Mercado Libre XS DB Design

Daniel Alonso Chavarro Chipatecua
Code: 20241020066
Systems Engineer
Francisco Jose de Caldas District University
dachavarroc@udistrital.edu.co

***Abstract*—This article presents the design and development process of a database for the e-commerce platform ”Mercado Libre”. The application facilitates the purchase and sale of products, from technology to vehicles and properties, both new and used. The main objective is to support e-commerce for small and medium-sized businesses, improving their visibility and expansion.**

***Index Terms*—component, entity, attribute, MER, functionalities**

I. Introduction

E-commerce has transformed the way people buy and sell products, in that way the databases of these applications must be robust to support such amounts of data. This article describes the phases of the database design of Mercado Libre-XS, starting with the collection of user stories, the identification of functionalities, and the construction of an entity-relationship model.

II. Business Model

”Mercado Libre” is an e-commerce platform that facilitates interaction between sellers and buyers. The application allows sellers to offer new or used products and buyers to find items according to their preferences. The business model focuses on providing an easy and accessible shopping experience for any user, with the additional goal of improving the visibility of small and medium-sized businesses through customized online stores.

III. Methodology

The methodology used for the design of the database begins with the previous information necessary for the database design, and from that information the entity relationship model (MER) is built using the 10-step procedure explained by Professor Carlos Sierra:

IV. Processes and Information Required

In order to build the application we require more information apart from the business model, such as user stories and from there we can extract functionalities to turn them into entities and finally make the entity relationship model.

The first phase to be able to design the database is to know the necessities of the user. For this, first an interview was conducted with people asking what they would like to see in a Commerce applications, where they mentioned their point of view based on personal experiences with similar applications. These interviews were transformed into user stories where all the interviews are summarized, these user stories are in the format.

As a (role), I want (action), so what (impact)

The interviews with people using similar commerce apps revealed the following needs:

- Create an account to start buying or selling products.
- Filter products by criteria such as price, category, and location.
- List products with detailed descriptions, images, and prices.
- Manage a shopping cart and checkout securely.

These stories were transformed into key features of the app, such as advanced product search, the creation of personalized online stores, and the ability to offer discounts and promotions.

V. Design of the Database

A. Define Components

The components are the application modules; in this case they are:

- Transactions
- Client Service
- Products to Buy
- Delivery Service
- Sale of products

B. Define Entities

With the components defined, we can identify main entities found from the components, however the amount of entities is over 20, so the main entities are:

- User: Includes attributes such as first name, last name, address, email, and account status.
- Product: Includes details such as name, description, price, category, and product status.
- Transaction: Captures purchase details such as payment method, transaction status, and amount.

C. Relations between entities

Key relationships between entities were defined to ensure data integrity. For example, a user may have multiple transactions, and a product may be part of multiple transactions and shopping carts.
Nevertheless, as the amount of the entities are too big to be documented in the article, they will be ignored.

D. First MER Diagram

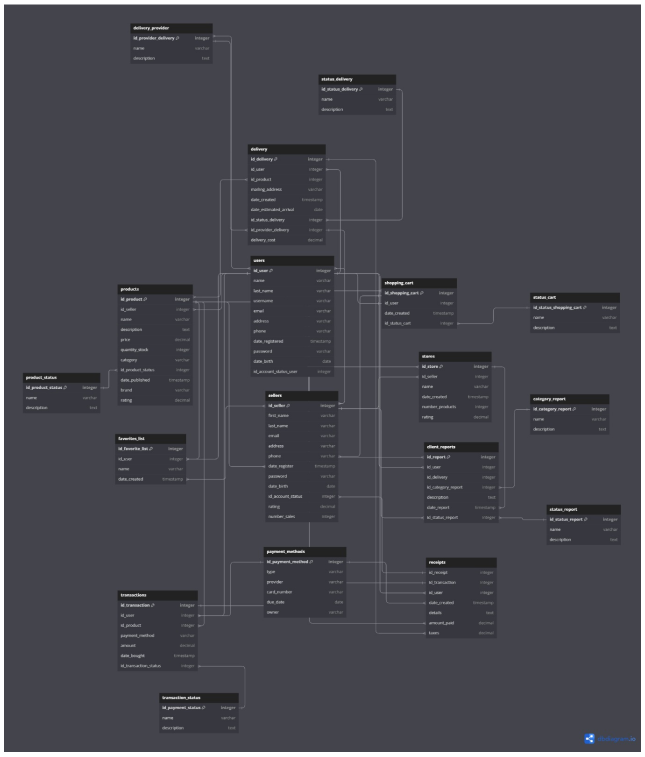

Fig. 1. First design MER

This diagram is the result of the first 5 steps applied, however will be more entites to add later.

E. Many-Many Consideration, Structure Data, Properties and Constrains

If you see the image showed above, you will see some Many-Many relationships, which are bad having in the MER, then the procedure to solve this is create middle entities to make 2 One-Many Entities like the relation between Seller and User.

Also defining the Structure data, properties and constrains are important to the integrity of the database. Some Data like email in user must be filled and have a limit of characters.

VI. Final Result

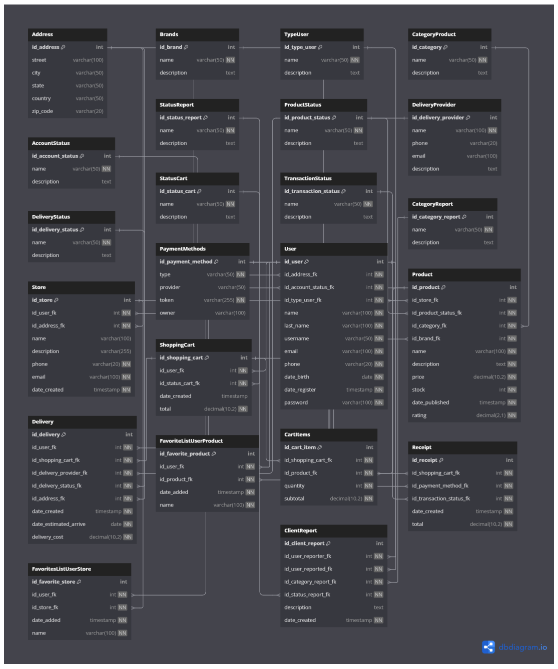

Fig. 2. Final Design of the MER

VII. Considerations

This design is basic compared to a fully E-commerce applications such as Mercado Libre or Amazon, so there is no total functionality, however this design would be considered as a base model for an E-commerce application.

VIII. Backend Implementation and Query Explanation

The backend of Mercado Libre-XS is built using FastAPI, a modern Python web framework for building APIs. The backend handles all the business logic, database interactions, and communication between the client and the database. The database operations are performed using Data Manipulation Language (DML) and Data Query Language (DQL) commands, which are encapsulated in CRUD (Create, Read, Update, Delete) classes.

A. Data Manipulation Language (DML)

DML operations are used to modify the data stored in the database. These operations include inserting new records, updating existing records, and deleting records. Below are some examples of DML operations used in the project:

1) Insert Operation: The `INSERT` command is used to add new records to a table. For example, when a new user registers on the platform, the following query is executed:

```
INSERT INTO Users (id_address_fk,
id_account_status_fk,
id_type_user_fk, name, last_name,
username, email, phone,
date_birth, date_register, password)

VALUES (1,1,1,'John', 'Doe',
'The Buyer', 'john.doe@example.com',
'3014258796', '1997-12-31','2021-5-24'. 'password'
```

This query inserts a new user into the `Users` table with the provided details.

2) Update Operation: The `UPDATE` command is used to modify existing records. For example, when a user updates their profile information, the following query is executed:

```
UPDATE Users
SET Name = 'Jane', Last_Name = 'Smith',
Email = 'jane.smith@example.com'
```

```
WHERE ID_User = 1;
```

This query updates the user's name, last name, and email in the `Users` table.

3) Delete Operation: The `DELETE` command is used to remove records from a table. For example, when a user deletes their account, the following query is executed:

```
DELETE FROM Users
WHERE ID_User = 1;
```

This query deletes the user with the specified ID from the `Users` table.

B. Data Query Language (DQL)

DQL operations are used to retrieve data from the database. The primary DQL command used in the project is `SELECT`. Below are some examples of DQL operations:

1) Retrieve User by ID: To retrieve a user's details by their ID, the following query is executed:

```
SELECT * FROM Users
WHERE ID_User = 1;
```

This query retrieves all the details of the user with the specified ID.

2) Retrieve Products by Category: To retrieve all products in a specific category, the following query is executed:

```
SELECT * FROM Products
WHERE id_category_fk = 32;
```

This query retrieves all products in the "Electronics" category (id 32).

3) Retrieve Transactions by User: To retrieve all transactions made by a specific user, the following query is executed:

```
SELECT * FROM Receipt
WHERE ID_User = 1;
```

This query retrieves all transactions made by the user with the specified ID.

C. Backend Workflow

The backend workflow involves the following steps:

1) **Request Handling**: When a client sends a request to the API, FastAPI routes the request to the appropriate service based on the endpoint.
2) **Validation**: The request data is validated to ensure it meets the required format and constraints.
3) **CRUD Operations**: The corresponding CRUD operation is executed to interact with the database. For example, if the request is to create a new user, the `create` method in the `UserCRUD` class is called.
4) **Database Interaction**: The CRUD method executes the appropriate SQL query to perform the operation (e.g., `INSERT`, `UPDATE`, `DELETE`, or `SELECT`).
5) **Response Generation**: The result of the database operation is formatted into a response and sent back to the client.

D. Example: User Registration

Let's take the example of a user registration process:

1) The client sends a POST request to the `/api/v1/user/create` endpoint with the user's details (name, email, password, etc.).
2) FastAPI routes the request to the `UserService` class.
3) The `create` method in the `UserService` class validates the request data and calls the `create` method in the `UserCRUD` class.
4) The `UserCRUD` class executes the following SQL query to insert the new user into the database:

```
    INSERT INTO Users (id_address_fk,
id_account_status_fk,
id_type_user_fk, name, last_name,
username, email, phone,
date_birth, date_register, password)

VALUES (1,1,1,'John', 'Doe',
'The Buyer', 'john.doe@example.com',
'3014258796', '1997-12-31','2021-5-24'. 'passw
```

5) The result of the query (e.g., the ID of the newly created user) is returned to the client as a Boolean

IX. Conclusion

The backend of Mercado Libre-XS is designed to handle a wide range of operations, from user registration to product management and transaction processing. By using FastAPI and a well-structured database, the backend ensures efficient and secure data handling. The use of DML and DQL commands allows for flexible and powerful data manipulation, while the CRUD classes encapsulate the database logic, making the codebase maintainable and scalable.