Building a Retail Sysem Based on Distributed Databases Model

Phan Thi Ha, Trinh Thi Van Anh

Abstract. In recent years, distributed databases have become an important field of processing information, overcoming some limitations of centralized database such as overloading, bottlenecking while accessing, availability/ reliability of low fault tolerance. Our article proposes to build a distributed system (functions and databases) for POS (point of sale) retailers, data will be distributed across different locations but can still be linked together when required. At each location retailers can sell directly on the system (online or offline) so they can administer local databases and execute their local applications (business). Here we deploy the system on the distributed database management system based on Client-Server model. Therefore, aside from local management of data at clients (POS), there is also a server (manager) that stores data, manages and controls the entire system.

Key word: reactjs, Progressive Web App, Magento, commerce, Distributed Databases, Distributed System.

I. INTRODUCTION

In recent years, e-commerce activities are being developed rapidly. A survey about sale management shows that developing an online sales system is essential. However, in reality, retailing is still the most popular form of business. In the Vietnamese market today, activities in stores and chain stores in providing products to consumers still take place directly through software installed at each store.

The purpose of this article is to analyze, design a system that meets both online shopping demands and shopping at store demands, synchronize data across the system. To support the construction of the system, we integrate new technology Magento[2], ReactJS[3], Progressive Web App (PWA)[4] into the system to help: fast and smooth application, run smoothly and stably, work with or without connection to the server. In addition, we also use IndexedDB to create web applications which have superb query capabilities with large data and can work both online and offline. The layout of the article is as follows in addition to part I, part II is system analysis and design and part III is system implementation and testing. Finally, the article concludes with an overall review and the application of our system in the future.

II. SYSTEM ANALYSIS AND DESIGN

In order to help the managers and salespeople manage time efficiently, improve availability and accuracy of work, we build a distributed system for retailer as follows: At each point of sale, POS (client) allows: searching products; manage products by category; scan barcode; add product to cart; add, edit, delete and search customer; add discount code to shopping cart; create and pay the bill; print invoice; synchronize data with server.

Revised Manuscript Received on August 05, 2019

Phan Thi Ha, Department Of Information TechnologyPosts and Telecommunications Institute of TechnologyHanoi, Vietnam

Trinh Thi Van Anh, Computing Fandamental Dapartment, FPT University, Hanoi 10000, Vietnam

Creat a payment bill even without internet connection and synchronizing as soon as it is reconnected, at each POS, the software can operate on multiple platforms, on computers, mobile devices, etc., to meet user needs.

At managing location (Server) allows: handle requests from POS; manage employees; manage roles; manage role-assigned employees; manage POS; manage Location; manage setting for applications.

2.1Overall structure of the system

Figure [1] describes the entire structure of the system in which POSi ((i = 1 ÷ n) is the sale of point at each area (Location). Each point of sale will have a device (computer or tablet) handling the sales of the system. Data processed at each POS will be stored locally and synchronized to the server once connected to the internet. The application (Retailer POS) we build will serve salespeople and is also a web app on client that interacts directly with users

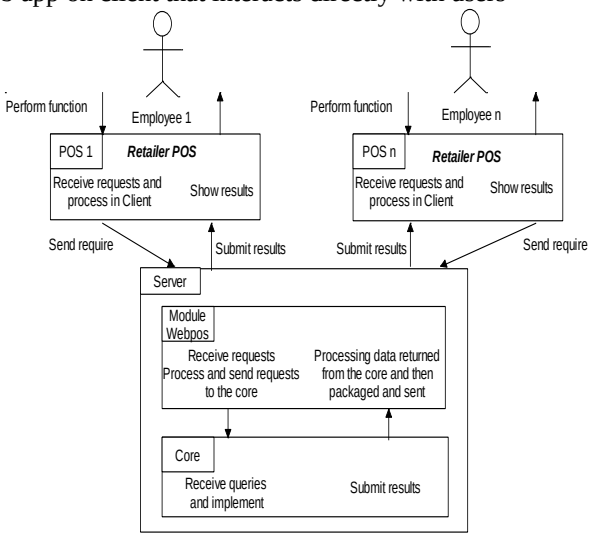

Figure 1. Overall structure of the system

Manager Position (Server) will store all data of the system and do tasks such as analyze statistics and manage. Parts included are: core components for logical processing, core functions of the system; components of the module web POS: a backbend module that handles requests from client for the manage to manage the entire POS system. Functions performed include: handle requests from client, manage employees, manage roles, manage POS, manage locations, manage application setting.

2.2 Distributed database model of the system

With different managing model at different geographical location, to secure data and speed up the work locally at each point of sale, we build distributed database model, figure [2], installing database managing system based on Client/Server structure.

The system uses relational SQL database to store main server database. In Client, Retailer POS application uses IndexedDB as database.

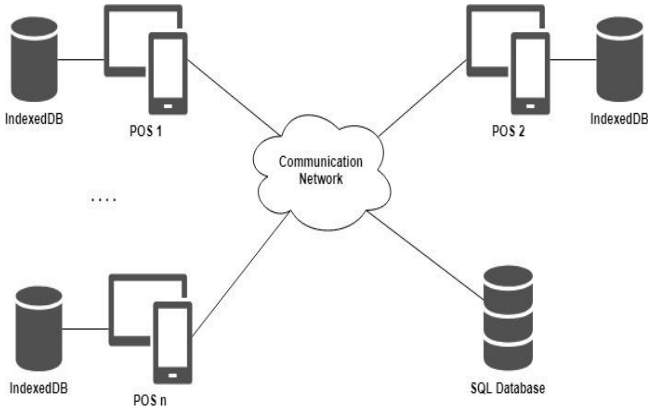

Figure [2]. Distributed database model of the system

Figure [3] describes relational database of the systemat Server.

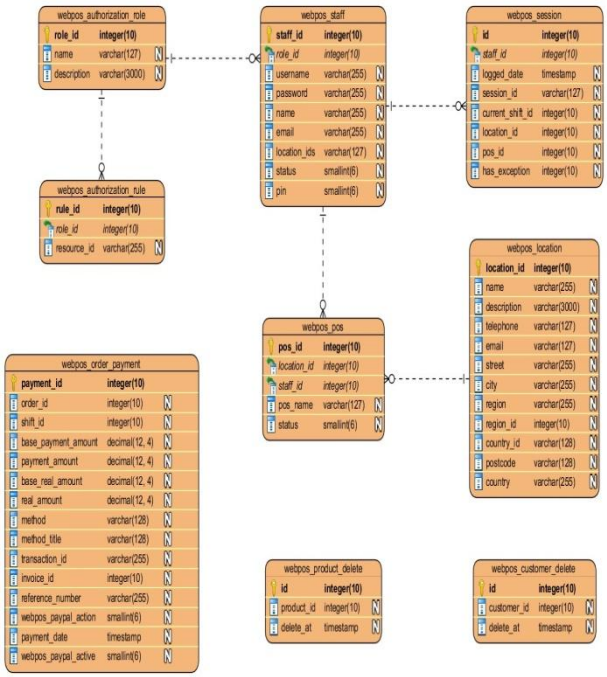

Figure [3]. Database of the system at Server

webpos_authorization_role Table: store information about the role of employee account; webpos_authorization_rule Table: store data of the access of the role, webpos_staff Table: store data of the employees; webpos_session Table: store data of employees' shifts; webpos_location Table: store data of location; webpos_pos Table: store data of POS; webpos_order_payment Table: store data of payment of order; webpos_search_product Table: store data of search of product; webpos_product_delete Table: store data of deleted products on server; webpos_custome_delete Table: store data of deleted customers on server.

Figure [4] describes database at POS (client) of the system. We use IndexedDB to store data under key and value.

Figure [4]. Database at POS (client)

action_log Table: store data of request needing transmitted to server such as adding, editing, deleting data, for the application to work without internet connection; Table error_log: store data of faulty action_log that cannot be transmitted to server; sync Table: store data synchronization with server such as product, customer, stock…; order Table: store data of orders; product Table: store data of products; stocks Table: store data of number of products in warehouse; category Table: store data of product categories; payment Table: store data of forms of payment; customer Table: store data of customers; product_index Table: store data of quick product search; customer_index Table: store data of quick customer search.

2.3 Data synchronization in the system

Every time salesperson log into the client's Retailer POS application must send a request to retrieve all necessary data from the server. Afterwards, the application will automatically send a request to check the changes in server every minute, if there is change in data, the server will return new data and the client will update that data to the database. When the data is not synchronized, the application will send a request to server through API instead of IndexedDB. Once the synchronization is complete, the data will update to IndexedDB at the corresponding client and the application will be switched to "offline" mode to retrieve data from IndexedDB to operate so as to help improve the application's performance and work even without network connection. Figure [5] describes the process of synchronization from Server to Client.

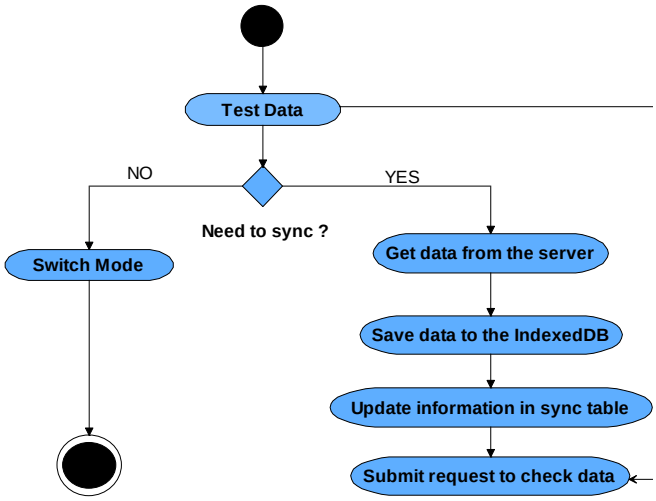

Figure 5. Synchronization from Server to Client

Figure [6]. Synchronization from Client to Server

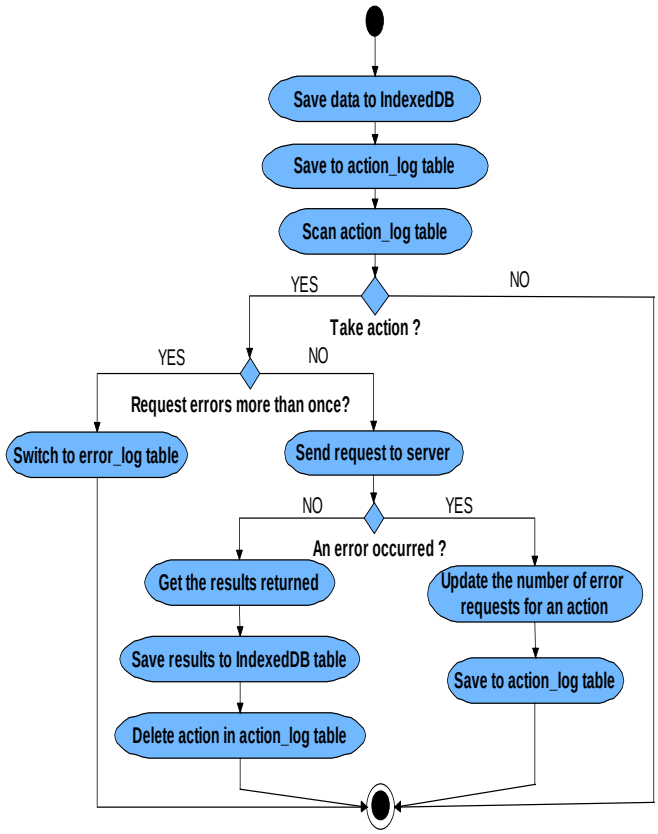

Figure 6.Synchronization form Client to Server

To improve productivity, preserve data and operate with or without network connection, all activities including creating, editing data at client are directly processed and saved at client. If then there is a connection to server, application will automatically sort out data in executing order so that the server can process data and return the result to client.

III. SYSTEM IMPLEMENTATION AND TESTING

3.1Build and deploy web POS module

This module is written to build API for application and interfaces for administrators.

Directory structure:

```
Webpos
  |- Block
  |- Controller
  |- etc
  |- Helper
  |- i18n
  |- Model
  |- Setup
  |- view
  `- registration.php
```

Block, containing class php, is the logic used to solve a problem or contains methods related to data, this directory contains logic processing for both frontend and backend; Controller: receive requests from users from http then forward the requests to logic processing and return the result; etc: include xml files used for configuration, notifying the system and related to parameters set for module such as permissions or locations of menu; Helper: contain support class such as calculating or storing constant variables; i18n: contain language files to change languages in the system; Model: contain files related data, database or a simple data structure such as Array, Collection, in general data source; Setup: directory related to database installation; View: store files related to interfaces, both frontend and backend.

3.2Build and deploy Retailer POS application

The system is implemented through 2 agents:

Salespeople: doing direct sales at POS locations.

Manager: interact with the entire system, control and supervise the entire operation of the system

Figure [7] illustrates the layers of Retailer POS application from client side, the components used in the structure of the application.

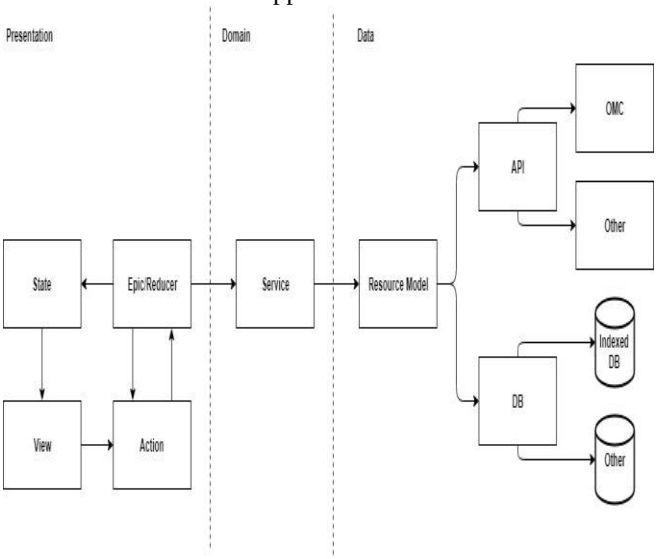

Figure [6]. The structure of POS application

View : displaying interface

Action : actions involving users'interaction with the application or actions carried out by the system (for example: users click on the product, lose internet connection…)

Epic/Reducer : handle actions

State : the current state of the application

Service : provides abstract classes for the Resource Model as a service contract.

Resource Model : Provide adapter to work with data. We can receive online data from online data sources (Server) or offline (IndexedDB)

API : provide data from online sources, through sending request APIs to the server

DB: provide data from offline sources, through accessing browser's IndexedDB.

3.3 Synchronizing data in the system

To synchronize data from server to client, we create SyncService.getAll(), from Client to Server we createsyncActionLog(), This two functions will implicitly execute in the background when operating.

3.4 *Performance* testing

Figure [6] is performance testing of system.

On the data set: 100,000 products and 100,000 customers, on the server system (configuration), client (configuration)

Test: 200 order every minute

Average response time: 928ms

Time : 1 minute

Successful number of requests: 199

Client: When Online averages 1s, Offline averages 600ms. These 2 parameters are tested on the client with speed of response and return the data of POS web application.

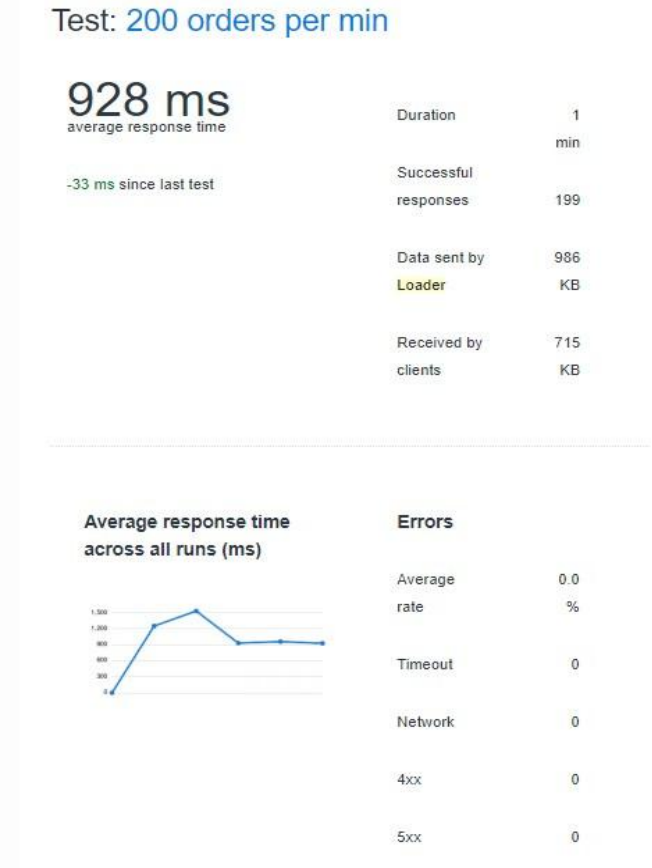

Figure [6]. Performance testing

IV. CONCLUSION

The system has achieved the following: basic functions for sales; user-friendly interface, convenient, easy to use; Multi-platform, can work well on mobile almost as a native application without installation; Easy to update without going through the Appstore; Help salespeople create orders quickly and accurately; operate with or without network connection; stable running system; In addition, local data at each station has overcome some limitations of centralized databases such as server overloading, bottlenecks when accessing, availability / reliability of low fault tolerance.

REFERENCES

1. B. Ajzele, Magento 2 Developer's Guide, 2015.
2. K. Edition, The Road to lear n React, 2017.
3. D. A. Hume, Progressive Web Apps, MANNING, 2017.
4. https://www.bigcommerce.com/blog/ecommerce-trends/, 2018
5. http://devdocs.magento.com, 2018
6. https://reactjs.org, 2018
7. http://stackoverflow.com, 2018
8. https://www.magestore.com, 2018

Author's profile

Dr. Phan Thi Ha is currently a lecturer at the Faculty of Information Technology at Posts and Telecommunications Institute of Technology in Vietnam and Computing Fundamental Department, FPT University, Hanoi 10000, Vietnam as well. She received a B.Sc.in Math & Informatics, a M.Sc. in Mathematic Guarantee for Computer Systems and a PhD. in Information Systems in 1994, 2000 and 2013, respectively. Her research interests include machine learning, natural language processing and mathematics applications.

Email:hapt@ptit.edu.vnand ***hapt@ptit.edu.vn***

ThS. Trint Thi Van is currently a lecturer and a researcher at the Faculty of Information Technology at Posts and Telecommunications Institute of Technology in Vietnamand Computing Fundamental Department, FPT University, Hanoi 10000, Vietnam as well. She received a B.Sc. in Electronics-Telecommunications in 1993 (HUST), M.Sc. in Computer Science in 1998 (HUST). Her research interests include machine learning, NLP and opinion mining.

Email: vanh22@yahoo.com, anhttv@ptit.edu.vnand anhttv@ptit.edu.vn