

**Федеральное государственное автономное образовательное учреждение высшего  
образования  
«Национальный исследовательский университет «Высшая школа экономики»  
Факультет компьютерных наук**

**О Т Ч Е Т  
по дисциплине  
«Теория Баз Данных»**

**Выполнили студенты**

Бекетов Никита Дмитриевич (группа 194)  
Горбачев Ринат (группа 194)  
Жирнов Кирилл Владимирович (группа 197)

**Проверил:**

*(должность, ФИО руководителя практики)*

*23.12.2021*

**2021 год**

# Содержание

Содержание.....	2
1. Описание предметной области.....	3
1.1. Цель .....	3
1.2. Основные сценарии использования.....	3
2. Концептуальная модель .....	4
2.1. Диаграмма «Сущность-связь».....	4
2.2. Описание сущностей и связей .....	4
3. Логическая модель .....	6
3.1. Диаграмма «Таблица-связь».....	6
4. Дatalogическая модель .....	7
4.1. Используемая СУБД и диалект <i>SQL</i> .....	77
4.2. <i>DDL</i> -скрипты .....	8
5. Клиентское приложения .....	12
5.1. Архитектура .....	12
5.2. Пользовательский интерфейс .....	12
5.3. Отчёты .....	14
6. Заключение.....	20
6.1. Объёмные характеристики разработки .....	20

# 1. Описание предметной области

## 1.1. Цель

Целью данного проекта является создание интуитивно понятного веб-сервиса продажи различных товаров, например одежды, который бы легко взаимодействовал с базой данных и по средствам нее позволял редактировать наш сервис.

## 1.2. Основные сценарии использования

Данный сервис предназначается как для клиентов, так и для управляющих интернет магазина. Для клиентов главными сценариями использования является возможность авторизации, добавление необходимых товаров в корзину (или их удаление), разбиение товаров на категории, оформление заказа с учетом адреса и других атрибутов, а так же просмотр истории заказов. Для управляющих же возможность полного редактирование полей самих товаров, отслеживать популярность и статистику по каждому товару.

## 2. Концептуальная модель

### 2.1. Диаграмма «Сущность-связь»

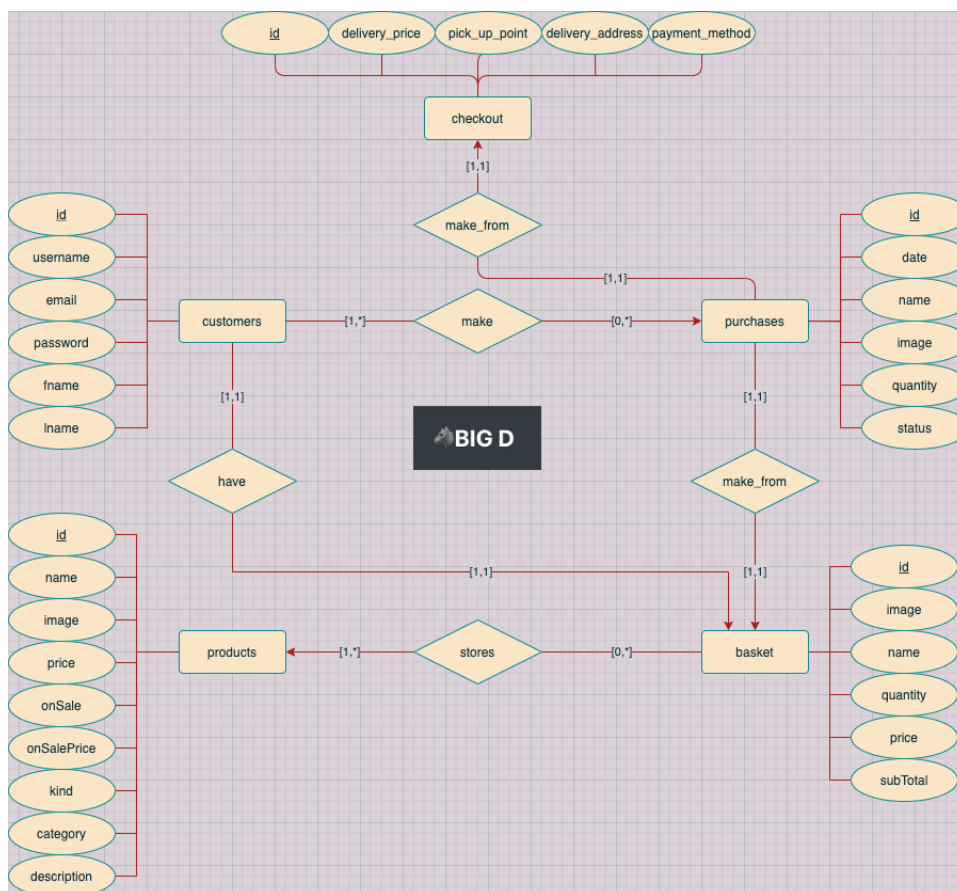


Рисунок 2.1. ER-диаграмма в нотации Чена

### 2.2. Описание сущностей и связей

При создании концептуальной модели было выделено 5 основных сущностей. У объектов каждой сущности есть свой уникальный идентификатор. Между разными сущностями существуют связи разных типов (так, например, один заказ может соответствовать ровно одному клиенту).

#### Сущности:

1. Сущность «Customers» содержит в себе информацию о пользователе: id, имя, фамилию, email, пароль и уникальный username.

2. Сущность «Products» хранит в себе всю информацию о товаре: id, название, изображение товара, цену, цену со скидкой, флаг для скидки, пол, категорию и описание.

3. Сущность «Basket» представляет из себя список выбранных товаров одним клиентом за один раз и содержит в себе: id, изображение товара, название товара, количество товара, цена товара, сумма заказа - subTotal.

4. Сущность «Purchases» представляет из себя список заказанных товаров одним клиентом за один раз. Содержит в себе id заказа, id заказчика, название товара, изображение товара, количество товара, дату совершения заказа и булевский статус доставки.

5. Сущность «Checkout» представляет собой оформление заказа, в нем содержится ключ, стоимость доставки, адрес доставки, метод оплаты и точка получения заказа.

#### **Связи:**

1. Связь «Customer-Basket» [один-к-одному] между клиентом и корзиной. У клиента есть ровно одна корзина, которую он наполняет элементами, у корзины так же только один владелец.

2. Связь «Basket-Product» [многие-к-многим] между корзиной и товарами. У корзины есть наполнение из многих элементов или корзина может быть пустой, один товар может относиться к нескольким корзинам или вовсе не быть в корзине.

3. Связь «Basket-Purchase» [один-к-одному] между корзиной и заказом. Заказ формируется из корзины.

4. Связь «Customer-Purchase» [один-ко-многим] между клиентом и заказом. Клиент может иметь несколько заказов, у заказа только один клиент.

5. Связь «Purchase-Checkout» [один-к-одному] между заказом и оформлением заказа. Заказ имеет одно оформление и оформление относится к одному заказу.

### 3. Логическая модель

#### 3.1 Диаграмма Таблица-Связь.

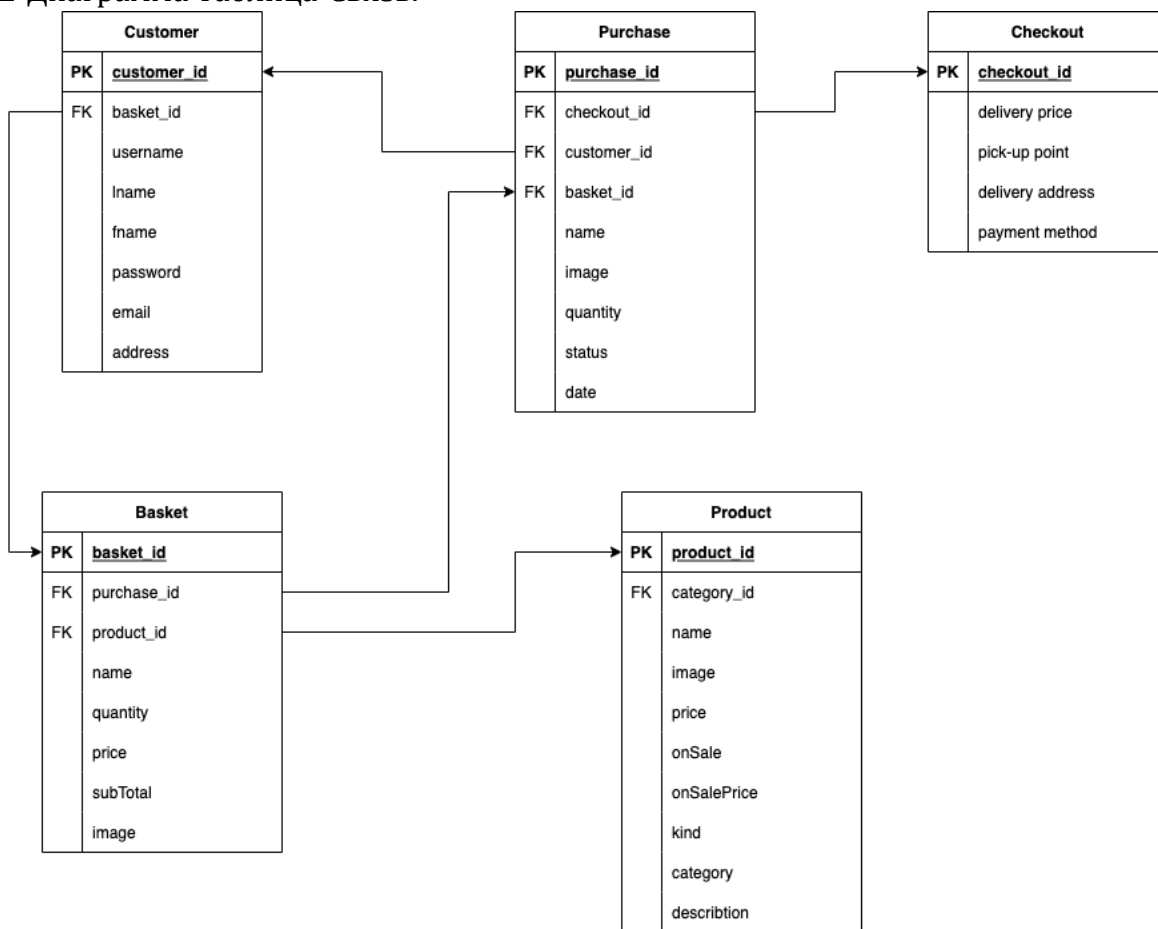


Рисунок 3.1. TR-диаграмма в нотации Чена

## 4. Логическая модель

### 4.1. Используемая СУБД и диалект SQL

Использована СУБД SQLite и диалект *Transact-SQL*.

### 4.2. DDL-скрипты

Создание схемы БД

```
CREATE TABLE basket (  
    image TEXT,  
    name TEXT,  
    qty INTEGER,  
    price REAL,  
    subTotal REAL,  
    id INTEGER PRIMARY KEY NOT NULL UNIQUE  
);  
  
CREATE TABLE customers (  
    id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    username TEXT NOT NULL UNIQUE,  
    password TEXT NOT NULL,  
    fname TEXT NOT NULL,  
    lname TEXT NOT NULL,  
    email TEXT NOT NULL UNIQUE,  
    adress TEXT NOT NULL  
);  
  
CREATE TABLE products (  
    id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    name TEXT NOT NULL,  
    image TEXT NOT NULL,  
    price REAL NOT NULL,  
    onSale INTEGER,  
    onSalePrice REAL,  
    kind TEXT NOT NULL,  
    category TEXT NOT NULL,  
    description TEXT  
);  
  
CREATE TABLE purchases (  
    uid TEXT,  
    name TEXT,  
    image TEXT,  
    quantity INTEGER,  
    id INTEGER,  
    date DATE NOT NULL DEFAULT CURRENT_DATE,  
    status BOOLEAN DEFAULT (False)  
);  
  
CREATE TABLE checkout (  
    checkout_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
```

```
delivery_price INTEGER NOT NULL,  
delivery_address TEXT NOT NULL,  
payment_method TEXT NOT NULL,  
pick_up_point TEXT  
);
```



## 5. Клиентское приложение

### 5.1. Архитектура

В качестве клиентского приложения было создано web-приложение с помощью языков Python, CSS, HTML, Java script.

Используемые библиотеки: cs50, sqlite3, flask, datetime, requests.

Основные функции реализованные в python файле с использованием flask:

1. def index() – открывается стартовая страница с товарами.
2. def buy() – позволяет выводить на экран товары.
3. def update() – позволяет поменять товары в корзине.
4. def filter() – сортирует товары.
5. def form() – направляет на страницу checkout'a.
6. def checkout() - добавляет заказ в заказы пользователя, отчищает корзину после checkout'a.
7. def remove() - позволяет удалить товары из корзины.
8. def login() - направляет на страницу ввода username и пароля.
9. def new() - направляет на страницу регистрации.
10. def logged() - проверка регистрации.
11. def history() - позволяет добавлять заказы и следить за ними, открывает страничку с заказами.
12. def logout() - позволяет выйти с аккаунта.
13. def registration() - позволяет зарегистрироваться.
14. def cart() - позволяет добавлять товары в корзину, открывает страницу корзины.

## 5.2. Пользовательский интерфейс.

Интерфейс представляет из себя web-приложение, где на главной странице отображены все товары, в шапке можно выбрать категорию, зайти или зарегистрироваться на сайте. Существуют отдельные HTML страницы для различных действий. Примеры интерфейса представлены на скриншотах в разделе «Отчёты».

## 5.3. Отчёты.

1. Регистрация пользователя. Пользователю необходимо указать его уникальный username, придумать пароль и подтвердить его, заполнить поля с именем, фамилией и email в соответствии с предъявленными требованиями. После заполнения всех полей необходимо нажать кнопку Register.

**BIG D**

### Register

Username Username must be 5 characters or more

Password Password needs to be at least 8 characters long

Confirm Password Passwords don't match

First Name First name must not be empty

Last Name Last name must not be empty

Email You need to enter a valid email address

Clear

Функция registration:

```
250 @app.route("/register/", methods=["POST"])
251 def registration():
252     # Get info from form
253     username = request.form["username"]
254     password = request.form["password"]
255     confirm = request.form["confirm"]
256     fname = request.form["fname"]
257     lname = request.form["lname"]
258     email = request.form["email"]
259     # See if username already in the database
260     rows = db.execute("SELECT * FROM customers WHERE username = :username ", username=username)
261     # If username already exists, alert user
262     if len(rows) > 0:
263         return render_template("new.html", msg="Username already exists!")
264     # If new user, upload his/her info into the customers database
265     new = db.execute(
266         "INSERT INTO customers (username, password, fname, lname, email) VALUES (:username, :password, :fname, :lname, :email)",
267         username=username, password=password, fname=fname, lname=lname, email=email)
268     # Render login template
269     return render_template("login.html")
```


Так выглядит часть данных из таблицы Customers.

	id	username	password	fname	lname	email
1	10	liomessi10	barcelona	Lio	Messi	lmessi@fcbarca.com
2	11	qwerty	qwerty123	Nikita	Beketov	qwerty@mail.ru
3	12	zxcghool	qwerty123	sb	xnj	1000-7@deadinside.ru

- Вход пользователя осуществляется с помощью уникального username и соответствующего пароля, которые хранятся в базе данных. При вводе неверных данных процедура повторяется. Пользователь также может выйти из аккаунта.

**BIG D**

### Log In to Buy




Ввели неправильные данные:

**BIG D**

### Log In to Buy

Wrong username or password.



Функция login для открытия html страницы:

```
190 @app.route("/login/", methods=["GET"])
191 def login():
192     return render_template("login.html")
193
```

В функции logged прописана обработка неверных данных:

```

201 @app.route("/logged/", methods=["POST"])
202 def logged():
203     user = request.form["username"].lower()
204     pwd = request.form["password"]
205     if user == "" or pwd == "":
206         return render_template("login.html")
207     # Find out if info in form matches a record in user database
208     query = "SELECT * FROM customers WHERE username = :user AND password = :pwd"
209     rows = db.execute(query, user=user, pwd=pwd)
210
211     if len(rows) == 1:
212         session['user'] = user
213         session['time'] = datetime.now()
214         session['uid'] = rows[0]['id']
215     if 'user' in session:
216         return redirect("/")
217     return render_template("login.html", msg="Wrong username or password.")
218

```

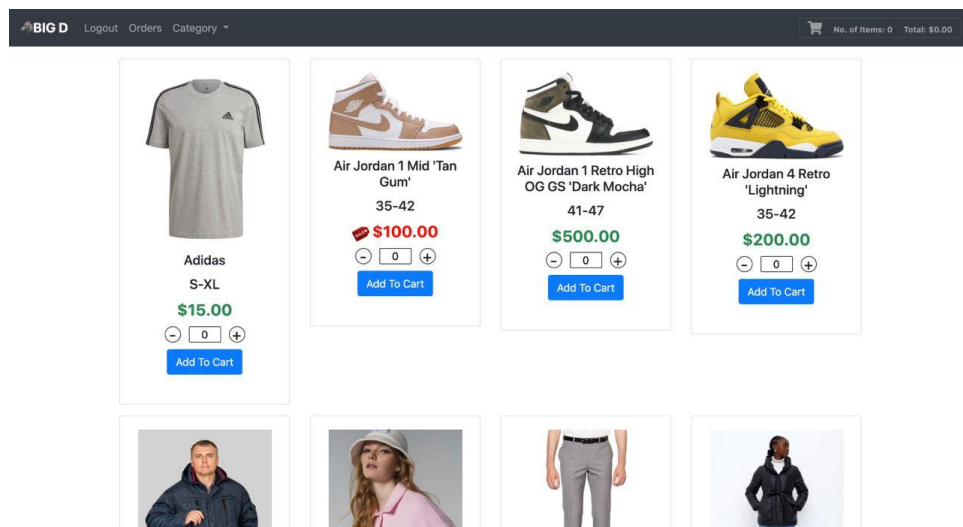
Функция logout:

```

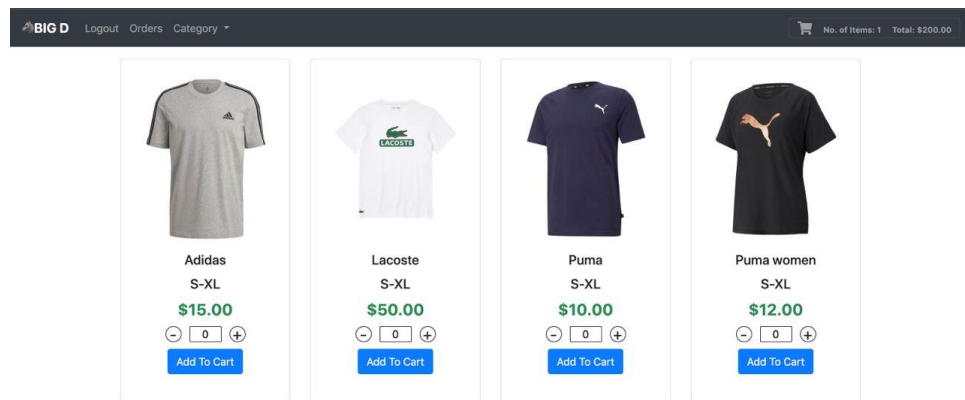
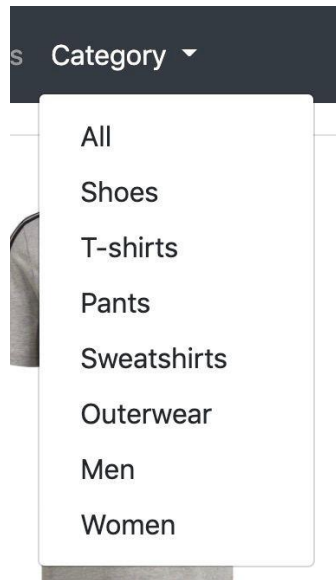
212 @app.route("/logout/")
213 def logout():
214     db.execute("DELETE from basket")
215     session.clear()
216     return redirect("/")

```

- Выбор товаров и добавление в корзину. Пользователь может выбрать товар на странице указав количество, также для удобства добавлена возможность выбирать товары по категориям.



Товары из категории T-shirts:



```

95 @app.route("/filter/")
96 def filter():
97     if request.args.get('category'):
98         query = request.args.get('category')
99         products = db.execute("SELECT * FROM products WHERE category = :query ORDER BY name ASC", query=query)
100     if request.args.get('sale'):
101         query = request.args.get('sale')
102         products = db.execute("SELECT * FROM products WHERE onSale = :query ORDER BY name ASC", query=query)
103     if request.args.get('id'):
104         query = int(request.args.get('id'))
105         products = db.execute("SELECT * FROM products WHERE id = :query ORDER BY name ASC", query=query)
106     if request.args.get('kind'):
107         query = request.args.get('kind')
108         products = db.execute("SELECT * FROM products WHERE kind = :query ORDER BY name ASC", query=query)
109     if request.args.get('price'):
110         query = request.args.get('price')
111         products = db.execute("SELECT * FROM products ORDER BY onSalePrice ASC")
112     shirtsLen = len(products)
113     shoppingCart = []
114     shopLen = len(shoppingCart)
115     totItems, total, display = 0, 0, 0
116     if 'user' in session:
117         shoppingCart = db.execute("SELECT name, image, SUM(qty), SUM(subTotal), price, id FROM basket GROUP BY name")
118         shopLen = len(shoppingCart)
119         for i in range(shopLen):
120             total += shoppingCart[i]["SUM(subTotal)"]
121             totItems += shoppingCart[i]["SUM(qty)"]
122         return render_template("index.html", shoppingCart=shoppingCart, products=products, shopLen=shopLen,
123                               shirtsLen=shirtsLen, total=total, totItems=totItems, display=display, session=session)
124     return render_template("index.html", products=products, shoppingCart=shoppingCart, shirtsLen=shirtsLen, shopLen=shopLen,
125                           total=total, totItems=totItems, display=display)
126
127

```

При выборе пользователем нужных товаров в нужном объеме, они попадают в корзину:

Basket

#

Item


Name

Quantity

Unit Price

Sub-Total

1



Air Jordan 4 Retro 'Lightning'

1

\$200.00

\$200.00

Total: \$200.00

MAKE CHANGES

CONTINUE SHOPPING

QUICK CHECKOUT

Редактирование корзины происходит после нажатия кнопки make changes:

Можно изменить количество - после чего нажать update, или убрать товар из корзины - кнопка remove.

BIG D

Logout Orders Category

No. of Items: 1

Total: \$200.00

Shopping Cart

#

Item


Name

Quantity

Unit Price

Sub-Total

1



Air Jordan 4 Retro 'Lightning'

1

UPDATE

\$200.00

\$200.00

Remove

Total: \$200.00

CONTINUE SHOPPING

PROCEED TO CHECKOUT

Функция update:

```

66 @app.route("/update/")
67 def update():
68     shoppingCart = []
69     shopLen = len(shoppingCart)
70     totItems, total, display = 0, 0, 0
71     qty = int(request.args.get('quantity'))
72     if session:
73         id = int(request.args.get('id'))
74         db.execute("DELETE FROM basket WHERE id = :id", id=id)
75         products = db.execute("SELECT * FROM products WHERE id = :id", id=id)
76         if products[0]["onSale"] == 1:
77             price = products[0]["onSalePrice"]
78         else:
79             price = products[0]["price"]
80         name = products[0]["name"]
81         image = products[0]["image"]
82         subTotal = qty * price
83         db.execute(
84             "INSERT INTO basket (id, qty, name, image, price, subTotal) VALUES (:id, :qty, :name, :image, :price, :subTotal)",
85             id=id, qty=qty, name=name, image=image, price=price, subTotal=subTotal)
86         shoppingCart = db.execute("SELECT name, image, SUM(qty), SUM(subTotal), price, id FROM basket GROUP BY name")
87         shopLen = len(shoppingCart)
88         for i in range(shopLen):
89             total += shoppingCart[i]["SUM(subTotal)"]
90             totItems += shoppingCart[i]["SUM(qty)"]
91         return render_template("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total, totItems=totItems,
92                               display=display, session=session)
93 
```

Функция remove:

```

146 @app.route("/remove/", methods=["GET"])
147 def remove():
148     out = int(request.args.get("id"))
149     db.execute("DELETE from basket WHERE id=:id", id=out)
150     totItems, total, display = 0, 0, 0
151     shoppingCart = db.execute("SELECT name, image, SUM(qty), SUM(subTotal), price, id FROM basket GROUP BY name")
152     shopLen = len(shoppingCart)
153     for i in range(shopLen):
154         total += shoppingCart[i]["SUM(subTotal)"]
155         totItems += shoppingCart[i]["SUM(qty)"]
156     display = 1
157     return render_template("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total, totItems=totItems,
158                           display=display, session=session)
159
160

```

4. Оформление заказа. После нажатия кнопки proceed to checkout мы переходим на страницу оформления:

### Checkout form

**Billing address**

Address

Delivery company

Delivery price

☐ Use pick-up point

Pick-up point

**Payment**

☒ Credit card  
☐ Debit card  
☐ PayPal

Name on card

Credit card number

Full name as displayed on card

Expiration

CVV

[Home](#) [Log in](#)

Тут необходимо заполнить все поля, после чего оформить заказ кнопкой checkout.

Функция checkout заносит данные о заказе в базу.

```

128 @app.route("/form/")
129 def form():
130     return render_template("form.html")
131
132
133 @app.route("/checkout/")
134 def checkout():
135     order = db.execute("SELECT * from basket")
136     for item in order:
137         db.execute("INSERT INTO purchases (uid, id, name, image, quantity) VALUES(:uid, :id, :name, :image, :quantity)",
138                 uid=session["uid"], id=item["id"], name=item["name"], image=item["image"], quantity=item["qty"])
139     db.execute("DELETE from basket")
140     shoppingCart = []
141     shopLen = len(shoppingCart)
142     totItems, total, display = 0, 0, 0
143     return redirect('/')
144

```

Пример таблицы purchases:

	uid	name	image	quantity	id	date	status
24	11	Air Jordan 4 Retro 'Lightning'	Air Jordan 4 Retro 'Lightning'.png	1	4	2021-12-23	NULL
25	11	Air Jordan 4 Retro 'Lightning'	Air Jordan 4 Retro 'Lightning'.png	1	4	2021-12-23	NULL
26	11	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
27	11	Adidas	manT_1_Adidas.jpg	1	5	2021-12-25	NULL
28	11	Adidas	manT_1_Adidas.jpg	1	5	2021-12-25	NULL
29	11	Sweatshirt printed	sweatshirt printed men.png	1	17	2021-12-25	NULL
30	12	Hello kitty women	sweatshirt hello-kitty women.png	1	15	2021-12-25	NULL
31	12	Air Jordan 1 Mid 'Tan Gum'	Air Jordan 1 Mid 'Tan Gum'.png	1	3	2021-12-25	NULL
32	12	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
33	12	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
34	12	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
35	12	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
36	12	Adidas	manT_1_Adidas.jpg	1	5	2021-12-25	NULL
37	12	Adidas	manT_1_Adidas.jpg	1	5	2021-12-25	NULL
38	12	Adidas	manT_1_Adidas.jpg	1	5	2021-12-25	NULL
39	12	Adidas	manT_1_Adidas.jpg	1	5	2021-12-25	NULL
40	12	Hello kitty women	sweatshirt hello-kitty women.png	1	15	2021-12-25	NULL
41	11	Air Jordan 4 Retro 'Lightning'	Air Jordan 4 Retro 'Lightning'.png	1	4	2021-12-25	NULL
42	11	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
43	11	Travis Scott x Air Jordan 6 Retro 'British Khaki'	Travis Scott x Air Jordan 6 Retro 'British Khaki'.png	1	1	2021-12-25	NULL
44	11	Travis Scott x Air Jordan 6 Retro 'British Khaki'	Travis Scott x Air Jordan 6 Retro 'British Khaki'.png	1	1	2021-12-25	NULL
45	11	Air Jordan 1 Retro High OG GS 'Dark Mocha'	Air Jordan 1 Retro High OG GS 'Dark Mocha'.png	1	2	2021-12-25	NULL
46	11	Without women	gena.png	1	14	2021-12-25	NULL
47	11	Without women	gena.png	1	14	2021-12-25	NULL
48	11	Without women	gena.png	1	14	2021-12-25	NULL

5. Сценарий просмотра своих заказов. Мы можем посмотреть свои заказы с помощью нажатия на кнопку Orders:

[Logout](#)
[Orders](#)
[Category](#)

No. of Items: 0    Total: \$0.00

### Your Shopping History

Items you've bought in the past.

#	Item	Name	Quantity	Date	Status	
1		Hello kitty women	1	2021-12-25	None	<a href="#">Buy Again</a>
2		Air Jordan 1 Mid 'Tan Gum'	1	2021-12-25	None	<a href="#">Buy Again</a>
3		Air Jordan 1 Retro High OG GS 'Dark Mocha'	1	2021-12-25	None	<a href="#">Buy Again</a>
4		Air Jordan 1 Retro High OG GS 'Dark Mocha'	1	2021-12-25	None	<a href="#">Buy Again</a>
5		Air Jordan 1 Retro High OG GS 'Dark Mocha'	1	2021-12-25	None	<a href="#">Buy Again</a>
6		Air Jordan 1 Retro High OG GS 'Dark Mocha'	1	2021-12-25	None	<a href="#">Buy Again</a>
7		Adidas	1	2021-12-25	None	<a href="#">Buy Again</a>
8		Adidas	1	2021-12-25	None	<a href="#">Buy Again</a>
9		Adidas	1	2021-12-25	None	<a href="#">Buy Again</a>
10		Adidas	1	2021-12-25	None	<a href="#">Buy Again</a>
11		Hello kitty women	1	2021-12-25	None	<a href="#">Buy Again</a>

6. Анализ данных. Мы можем посмотреть, например, топ по позициям товаров в таблице с помощью запросов к таблице purchases:



```
select name, count(name)
from purchases
group by name
```

## 6. Заключение

### 6.1 Объёмные характеристики разработки

В процессе работы над этим проектом была создана база данных, состоящая из 5 основных таблиц, которые были заполнены синтетическими данными, например, в таблице products 21 запись товаров. Осуществлены связи между таблицами и обновление. Было реализовано web-приложение с подключением к нашей базе с помощью библиотеки Flask. В коде на Python было реализовано 14 функций, с помощью которых осуществляется логика и связь с базой. Суммарный объем кода Python файла - 240 строк, было написано 7 html файлов и 2 JavaScript + CSS файл.