Development of an online store platform using object-oriented programming

Cristian Santiago López Cadena[1] 20222020027[1]
Carlos Alberto Barriga Gámez[2] 2022020179[2]

Universidad Distrital Francisco José de Caldas

Abstract— **Electronic commerce refers to any commercial transaction that uses the transfer of information over electronic networks to buy and sell goods or services.[2] In this way, the present research sought to prove how object-oriented programming is functional to develop an online store. For this purpose, the application design process was carried out, identifying the user stories, creating class, activity, sequence, deployment and state diagrams, as well as the navigation map between interfaces and screen prototypes (mockups). For implementation, Python version 3.12.2 was used for the development of the logical part of the software and the FastApi framework. For the data layer, the SQLAlchemy library was used to connect the database to the backend of the platform. For the development of the frontend, the software, Html, CSS and javascript, were used. The Django framework was also used to facilitate the implementation of the application on the web. Docker software was used to automate the deployment of the application as a local host. The research concluded that object-oriented programming is functional to develop an efficient and scalable virtual store.**

I. INTRODUCTION

Object-oriented programming is defined as a programming paradigm that attempts to simulate things in the real world through elements called objects. These objects have some characteristics such as inheritance, polymorphism, encapsulation, and abstraction. [3]

Abstraction is defined as a process where the essential characteristics of an object are extracted depending on their importance in solving the problem. On the other hand, encapsulation is the process of hiding information from the end user, establishing visible details that contribute to the user experience, and hidden details that support the software. Modularity aims to reduce a system into parts, in this way it is sought that the parts of the system have high connectivity (cohesion) and reduced association (coupling). Inheritance is a mechanism with which new classes are created that reuse and modify the behavior of previously created classes. Inheritance is defined hierarchically where a parent class is defined over several child classes. Polymorphism is defined as the ability of objects belonging to a class to generate a different response depending on the parameters sent to it. [3]

Likewise, these objects are defined as a series of behaviors called methods and properties known as attributes. [1].
Within the object-oriented programming paradigm, the SOLID principles are presented, aimed at improving the understandability and maintainability of software. [5]. The SOLID principles are discussed below: The single responsibility principle (S) determines that the software module must have a single reason for changing. Now, the principle of open and closed (O) states that the modules must be open for extension and closed for modification. The Liskov substitution principle (L) states that the classes of a program can be replaced by the instances of their subclasses, without altering the functionality of the program. The principle of interface segregation (I) states that if we create an interface we must ensure that the class that is going to implement this interface will have the ability to implement all the methods. Dependency inversion (D) means that parent and child modules should not depend on implementations, but on abstractions.
This project aims to develop an online store platform using the object-oriented programming paradigm. According to the author [2], electronic commerce refers to any commercial transaction in which the transfer of information through electronic networks is used to purchase and sell goods or services.
For the development of this research, the research titled “Design and implementation of a Marketplace platform integrated into the INFOREDCHILE site” is used as a reference, which shows the entire process carried out for the design of an online store, including the development of prototypes of graphical interfaces, as well as the design of the system architecture and the design of the software modules. Likewise, the research by (Zuñiga, 2021) shows the way in which the online store was implemented using the Python and JavaScript languages, along with the Node Js execution environment and the hypervisor software for hosting Virtual machines. Finally, in this research, a chapter was developed where the functionality of the system was tested, reviewing whether the interfaces created coincided with the prototypes made, as well as verifying whether the system met the functional and non-functional requirements of the software.
In this way, for the present investigation, it was decided to take the research of (Zuñiga, 2021) as the basis for testing the system. [4]
Starting from these concepts, it is important to highlight that this research aims to understand how object-oriented programming can be used to develop software for an online

store. In the same way, this project will be developed with a monolithic architecture, which is defined as an architecture in which all the layers of the application are compiled into a single unit. [6] So, in this document, the second chapter will discuss the methods and materials necessary to develop object-oriented software. In the next chapter, the experiments to be carried out to meet the objective of developing an online sales platform will be discussed and the results of this experimental practice will also be discussed. In a final chapter, the conclusions about the functionality of object-oriented programming in the creation of a functional online sales platform will be developed, which allows managing the purchase of products in an agile and effective way.

II. METHODS AND MATERIALS

A. *Methods*

This project aims to develop a functional online store platform. To do this, the following technical decisions will be made.

First, Python version 3.12.2 was used to develop the logical part of the software, and the FastApi framework was also used for communication between the different layers of the platform and the development of web services. For the data layer, an ORM tool was used, in this way the SQL alchemy library was selected, which facilitated the connection of the database from the back end of the platform, managing the database with a syntax similar to that of the language Python programming. For the development of the front end, the HTML tag language, the CSS style language and the JavaScript interpreted programming language were used. Likewise, for the front end, the Django framework was used to have the possibility of developing the website quickly. In the same way, Docker software was used to automate the deployment of the application as localhost.

Regarding the development process, we decided to use the GitHub storage software to facilitate the cooperative development of the online sales platform and the management of its versions.

Class diagram decisions: First of all, it was decided to use a User class, because it contains the necessary attributes to identify the user, define their access type (client or administrator) and the methods to create the account and log in. This class is the parent of the Admin and customer classes.

We decided to create a product class that has all the attributes that identify it. This class serves as a template to generate each of the product categories (Electronics, home-kitchen, sports-fitness and fashion). Likewise, it was decided to create a shopping history class, in which the products that belong to a user are stored. The catalog class was developed as a way to display several products in the same interface, and at the same time have the possibility of searching for a specific product.

On the other hand, a class called Shopping Cart was created, with this we want to relate a product with a customer to have the possibility of carrying out the transaction, modifying the stock and taking the product to the purchase history.

The credit card class was created with a numeric attribute, a functionality deadline, a cvv number to use the card securely, and a card name. In this way, with this class we have a specific payment method to use in the online store. Finally, in terms of developed classes, the User credentials class was created, this contains the user email and password attributes, its purpose is to solve problems with web services. The class diagram can be seen in the annexes section, in figure 4 Class diagram.

User interface decisions: For the development of the user interface models, the user stories made in the technical report were taken into account; in this way, a navigation map of the interfaces was created, which shows the order in which each interface is displayed according to the user's interaction with the application. The navigation map can be seen in the annexes section, in figure 5 Navigation map.

Likewise, for the development of the modeling of the graphic interfaces, mock ups were made for each of the interfaces with which the user interacts. In this way, the implementation of the interfaces was carried out based on mock ups. Below are the mock ups of the login interface and the home interface, which is the template for several interfaces of the application.

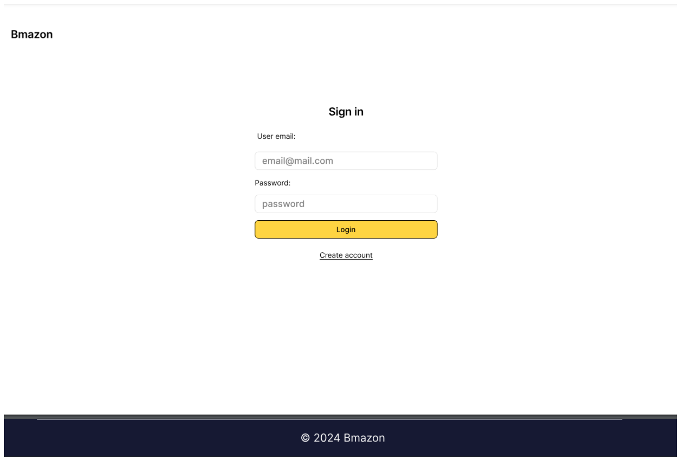


Fig. 1. Login interface

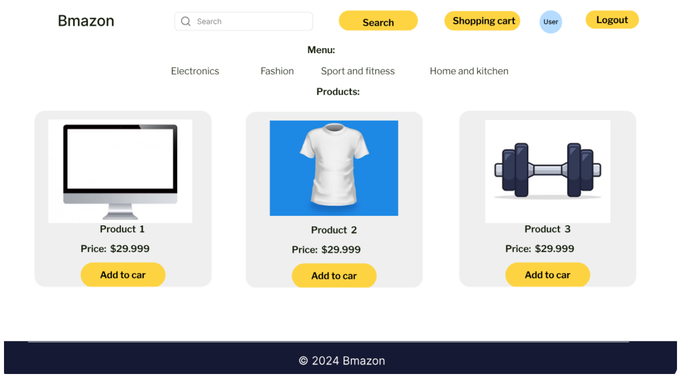
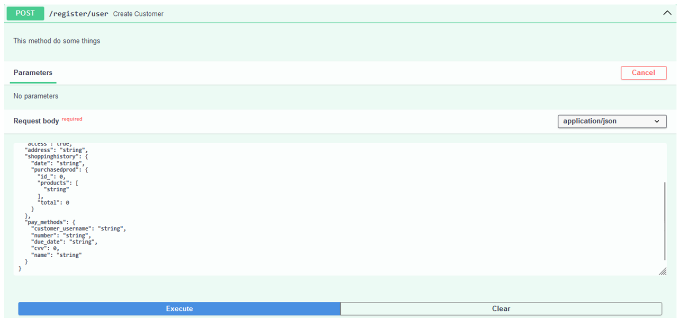


Fig. 2. Home interface

Services decisions:

Regarding the services performed in the application, the Fast API framework was used, following those projected for the services in the technical report. To test these services, the Swagger UI documentation from Fast API was used. Below are tests of some services:

Create customer (Post) (Customer): This service will receive as a parameter an object of the Customer class and will proceed to save it in the database. Below is the test carried out for this service:



/register/user Create Customer

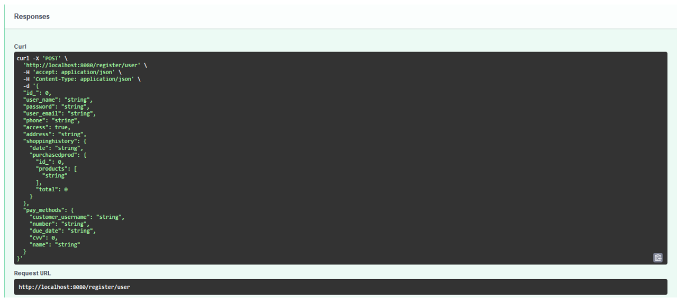
This method do some things

Parameters

No parameters

Request body required

```
  "access": true,
  "address": "string",
  "shoppinghistory": {
    "date": "string",
    "purchasedprod": {
      "id_": 0,
      "products": [
        "string"
      ],
      "total": 0
    }
  },
  "pay_methods": {
    "customer_username": "string",
    "number": "string",
    "due_date": "string",
    "cvv": 0,
    "name": "string"
  }
}
```



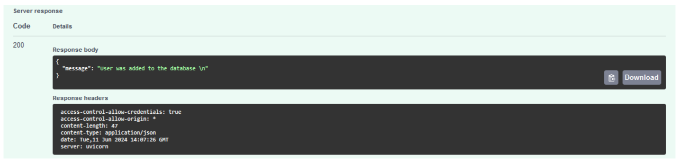
Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/register/user' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "id_": 0,
  "user_name": "string",
  "password": "string",
  "user_email": "string",
  "phone": "string",
  "access": true,
  "address": "string",
  "shoppinghistory": {
    "date": "string",
    "purchasedprod": {
      "id_": 0,
      "products": [
        "string"
      ],
      "total": 0
    }
  },
  "pay_methods": {
    "customer_username": "string",
    "number": "string",
    "due_date": "string",
    "cvv": 0,
    "name": "string"
  }
}'
```

Request URL

```
http://localhost:8080/register/user
```



Server response

Code Details

200

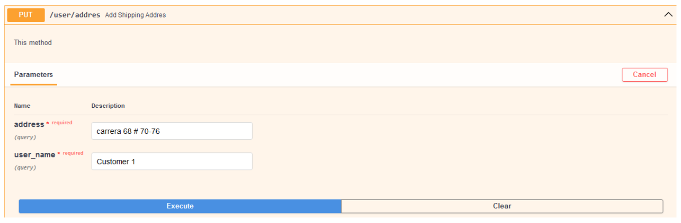
Response body

```
{
  "message": "User was added to the database \n"
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 47
content-type: application/json
date: Tue,11 Jun 2024 14:07:26 GMT
server: uvicorn
```

Add shipping address (Put) Parameters (address, user name): This service will receive as parameters the address that the user of type str wants to add and the name of the user who is adding it. Once the parameters have been received, it will proceed to search the database for the user with the name provided and make an update there. the address column, which will have a Nule value by default. Below is the test carried out for this service:

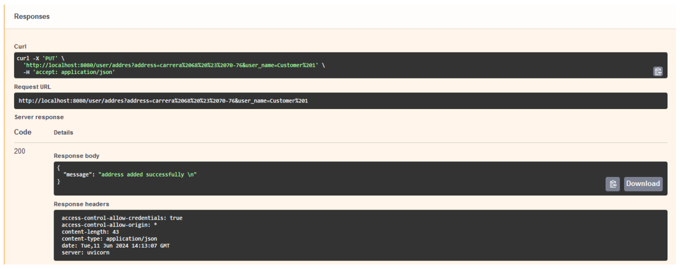


/user/addres Add Shipping Addres

This method

Parameters

Name	Description
address * required (query)	carrera 68 # 70-76
user_name * required (query)	Customer 1



Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/user/addres?address=carrera%2068%20%23%2070-76&user_name=Customer%201' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8080/user/addres?address=carrera%2068%20%23%2070-76&user_name=Customer%201
```

Server response

Code Details

200

Response body

```
{
  "message": "address added successfully \n"
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 43
content-type: application/json
date: Tue,11 Jun 2024 14:13:07 GMT
server: uvicorn
```

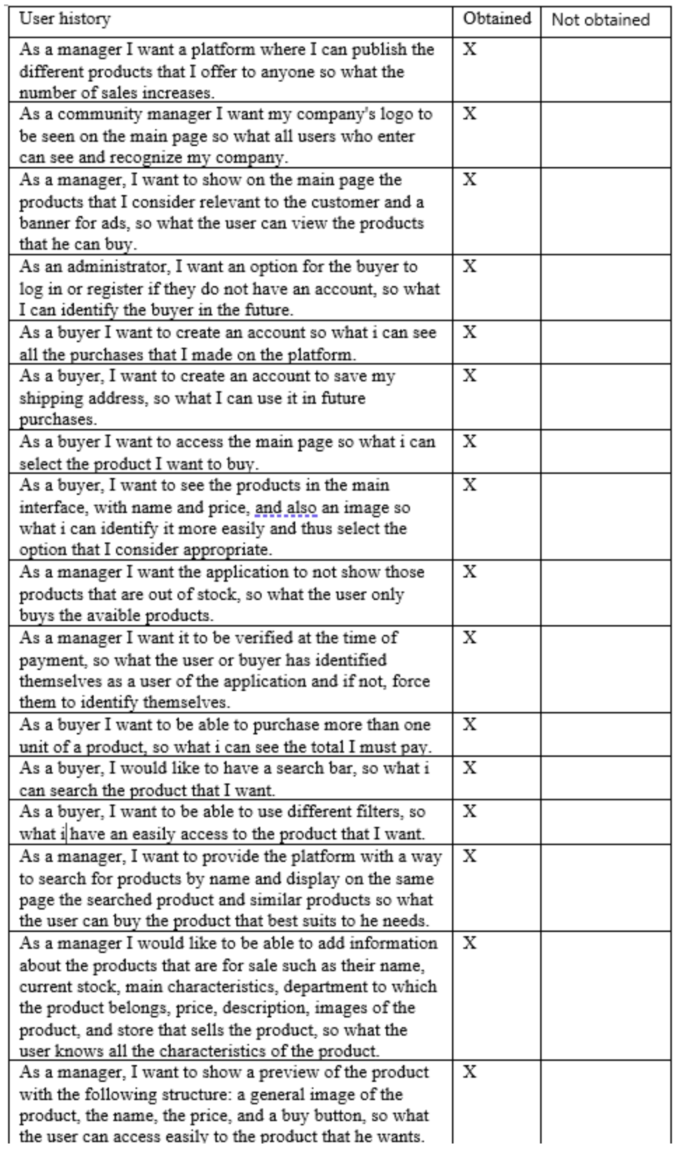
These tests were successful since when entering the requested parameters, the service generated the expected response.

B. Materials

In the project, some hardware elements will be used for the development of the online sales platform, among them 1 GB of ram will be used for the execution of the software since we consider it necessary for the efficient performance of the application. A 10th generation Intel core i5 processor will also be used because this processor can run the processes to meet the demand of the platform. Finally, a 1 GB storage space intended only for the execution of the application will be used, because we consider that this can store the software information and other software necessary for development, without generating problems with the space reserved for the operating system.

III. RESULTS

To verify the correct operation of the system, the following procedures were carried out. Firstly, after the implementation of the system, the user stories made in the technical report were taken into account to create a matrix of compliance with the software requirement, in this way with this matrix it was measured whether the software does respond to each user story. user, Below is the matrix:



User history	Obtained	Not obtained
As a manager I want a platform where I can publish the different products that I offer to anyone so what the number of sales increases.	X	
As a community manager I want my company's logo to be seen on the main page so what all users who enter can see and recognize my company.	X	
As a manager, I want to show on the main page the products that I consider relevant to the customer and a banner for ads, so what the user can view the products that he can buy.	X	
As an administrator, I want an option for the buyer to log in or register if they do not have an account, so what I can identify the buyer in the future.	X	
As a buyer I want to create an account so what i can see all the purchases that I made on the platform.	X	
As a buyer, I want to create an account to save my shipping address, so what I can use it in future purchases.	X	
As a buyer I want to access the main page so what i can select the product I want to buy.	X	
As a buyer, I want to see the products in the main interface, with name and price, and also an image so what i can identify it more easily and thus select the option that I consider appropriate.	X	
As a manager I want the application to not show those products that are out of stock, so what the user only buys the avaible products.	X	
As a manager I want it to be verified at the time of payment, so what the user or buyer has identified themselves as a user of the application and if not, force them to identify themselves.	X	
As a buyer I want to be able to purchase more than one unit of a product, so what i can see the total I must pay.	X	
As a buyer, I would like to have a search bar, so what i can search the product that I want.	X	
As a buyer, I want to be able to use different filters, so what i have an easily access to the product that I want.	X	
As a manager, I want to provide the platform with a way to search for products by name and display on the same page the searched product and similar products so what the user can buy the product that best suits to he needs.	X	
As a manager I would like to be able to add information about the products that are for sale such as their name, current stock, main characteristics, department to which the product belongs, price, description, images of the product, and store that sells the product, so what the user knows all the characteristics of the product.	X	
As a manager, I want to show a preview of the product with the following structure: a general image of the product, the name, the price, and a buy button, so what the user can access easily to the product that he wants.	X	

Fig. 3. User histories

In this way, the matrix shows that the system complied with the user stories. On the other hand, to test the application, the mock ups made previously were verified, and they were contrasted with the user interfaces made with HTML, CSS and JavaScript. When carrying out this procedure, it was concluded that the graphic interfaces designed are highly similar to the mock ups.

Finally, to test the persistence of the data, the data corresponding to user accounts, products, credit cards and addresses were created, which were stored in the database, on which the persistence was verified by making multiple purchases with different accounts. In this process, the attributes of the user account were verified when logging in and the correct creation and modification of a product was verified by observing the product and purchasing.

IV. CONCLUSIONS

Through this research it was possible to verify how object-oriented programming is functional for the development of a virtual store, because by using concepts such as those outlined in the solid principles, it is possible to achieve a modular application that is easily maintainable and scalable. Likewise, it is possible to demonstrate how, by using free software, web services technologies can be implemented to obtain efficient and effective software, which adapts to the needs of the current market. The development of tests to verify the compliance of the user stories allowed us to verify the correct implementation of the application according to the needs of a user of an online store. Likewise, when verifying the similarity between the mock ups made in the design process and the final interfaces obtained, it is concluded that the software met what was expected in the mock ups.

REFERENCES

[1] S. Valbuena and S. A. Cardona, "Object-oriented programming principles" Elizcom S.A.S, 2018, pp. 7.

[2] C. A. Robleto, "Electronic Commerce: Background, Definitions and Subjects", 2004, pp. 6-8.

[3] J. B .Bermudez, "Oriented object programming with java", 2012, pp. 7-8.

[4] M. Zuñiga, "Design and implementation of a Marketplace platform integrated into the inforedchile site", 2021,

[5] Autentia, "Software design, principles and patterns of the software development", 2012, pp. 9-17.

[6] Amazon, "What is the difference between monolithic and microservices architecture?", 2023.

V. ANEXXES

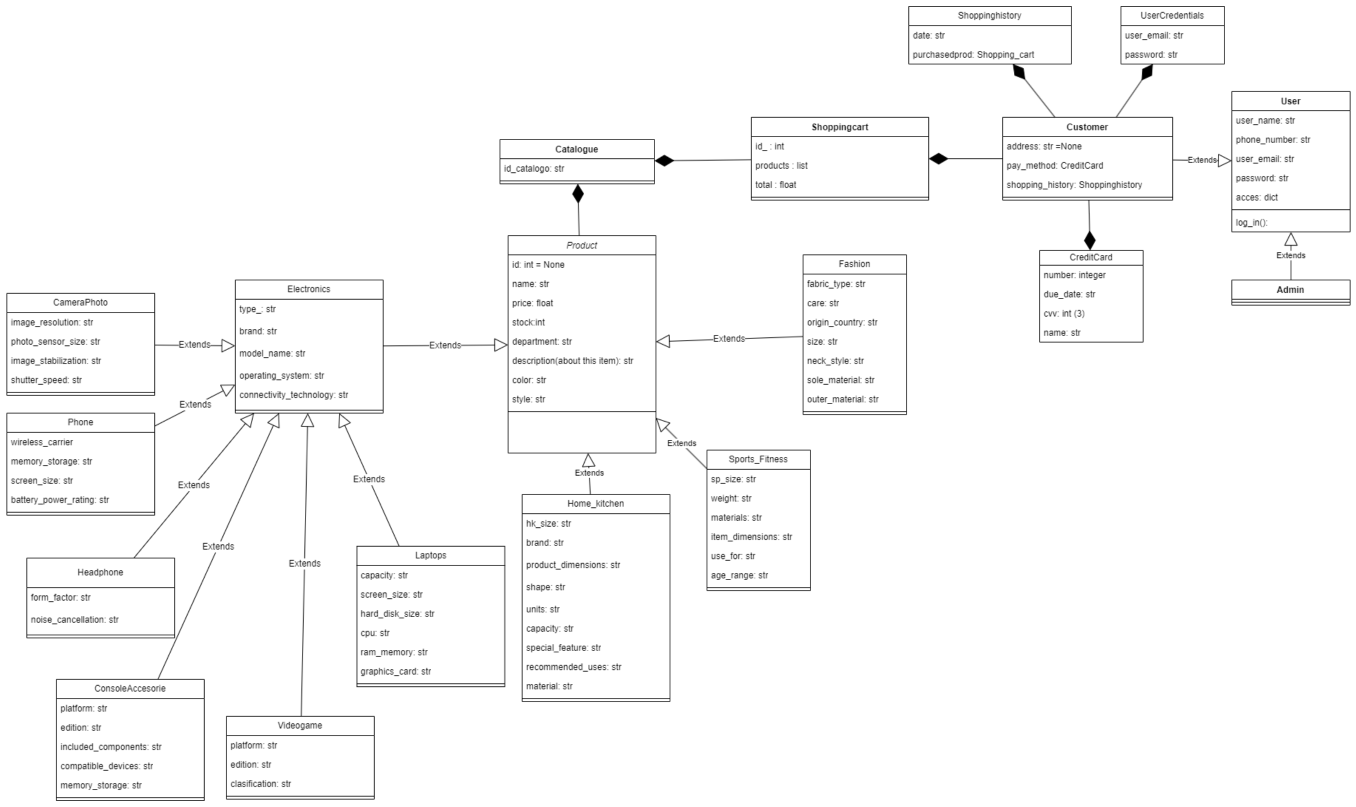


Fig. 4. Class diagram

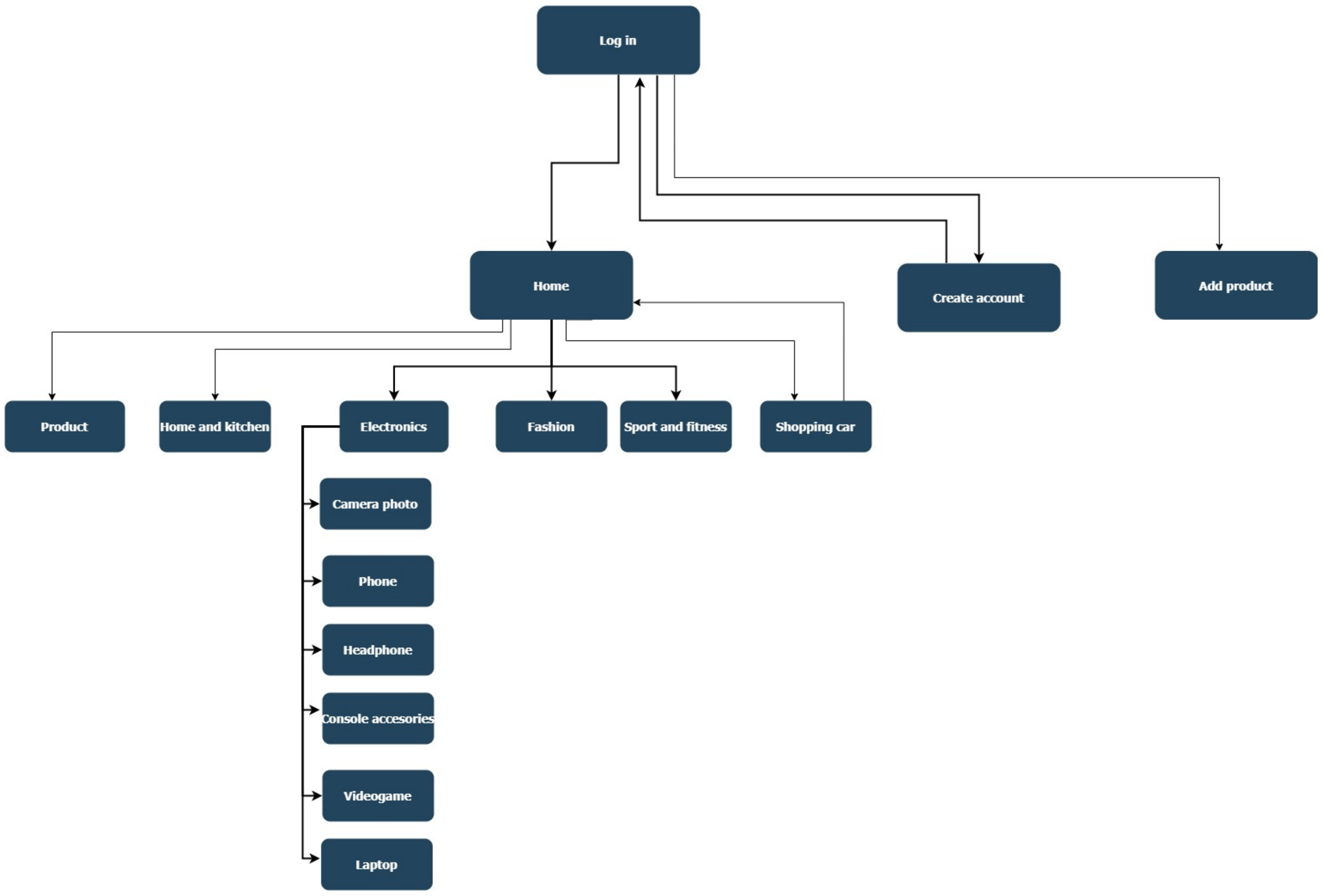


Fig. 5. Navigation map