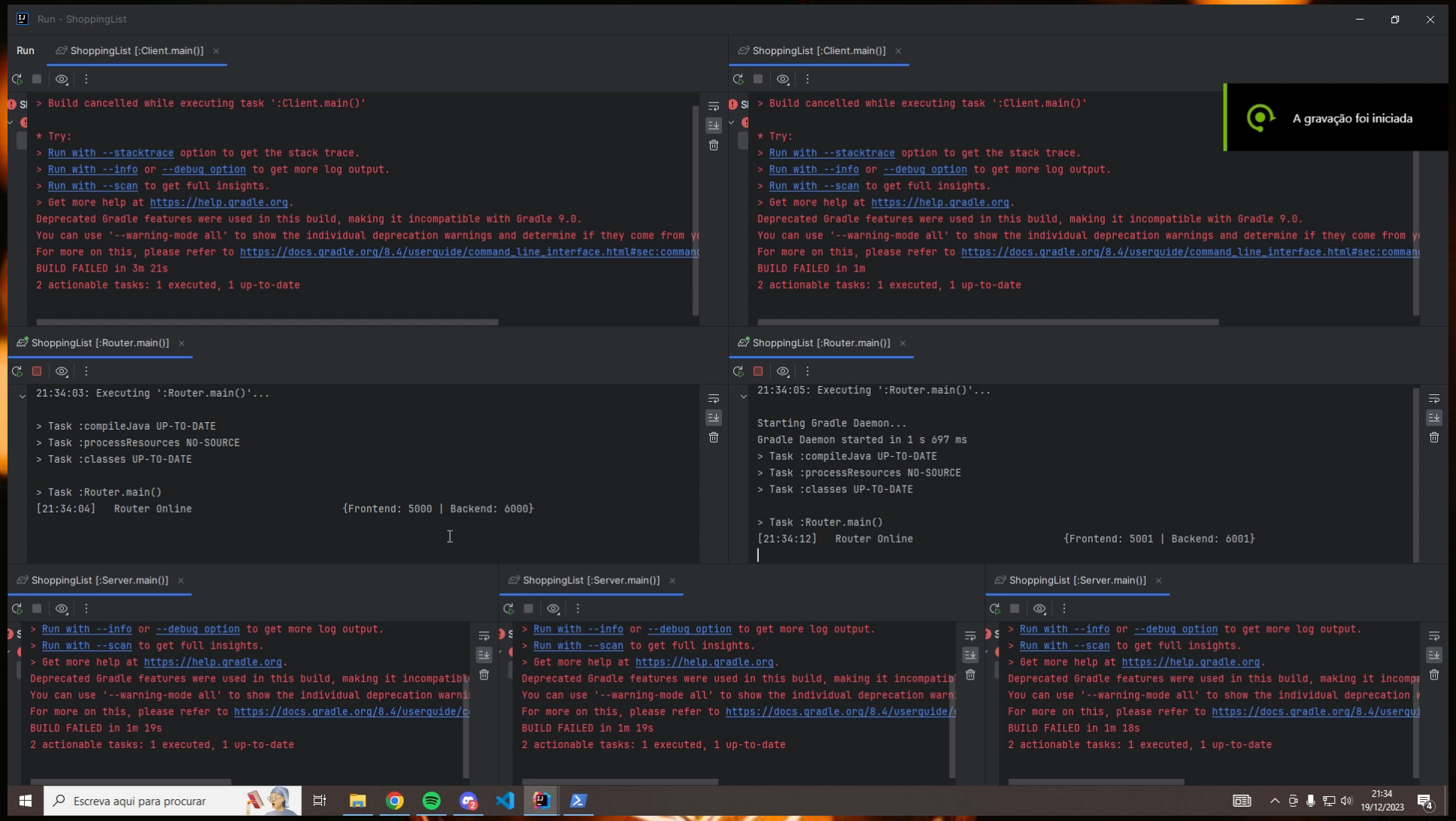
Consistent hashing

- Number of virtual nodes per server: 3
- Used sha256

```java
public class ConsistentHashing {
    private final int numberOfReplicas; // Number of virtual nodes per server
    private final SortedMap<String, Integer> hashRing = new TreeMap<>(); // Hash ring
    private final MessageDigest hashFunction;
```



Video Demo – reconnection router



Conclusions and future work

- Upgrade CRDT
- Communication between servers
- Improvement of hash ring
- Hinted Handoff