“Kyzylorda Nazarbayev International School of Chemistry and Biology” AEO
“Nazarbayev International School”

Computer Science Project

Title: Website of bakery

Student of the 12th grade: Nurlybai Beksultan

Teacher: Serik Akhmetov

Kyzylorda, 2022

Contents

Introduction

3.0 DESIGN

3.1 Objectives and Feedback

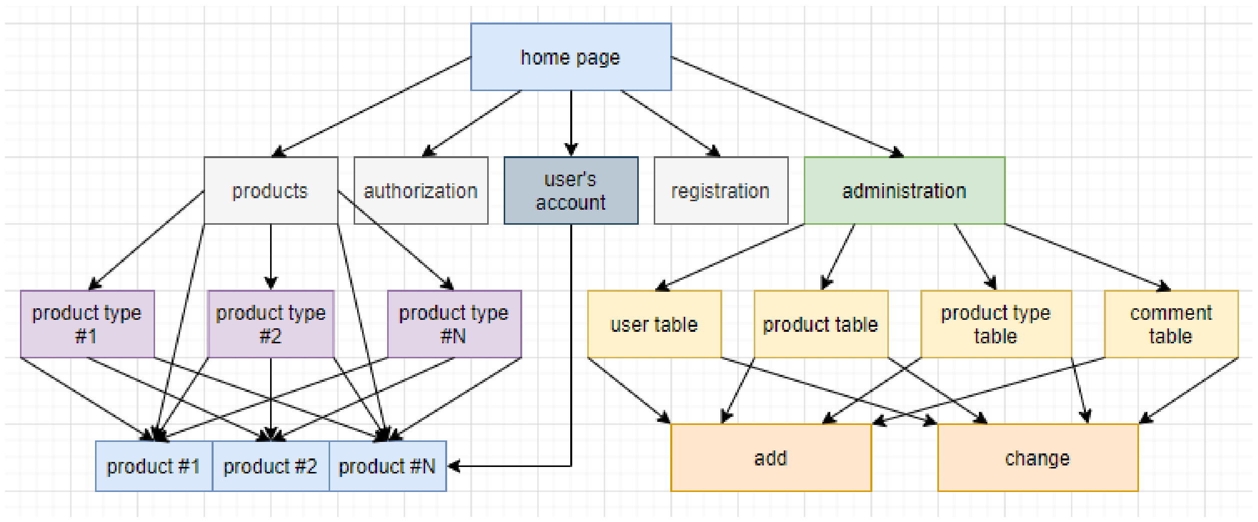
I decided that my website must contain base functional as authorization, registration, administration and others. All requirements are shown below:

- To create convenient and instinctively understandable interface.
- The design has to be made of modern technology by using HTML, CSS and JavaScript.
- All backend will be made of program language as python, especially django which is framework of python and is used to create mostly large websites.
- In process of authorization and registration, the demanded input data is an email and a password.
- The website should have the cart function which is used by registered users.
- The website should contain different variation of product.
- There must be forms to sort and to filter products.
- In the administration page, only administer can open the page and create, change or delete whole data.

3.2 Data input and output

Process	**Input**	**Output**
Authorization	Name and password	• To return to home page and to show message "Hello, "user's name"". • Message "Name or password is incorrect".
Registration	Name, email, password, password2	• New record in the table "user" and returning to home page.
Sending comments	Message	• To show comment on the product page.
Creating new record about a product by administrator	Name, ingredients, price, type of product, image	• New product in the website.
Sending product to cart	button	• New product in the user's account.

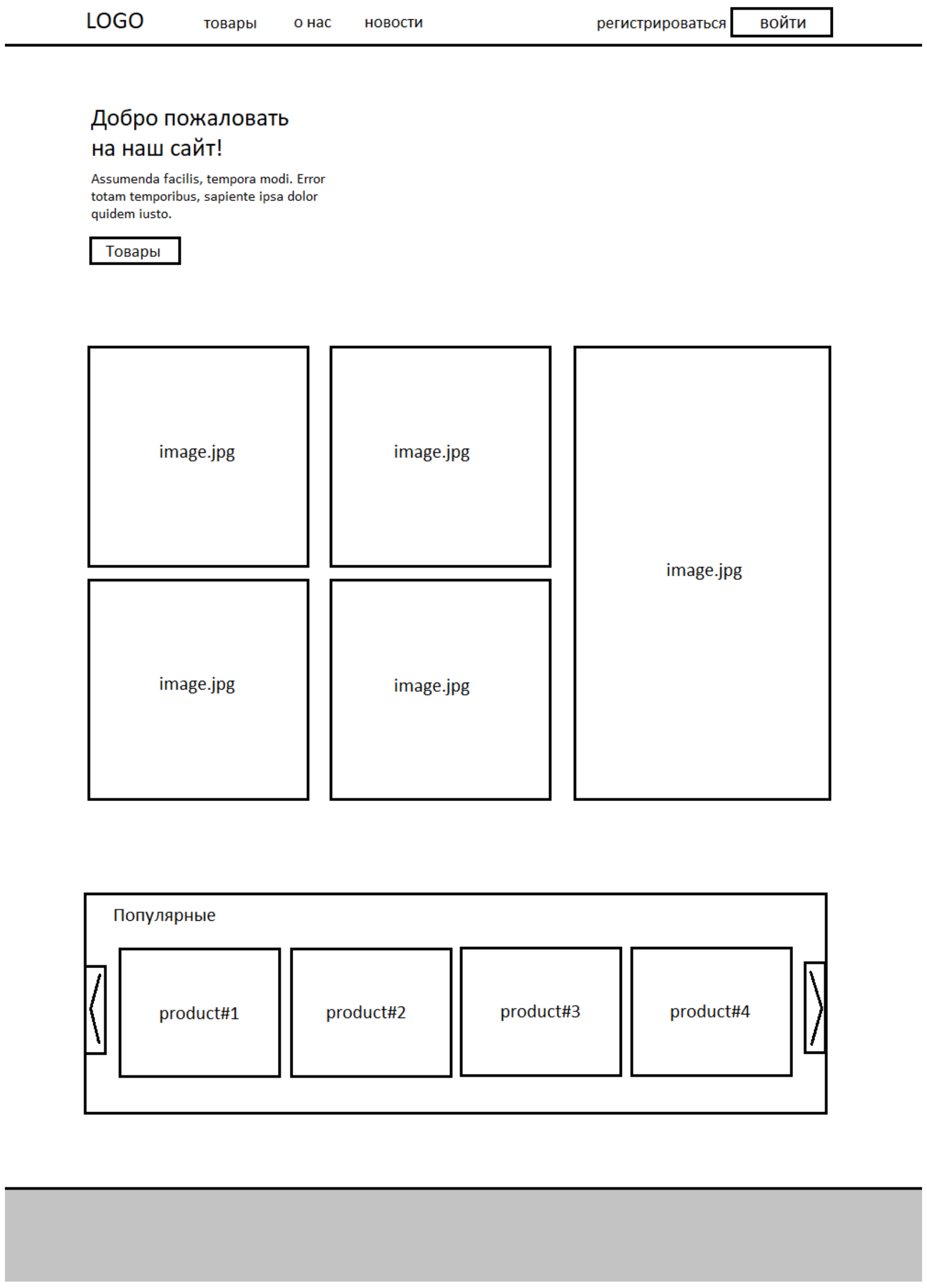
3.3 Site hierarchy and Page Functions



- Main page: it is used to show general information like popular products, photos of bakery and links with all main pages.
- Products page: it is used to represent whole variation of goods.
- Products type pages: it consists of products that have the same types like cakes or cookies.
- Product N page: there are all data about product N. In this page, users can transfer the product to account that is used as cart. In addition, users can send comments under the data.
- Authorization page: to give permission to users who have accounts and input right data.
- Registration page: to give account to guests via receiving their personal data.
- User's account page: it is used like cart where user carries products that are interesting to him.
- Administration page: it is used by administrator to create, change or delete records in tables.

3.4 Interface

I made a prototype of website by program paint.

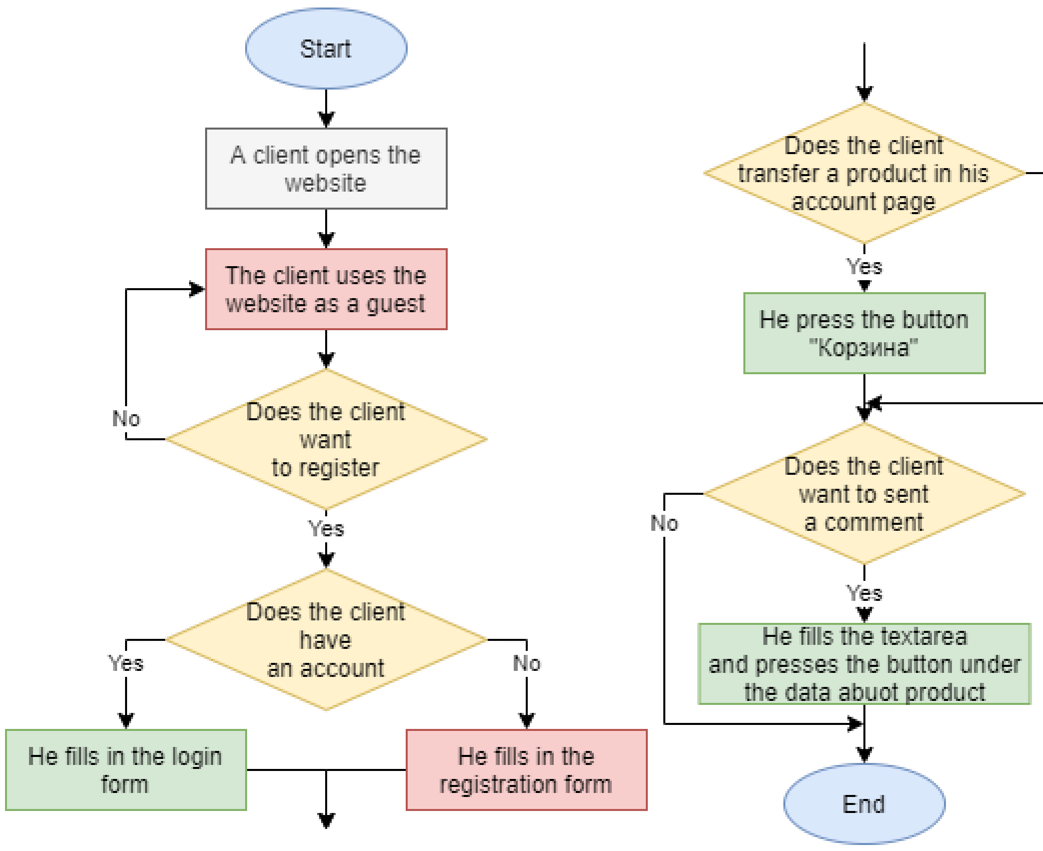


3.5 Diagrams of the new system

These kinds of diagrams are so useful when I will not only just make design, programming backend that is more harder becomes easier and understandable.

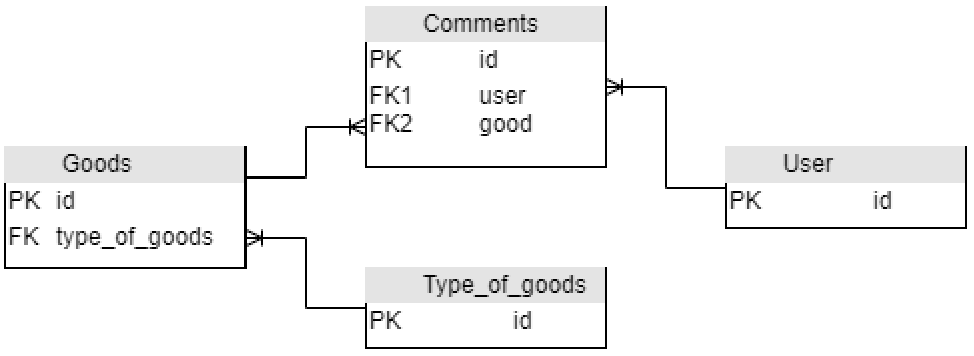
<u>Flowchart:</u>

I made flowchart of the new system that represent how user will act on the website and how the website will make actions.


Start
A client opens the website
The client uses the website as a guest
Does the client want to register
No
Yes
Does the client have an account
Yes
No
He fills in the login form
He fills in the registration form
Does the client transfer a product in his account page
No
Yes
He press the button "Корзина"
Does the client want to sent a comment
No
Yes
He fills the textarea and presses the button under the data abuot product
End

ERD

To understand better the reletionship between tables I illustrate it in ERD diagram.



3.6 Data dictionary

User

Field Name	Type	Field length	Validation	Description	Example
Id(Primary key)	int	11	Presence check, Format check	Unique ID	1
username	varchart	150	Presence check, Format check	Name of user	Beksultan

email	varchart	unlimited	Presence check	Email of user	User123@gmail.com
password	varchart	unlimited	Presence check, Format check	password	qwerty123
password2	varchart	unlimited	Presence check, Format check	to confirm password	qwerty123

Goods

Field Name	**Type**	**Field length**	**Validation**	**Description**	**Example**
Id (Primary key)	int	11	Presence check, Format check	Unique ID	1
title	varchart	100	Presence check, Format check	Name of product	Chocolate cake
price	int	unlimited	Presence check, Format check	price	2000
ingredients	varchart	300	Format check	Ingredients that are contained in product	Chocolate, milk, dyes.
photo	varchart	unlimited	Format check	photo	Cake2956.png
created_at	int	unlimited	Presence check, Format check	Date of creating product	1954328034
updated_at	int	unlimited	Presence check, Format check	Date of changing product	1954310026
is_published	boolean	unlimited	Presence check, Format check	To give or not permission to publish good	True
type_of_goods (foreign key)	varchart	unlimited	Presence check, Format check	Connect with table “Type_of_goods”	cake

Type_of_goods

Field Name	**Type**	**Field length**	**Validation**	**Description**	**Example**
id (Primary key)	int	11	Presence check, Format check	Unique ID	1
title	varchart	100	Presence check, Format check	Name of type	cake

Comments

Field Name	**Type**	**Field length**	**Validation**	**Description**	**Example**
id (Primary key)	int	11	Presence check, Format check	Unique ID	1
user (foreign key)	varchart	unlimited	Presence check, Format check	Connect with table “User”	Beksutan
good (foreign key)	varchart	unlimited	Presence check, Format check	Connect with table “Goods”	Chocolate cake
body	text	unlimited	Presence check,	Comment of	This is so

			Format check	user	delicious!
created_at	int	unlimited	Presence check, Format check	Date of creating comment	1954328034
is_published	boolean	unlimited	Presence check, Format check	To give or not permission to publish good	True

3.7 Intended benefits

1. Only administrator can manipulate products and their types.
2. Only registered users can send comment and have permission to account.
3. When texting comment, a user doesn't need to write his name or name of product.
4. There are sorting and filtering form to products and comments.
5. Passwords of users safe in database in the form of hash with extra encrypting functions.
6. Easy to transfer a product page to account by pressing button.

3.8 Limits of the scope of the Solution

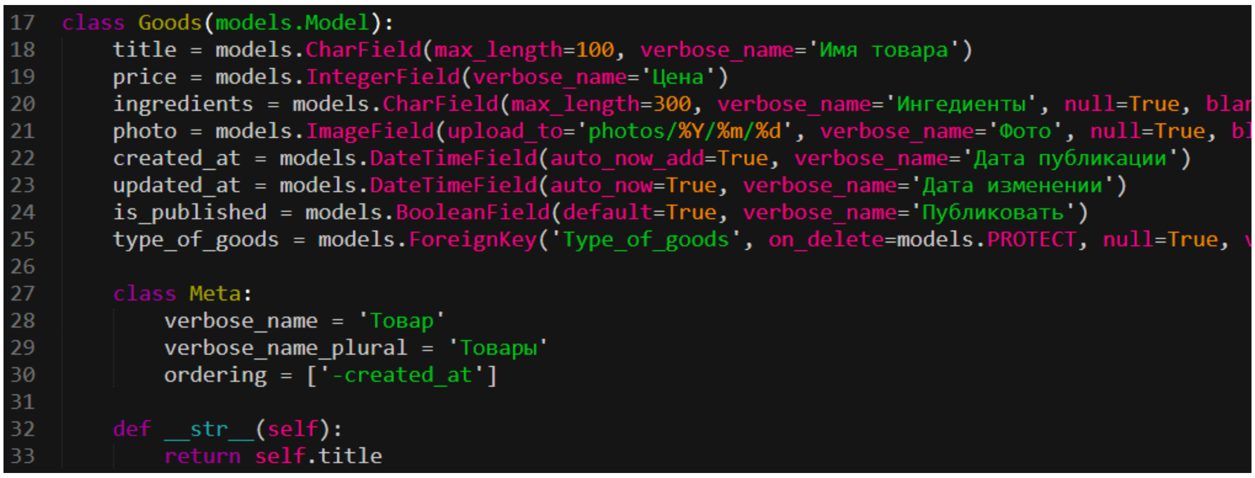
1. Users can't do payment, duo to uselessness of the function in the website and complexity of the function.
2. Users cannot restore their password, when they forget it.
3. Website was made for desktop. So it may doesn't work worse in mobile phones.

4.0 Development

4.1 SQL Statements and queries code

Duo to Django that I used to build my website, I did not use any SQL statements and give any queries. However, Django suggests models that are python classes which represents options of tables in database. For instance, extra options like order, name of record, name of table and fields in admin page are shown by Meta class and special functions, fields are attributes of class, their special options like auto increment, not null and others are created by giving data into attributes of Django class object.

So, to create table Goods, I write this code in file models.py:

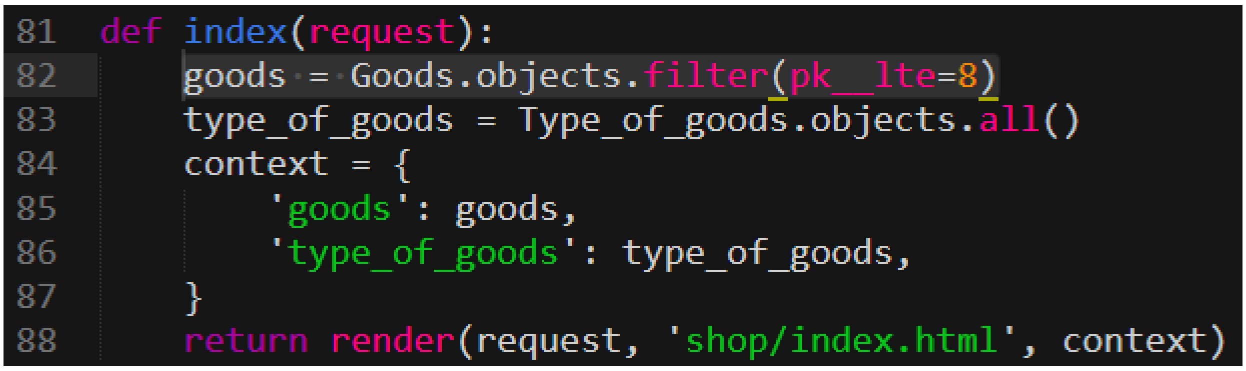


```python
class Goods(models.Model):
    title = models.CharField(max_length=100, verbose_name='Имя товара')
    price = models.IntegerField(verbose_name='Цена')
    ingredients = models.CharField(max_length=300, verbose_name='Ингедиенты', null=True, bla
    photo = models.ImageField(upload_to='photos/%Y/%m/%d', verbose_name='Фото', null=True, bl
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='Дата публикации')
    updated_at = models.DateTimeField(auto_now=True, verbose_name='Дата изменении')
    is_published = models.BooleanField(default=True, verbose_name='Публиковать')
    type_of_goods = models.ForeignKey('Type_of_goods', on_delete=models.PROTECT, null=True,

    class Meta:
        verbose_name = 'Товар'
        verbose_name_plural = 'Товары'
        ordering = ['-created_at']

    def __str__(self):
        return self.title
```

It equals to this SQL code:

```
CREATE TABLE “Goods” (
“id” int(11) NOT NULL AUTO_INCREMENT,
“title” varchar(100) NOT NULL,
“price” int(11) NOT NULL,
“ingredients” varchar(300),
“photo” varchar(256),
“created_at” int(30) NOT NULL,
“updated_at” int(30) NOT NULL,
“is_published” boolean()NOT NULL,
“Type_of_goods” int(11) NOT NULL,
PRIMARY KEY(“id”)
)
```

To select data of table Goods, I write code in file view.py (#82 string):

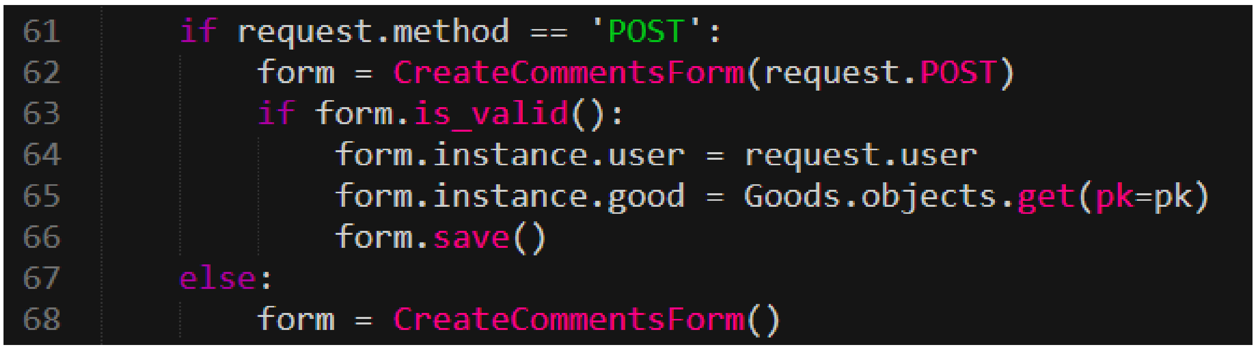


```python
def index(request):
    goods = Goods.objects.filter(pk__lte=8)
    type_of_goods = Type_of_goods.objects.all()
    context = {
        'goods': goods,
        'type_of_goods': type_of_goods,
    }
    return render(request, 'shop/index.html', context)
```

It equals to this SQL code:

SELECT * FROM “Goods” WHERE pk <= 8

To insert data into table Comments, I write code in file view.py :

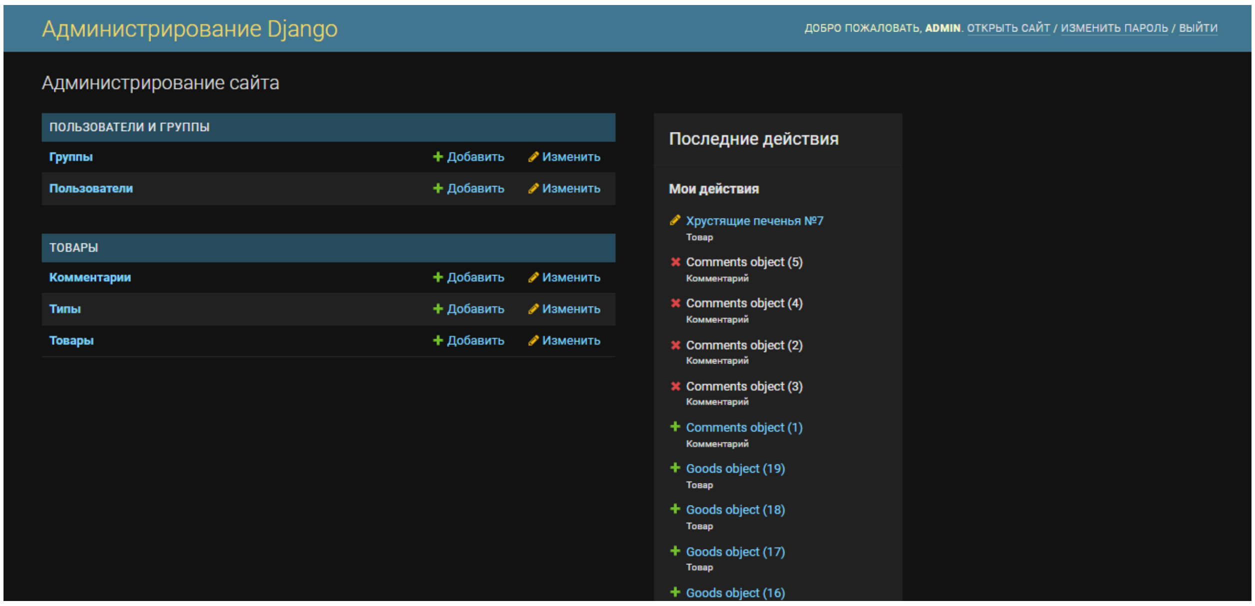


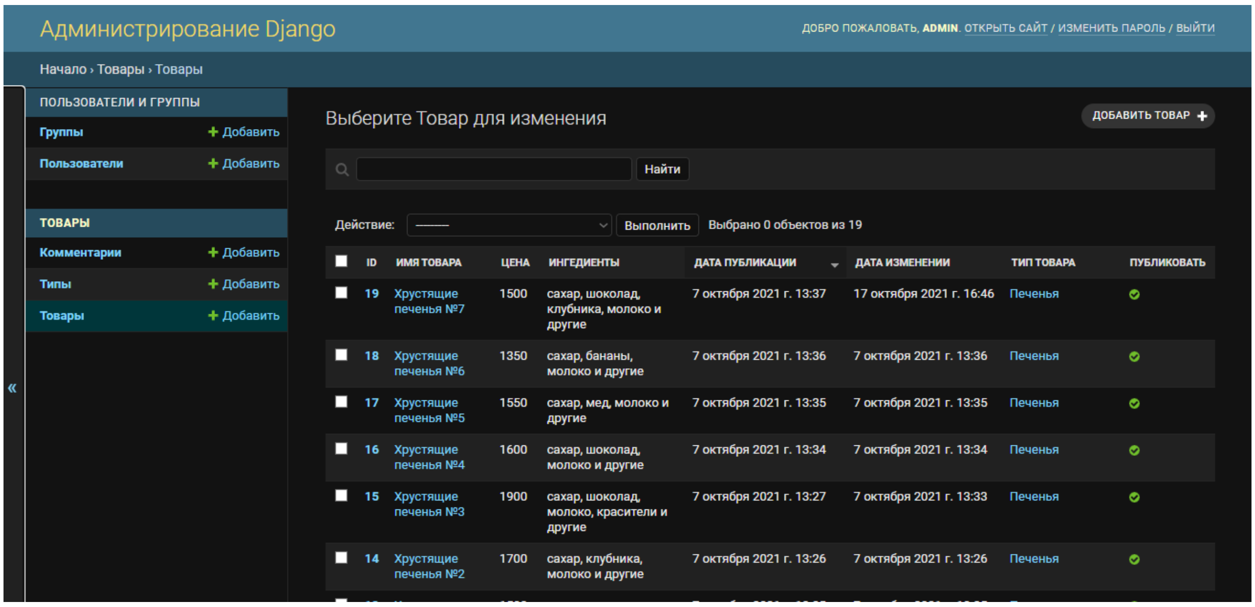
```python
    if request.method == 'POST':
        form = CreateCommentsForm(request.POST)
        if form.is_valid():
            form.instance.user = request.user
            form.instance.good = Goods.objects.get(pk=pk)
            form.save()
    else:
        form = CreateCommentsForm()
```

It equals to this SQL code:

INSERT INTO “Comments” (id, user, good, body, created_at, is_published) VALUES (4, “admin”, “cake#4”, “…”, 45671324, TRUE)

I don’t do deleting or changing forms to remove or to change data. But administer can create, change, select and remove any data in admin page. Because Django has his own admin page which is used by me and is shown below:





Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, **ADMIN**. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Начало › Товары › Товары

ПОЛЬЗОВАТЕЛИ И ГРУППЫ
- Группы + Добавить
- Пользователи + Добавить

ТОВАРЫ
- Комментарии + Добавить
- Типы + Добавить
- Товары + Добавить

Выберите Товар для изменения

ДОБАВИТЬ ТОВАР +

Найти

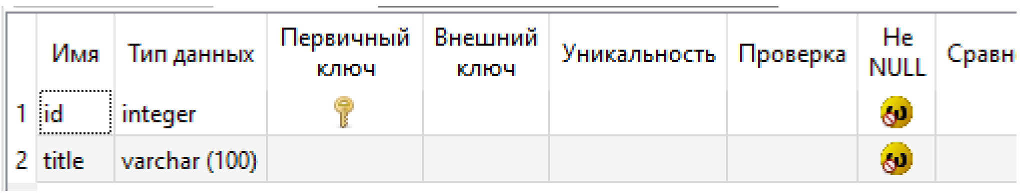
Действие: --------- Выполнить Выбрано 0 объектов из 19

	ID	ИМЯ ТОВАРА	ЦЕНА	ИНГЕДИЕНТЫ	ДАТА ПУБЛИКАЦИИ	ДАТА ИЗМЕНЕНИИ	ТИП ТОВАРА	ПУБЛИКОВАТЬ
☐	19	Хрустящие печенья №7	1500	сахар, шоколад, клубника, молоко и другие	7 октября 2021 г. 13:37	17 октября 2021 г. 16:46	Печенья	✓
☐	18	Хрустящие печенья №6	1350	сахар, бананы, молоко и другие	7 октября 2021 г. 13:36	7 октября 2021 г. 13:36	Печенья	✓
☐	17	Хрустящие печенья №5	1550	сахар, мед, молоко и другие	7 октября 2021 г. 13:35	7 октября 2021 г. 13:35	Печенья	✓
☐	16	Хрустящие печенья №4	1600	сахар, шоколад, молоко и другие	7 октября 2021 г. 13:34	7 октября 2021 г. 13:34	Печенья	✓
☐	15	Хрустящие печенья №3	1900	сахар, шоколад, молоко, красители и другие	7 октября 2021 г. 13:27	7 октября 2021 г. 13:33	Печенья	✓
☐	14	Хрустящие печенья №2	1700	сахар, клубника, молоко и другие	7 октября 2021 г. 13:26	7 октября 2021 г. 13:26	Печенья	✓

4.2 Database structure

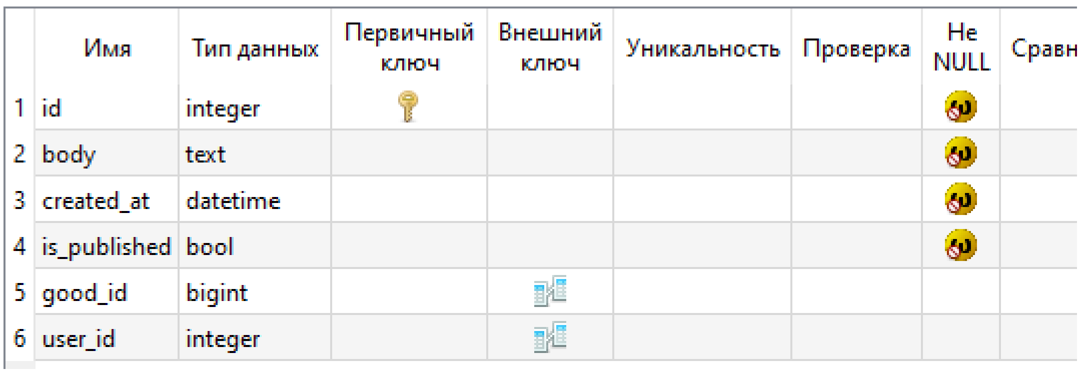
Whole data is saved in file db.sqlite3. To represent database structure, I use SQLiteStudio with version 3.3.2.

<u>Type_of_Goods:</u>



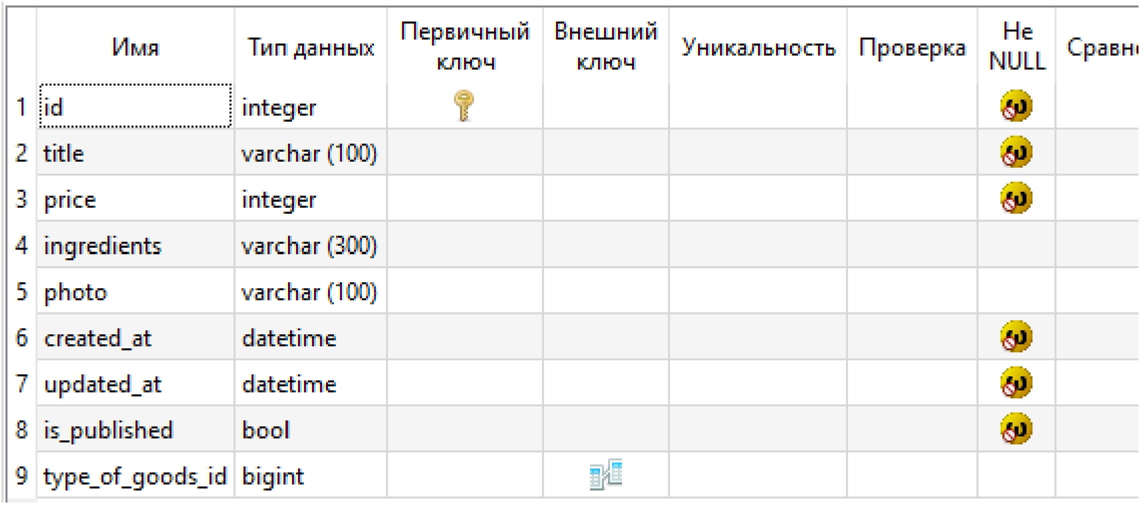
	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравн
1	id	integer	🔑				⛔	
2	title	varchar (100)					⛔	

<u>Comments:</u>



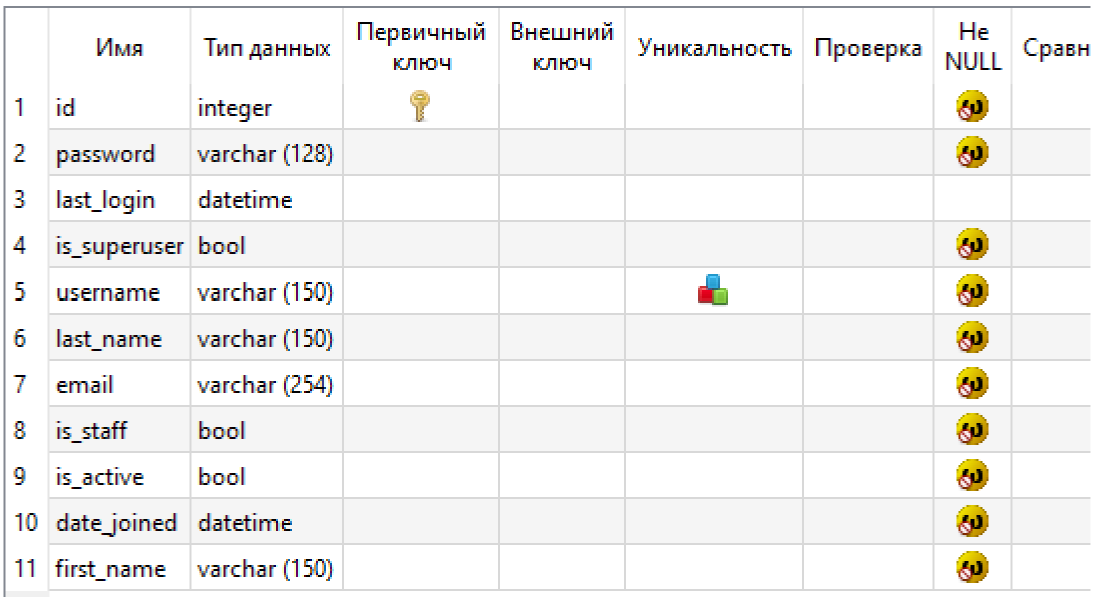
	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравн
1	id	integer	🔑				⛔	
2	body	text					⛔	
3	created_at	datetime					⛔	
4	is_published	bool					⛔	
5	good_id	bigint		🔗				
6	user_id	integer		🔗				

<u>Goods:</u>



	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравн
1	id	integer	🔑				✓	
2	title	varchar (100)					✓	
3	price	integer					✓	
4	ingredients	varchar (300)						
5	photo	varchar (100)						
6	created_at	datetime					✓	
7	updated_at	datetime					✓	
8	is_published	bool					✓	
9	type_of_goods_id	bigint		✓				

<u>User:</u>

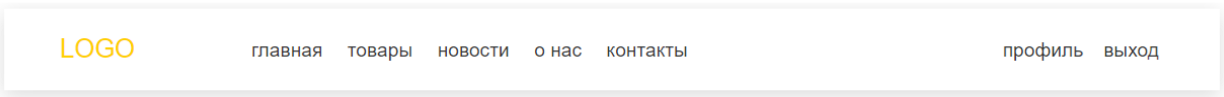


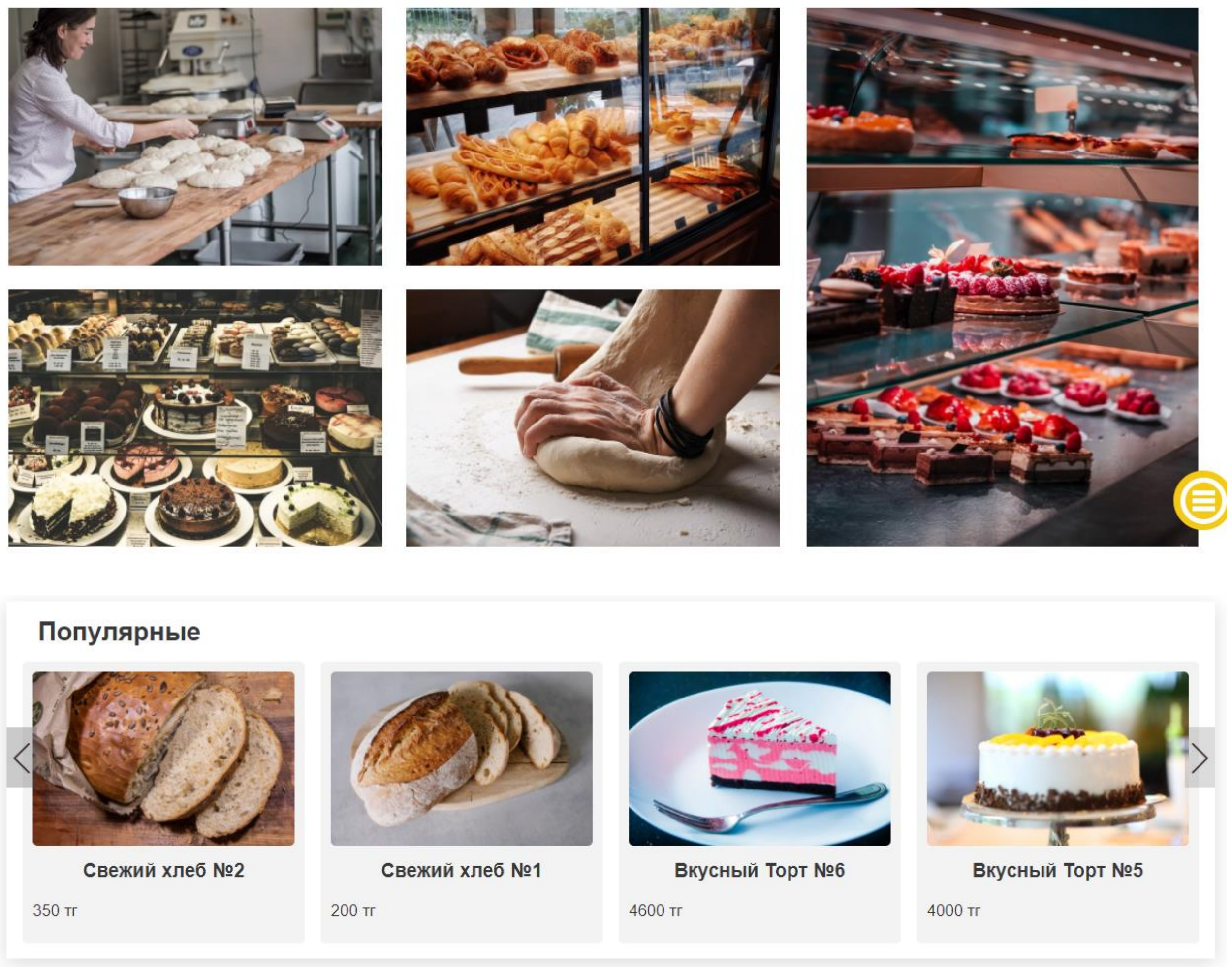
	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравн
1	id	integer	🔑				✓	
2	password	varchar (128)					✓	
3	last_login	datetime						
4	is_superuser	bool					✓	
5	username	varchar (150)			✓		✓	
6	last_name	varchar (150)					✓	
7	email	varchar (254)					✓	
8	is_staff	bool					✓	
9	is_active	bool					✓	
10	date_joined	datetime					✓	
11	first_name	varchar (150)					✓	

4.3 HTML and JavaScript

<u>Home page:</u>

There are header and footer which have links to pages such as shop, account, registration and login; images of bakery and a block with the best products in the website.





Популярные

Свежий хлеб №2

350 тг

Свежий хлеб №1

200 тг

Вкусный Торт №6

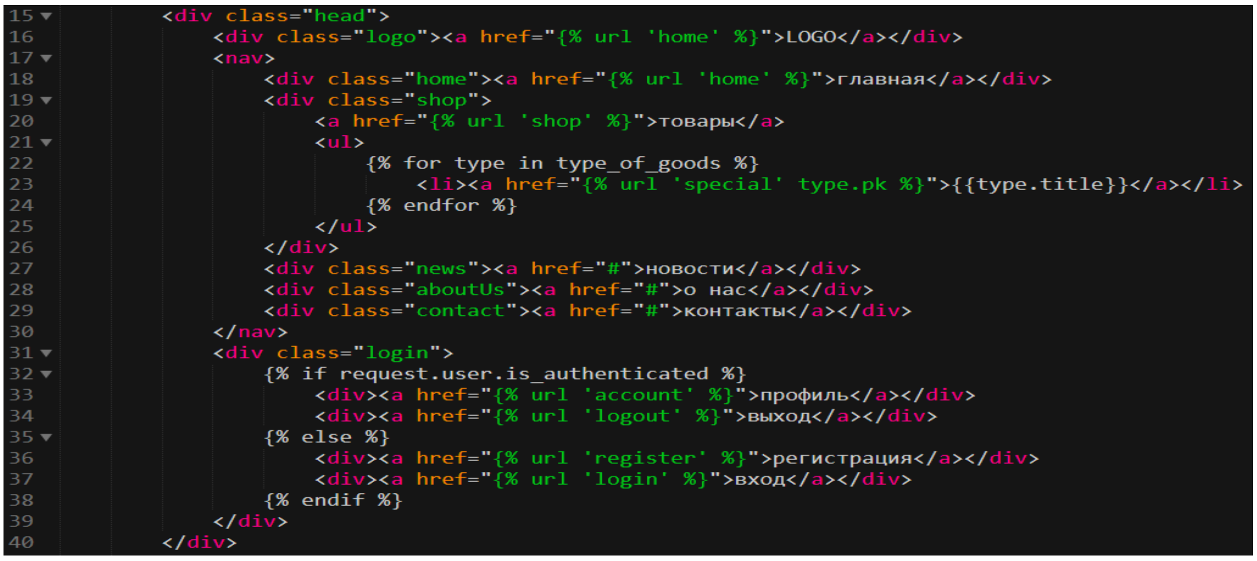
4600 тг

Вкусный Торт №5

4000 тг

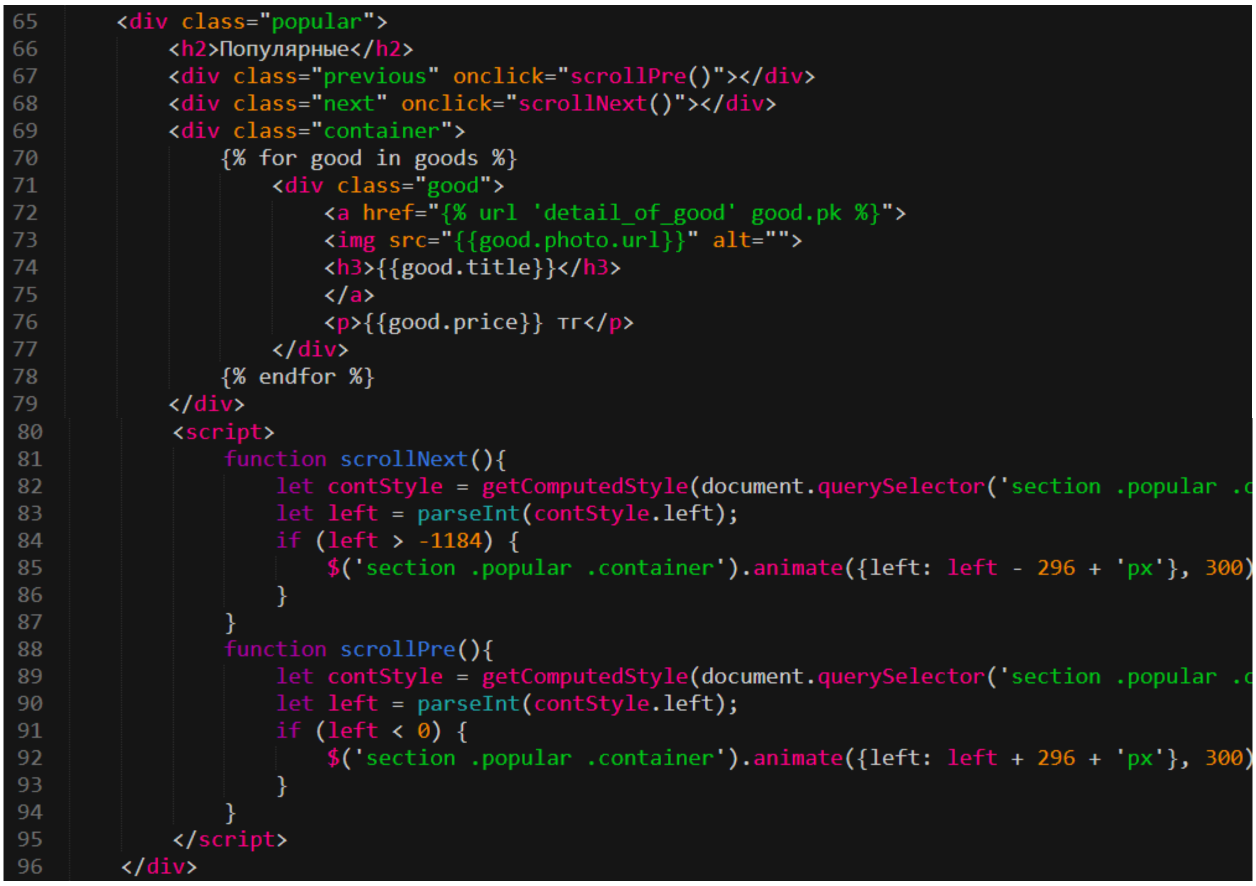
A fragment of HTML code of file base.html and index.html is shown below:

Base.html has a code of header:



```
        <div class="head">
            <div class="logo"><a href="{% url 'home' %}">LOGO</a></div>
            <nav>
                <div class="home"><a href="{% url 'home' %}">главная</a></div>
                <div class="shop">
                    <a href="{% url 'shop' %}">товары</a>
                    <ul>
                        {% for type in type_of_goods %}
                            <li><a href="{% url 'special' type.pk %}">{{type.title}}</a></li>
                        {% endfor %}
                    </ul>
                </div>
                <div class="news"><a href="#">новости</a></div>
                <div class="aboutUs"><a href="#">о нас</a></div>
                <div class="contact"><a href="#">контакты</a></div>
            </nav>
            <div class="login">
                {% if request.user.is_authenticated %}
                    <div><a href="{% url 'account' %}">профиль</a></div>
                    <div><a href="{% url 'logout' %}">выход</a></div>
                {% else %}
                    <div><a href="{% url 'register' %}">регистрация</a></div>
                    <div><a href="{% url 'login' %}">вход</a></div>
                {% endif %}
            </div>
        </div>
```

Index.html has a code of block with the best products:

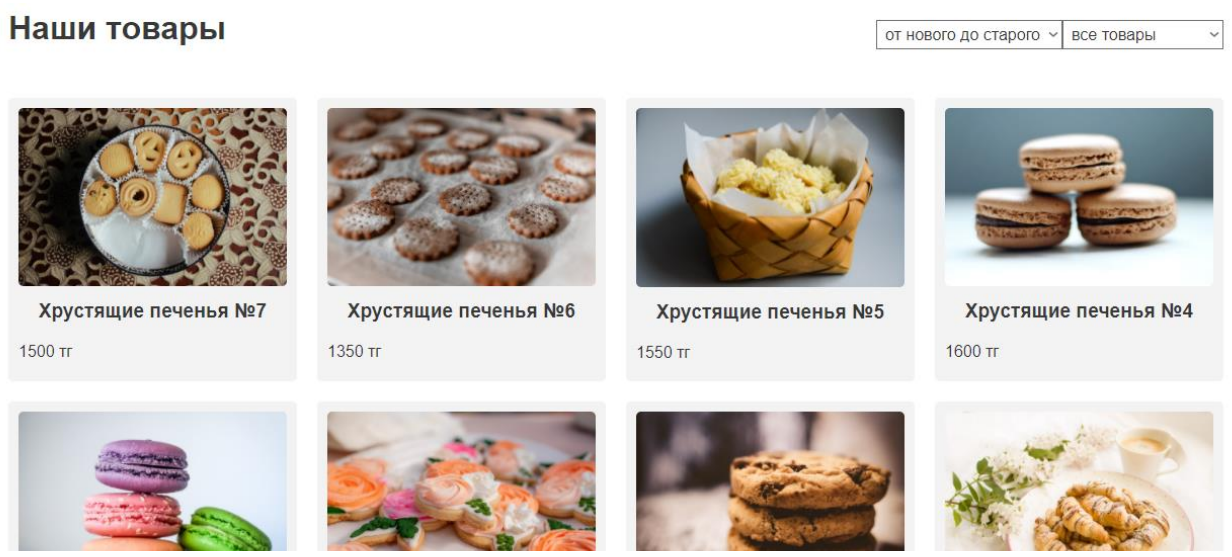


```
    <div class="popular">
        <h2>Популярные</h2>
        <div class="previous" onclick="scrollPre()"></div>
        <div class="next" onclick="scrollNext()"></div>
        <div class="container">
            {% for good in goods %}
                <div class="good">
                    <a href="{% url 'detail_of_good' good.pk %}">
                    <img src="{{good.photo.url}}" alt="">
                    <h3>{{good.title}}</h3>
                    </a>
                    <p>{{good.price}} тг</p>
                </div>
            {% endfor %}
        </div>
        <script>
            function scrollNext(){
                let contStyle = getComputedStyle(document.querySelector('section .popular .c
                let left = parseInt(contStyle.left);
                if (left > -1184) {
                    $('section .popular .container').animate({left: left - 296 + 'px'}, 300
                }
            }
            function scrollPre(){
                let contStyle = getComputedStyle(document.querySelector('section .popular .c
                let left = parseInt(contStyle.left);
                if (left < 0) {
                    $('section .popular .container').animate({left: left + 296 + 'px'}, 300
                }
            }
        </script>
    </div>
```

Django has technology that makes my html code to be shorter and easy to write by joining base template with other template.

<u>Shop page:</u>

It contains all products that are had in website. In addition, there are sorting and filtering forms. A code of the page is written in files base.html and shop.html:



Наши товары

от нового до старого | все товары

Хрустящие печенья №7
1500 тг

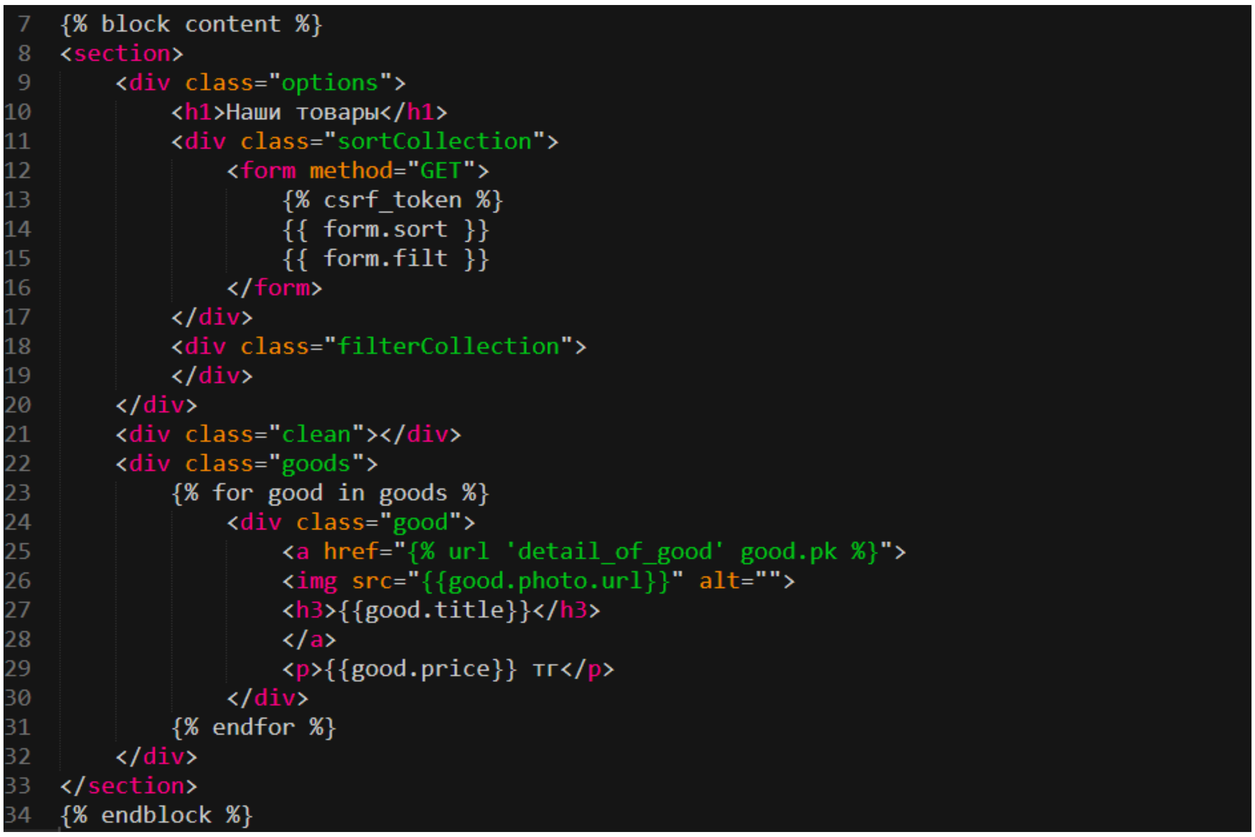
Хрустящие печенья №6
1350 тг

Хрустящие печенья №5
1550 тг

Хрустящие печенья №4
1600 тг

File base.html only has header and footer of all pages of website as long as they have header and footer.

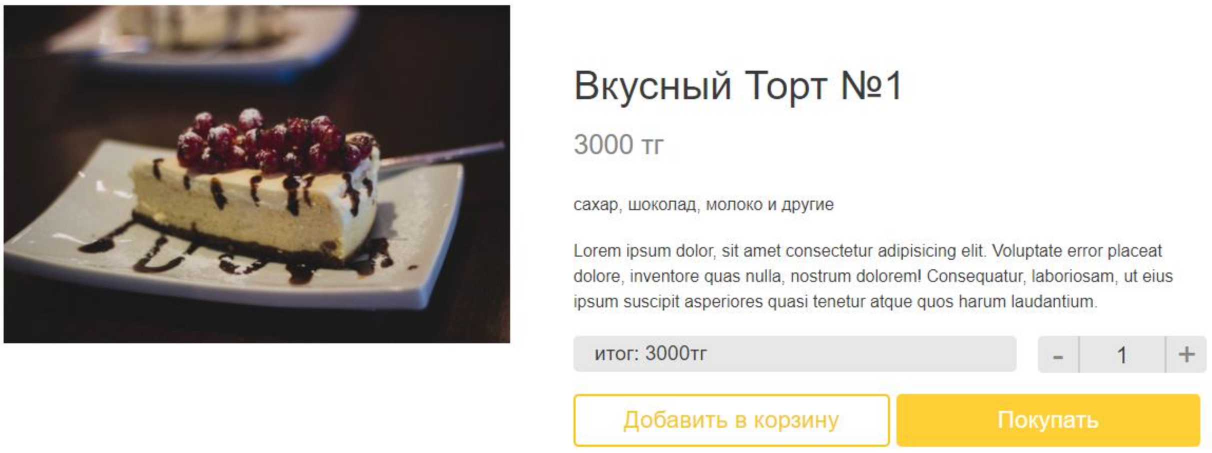
File shop.html has blocks of products and sorting and filtering forms as select input:



```
{% block content %}
<section>
    <div class="options">
        <h1>Наши товары</h1>
        <div class="sortCollection">
            <form method="GET">
                {% csrf_token %}
                {{ form.sort }}
                {{ form.filt }}
            </form>
        </div>
        <div class="filterCollection">
        </div>
    </div>
    <div class="clean"></div>
    <div class="goods">
        {% for good in goods %}
            <div class="good">
                <a href="{% url 'detail_of_good' good.pk %}">
                <img src="{{good.photo.url}}" alt="">
                <h3>{{good.title}}</h3>
                </a>
                <p>{{good.price}} тг</p>
            </div>
        {% endfor %}
    </div>
</section>
{% endblock %}
```

<u>Product page:</u>

There are all information of chosen product and button that is used to transfer the product to account page. In addition, registered users can send comment by comment form.

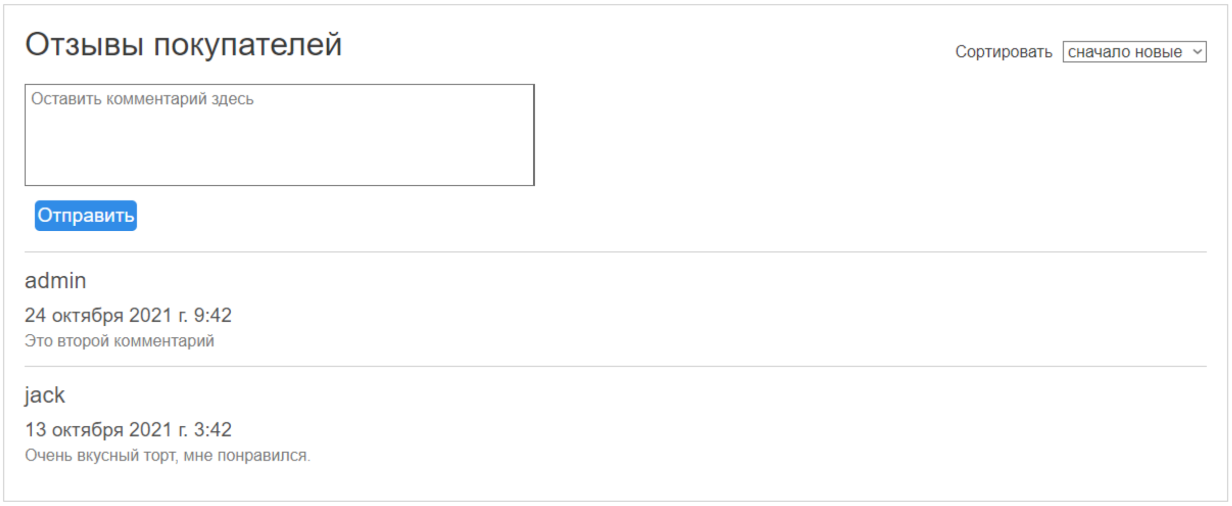


Вкусный Торт №1

3000 тг

сахар, шоколад, молоко и другие

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Voluptate error placeat dolore, inventore quas nulla, nostrum dolorem! Consequatur, laboriosam, ut eius ipsum suscipit asperiores quasi tenetur atque quos harum laudantium.



Отзывы покупателей

Сортировать сначало новые

Оставить комментарий здесь

Отправить

admin

24 октября 2021 г. 9:42

Это второй комментарий

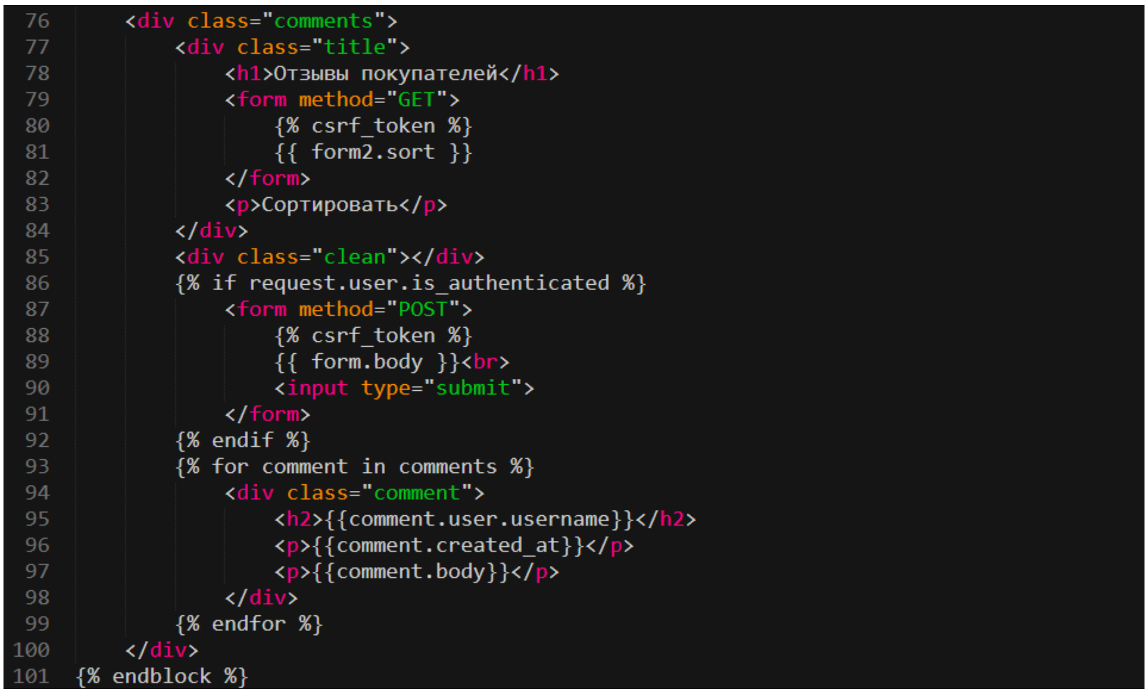
jack

13 октября 2021 г. 3:42

Очень вкусный торт, мне понравился.

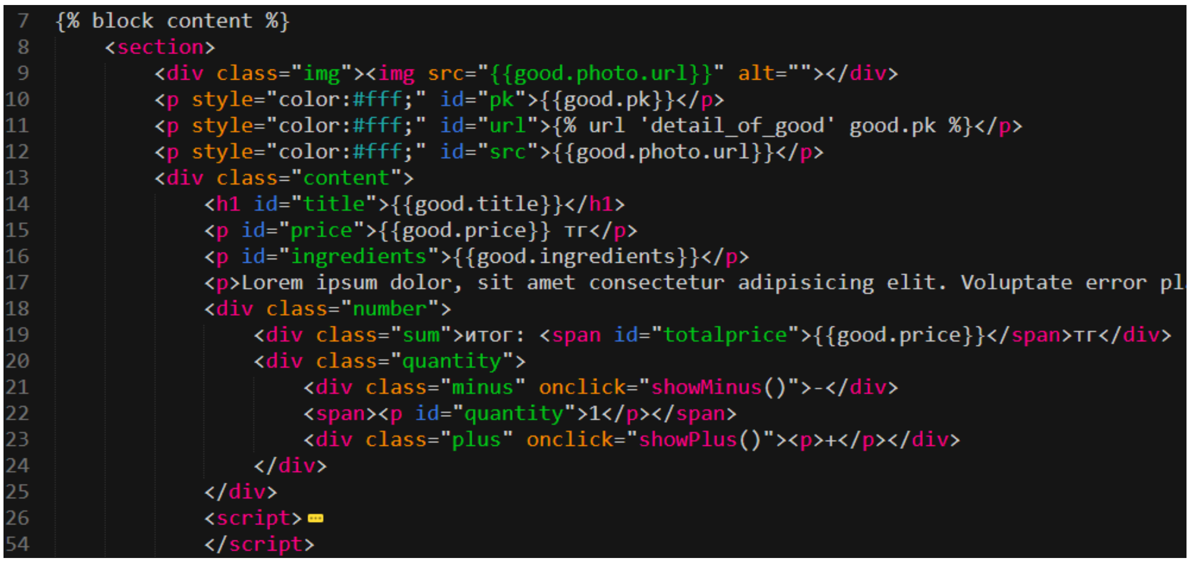
The page is created by files base.html and details.html.

Sorting and comment form:



```
    <div class="comments">
        <div class="title">
            <h1>Отзывы покупателей</h1>
            <form method="GET">
                {% csrf_token %}
                {{ form2.sort }}
            </form>
            <p>Сортировать</p>
        </div>
        <div class="clean"></div>
        {% if request.user.is_authenticated %}
            <form method="POST">
                {% csrf_token %}
                {{ form.body }}<br>
                <input type="submit">
            </form>
        {% endif %}
        {% for comment in comments %}
            <div class="comment">
                <h2>{{comment.user.username}}</h2>
                <p>{{comment.created_at}}</p>
                <p>{{comment.body}}</p>
            </div>
        {% endfor %}
    </div>
{% endblock %}
```

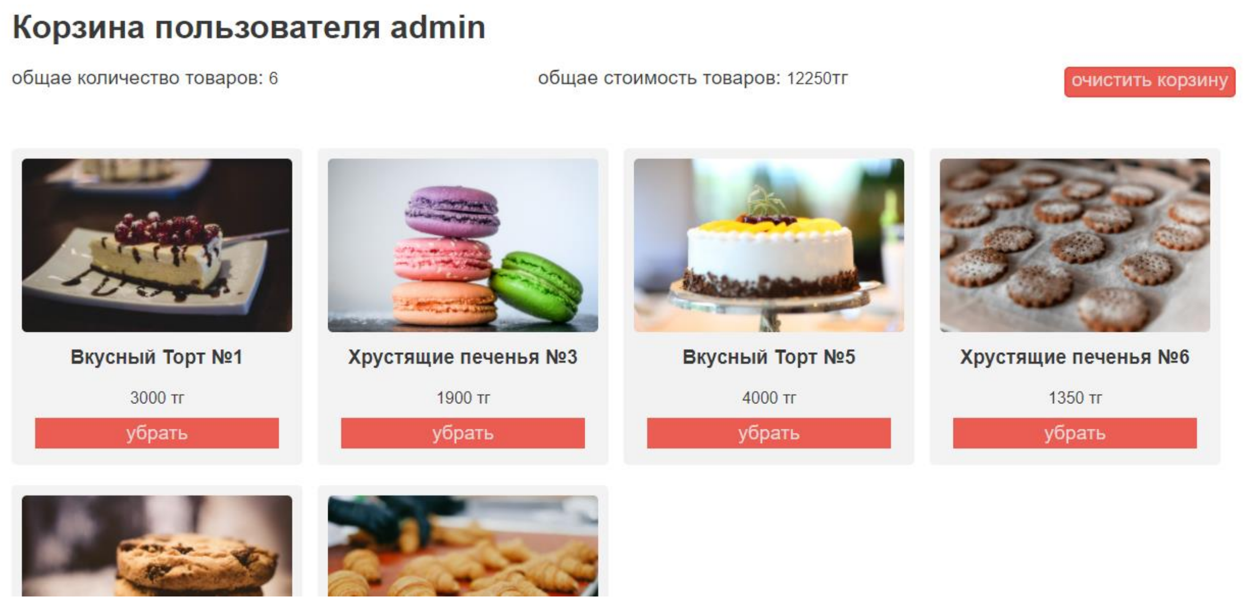
Data of product as image, price and name; counting forms and buttons buy and to cart (account):



```
{% block content %}
    <section>
        <div class="img"><img src="{{good.photo.url}}" alt=""></div>
        <p style="color:#fff;" id="pk">{{good.pk}}</p>
        <p style="color:#fff;" id="url">{% url 'detail_of_good' good.pk %}</p>
        <p style="color:#fff;" id="src">{{good.photo.url}}</p>
        <div class="content">
            <h1 id="title">{{good.title}}</h1>
            <p id="price">{{good.price}} тг</p>
            <p id="ingredients">{{good.ingredients}}</p>
            <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Voluptate error pl
            <div class="number">
                <div class="sum">итог: <span id="totalprice">{{good.price}}</span>тг</div>
                <div class="quantity">
                    <div class="minus" onclick="showMinus()">-</div>
                    <span><p id="quantity">1</p></span>
                    <div class="plus" onclick="showPlus()"><p>+</p></div>
                </div>
            </div>
            <script>
            </script>
```

Account page:

It contains products that user press the button “Добавить в корзину” in their page. There are special button that can delete products from the page and labels that shows general data like price and count.

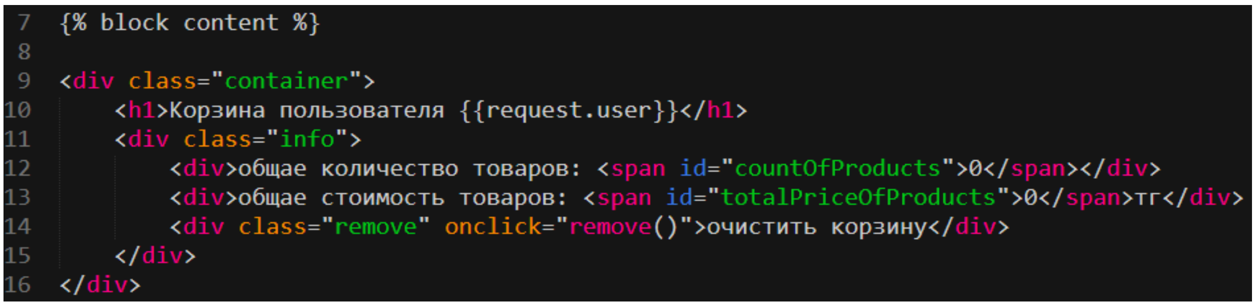


Корзина пользователя admin

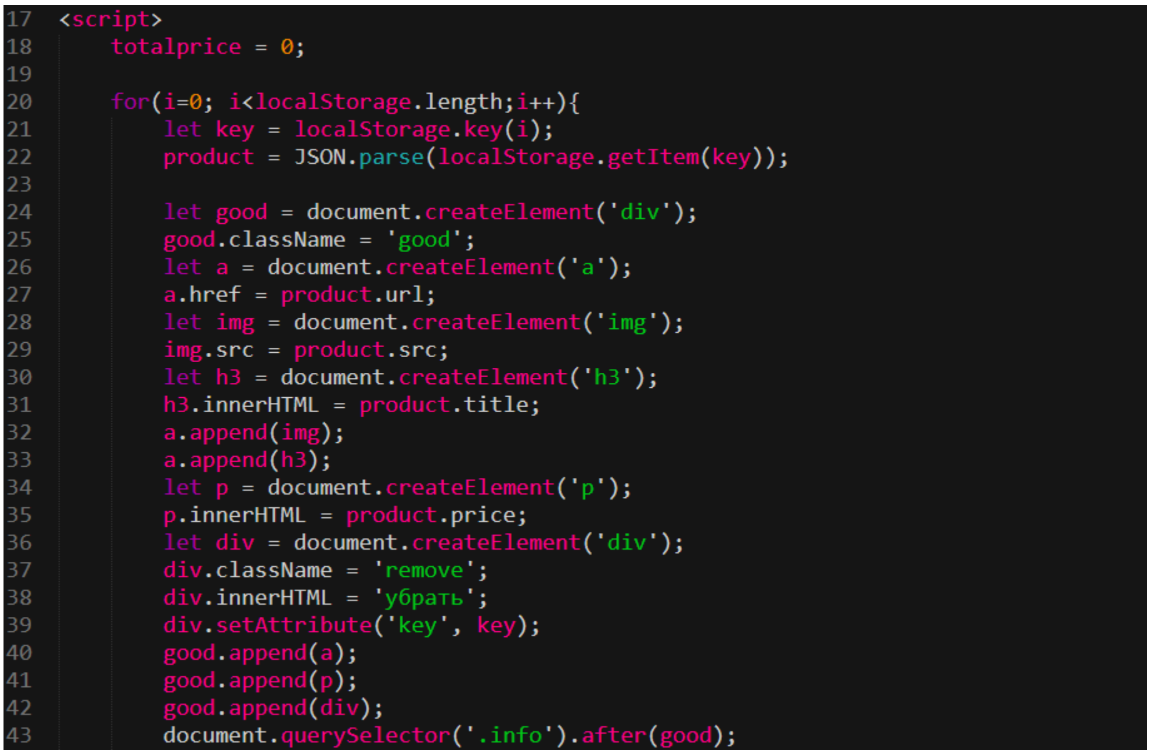
общае количество товаров: 6

общае стоимость товаров: 12250тг

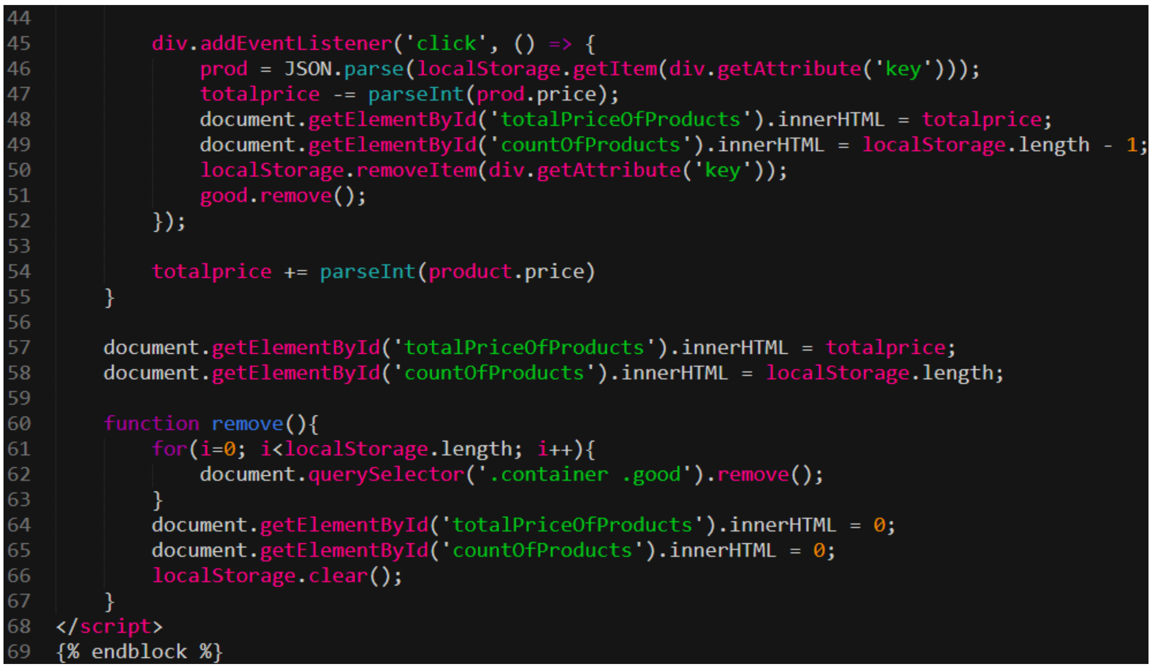
It uses base.html and account.html.



```
{% block content %}

<div class="container">
    <h1>Корзина пользователя {{request.user}}</h1>
    <div class="info">
        <div>общае количество товаров: <span id="countOfProducts">0</span></div>
        <div>общае стоимость товаров: <span id="totalPriceOfProducts">0</span>тг</div>
        <div class="remove" onclick="remove()">очистить корзину</div>
    </div>
</div>
```



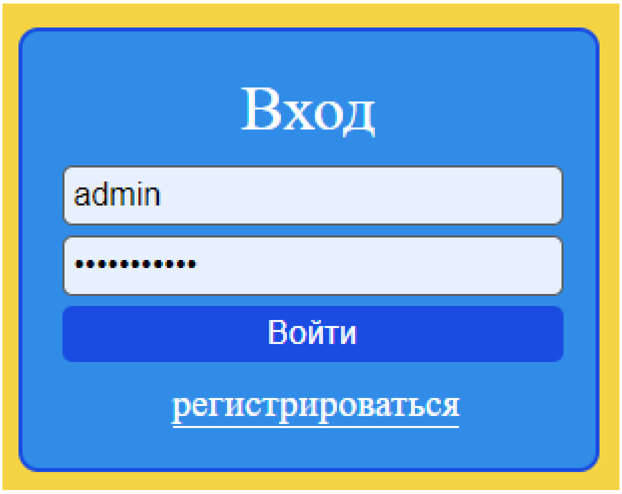
```
<script>
    totalprice = 0;

    for(i=0; i<localStorage.length;i++){
        let key = localStorage.key(i);
        product = JSON.parse(localStorage.getItem(key));

        let good = document.createElement('div');
        good.className = 'good';
        let a = document.createElement('a');
        a.href = product.url;
        let img = document.createElement('img');
        img.src = product.src;
        let h3 = document.createElement('h3');
        h3.innerHTML = product.title;
        a.append(img);
        a.append(h3);
        let p = document.createElement('p');
        p.innerHTML = product.price;
        let div = document.createElement('div');
        div.className = 'remove';
        div.innerHTML = 'убрать';
        div.setAttribute('key', key);
        good.append(a);
        good.append(p);
        good.append(div);
        document.querySelector('.info').after(good);
```



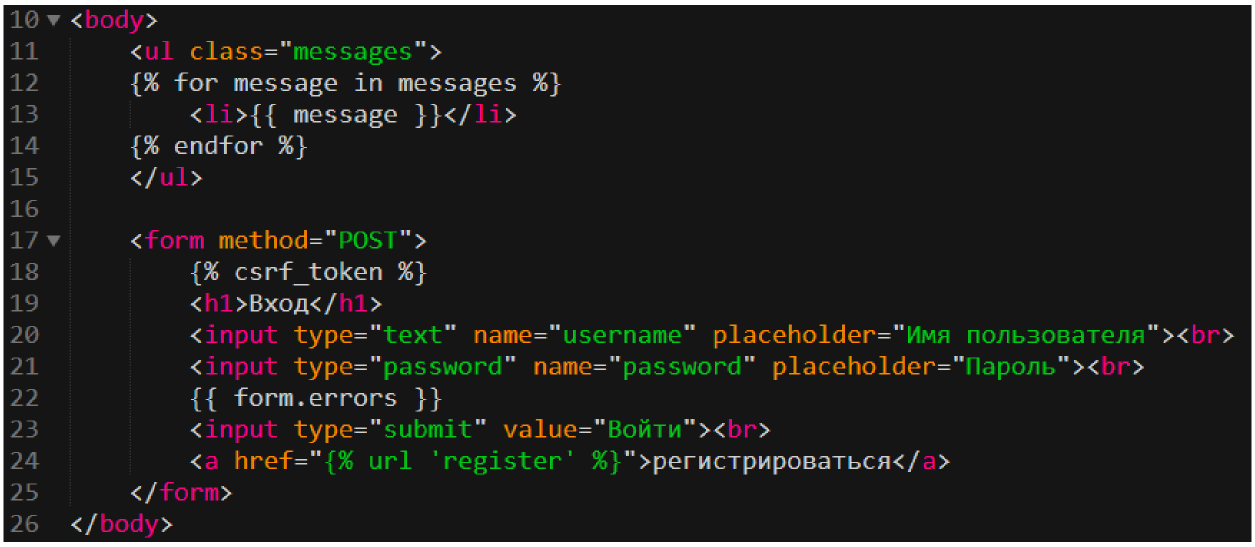
```
        div.addEventListener('click', () => {
            prod = JSON.parse(localStorage.getItem(div.getAttribute('key')));
            totalprice -= parseInt(prod.price);
            document.getElementById('totalPriceOfProducts').innerHTML = totalprice;
            document.getElementById('countOfProducts').innerHTML = localStorage.length - 1;
            localStorage.removeItem(div.getAttribute('key'));
            good.remove();
        });

        totalprice += parseInt(product.price)
    }

    document.getElementById('totalPriceOfProducts').innerHTML = totalprice;
    document.getElementById('countOfProducts').innerHTML = localStorage.length;

    function remove(){
        for(i=0; i<localStorage.length; i++){
            document.querySelector('.container .good').remove();
        }
        document.getElementById('totalPriceOfProducts').innerHTML = 0;
        document.getElementById('countOfProducts').innerHTML = 0;
        localStorage.clear();
    }
</script>
{% endblock %}
```

Login page:

There are only login form and message box if user give invalid data.



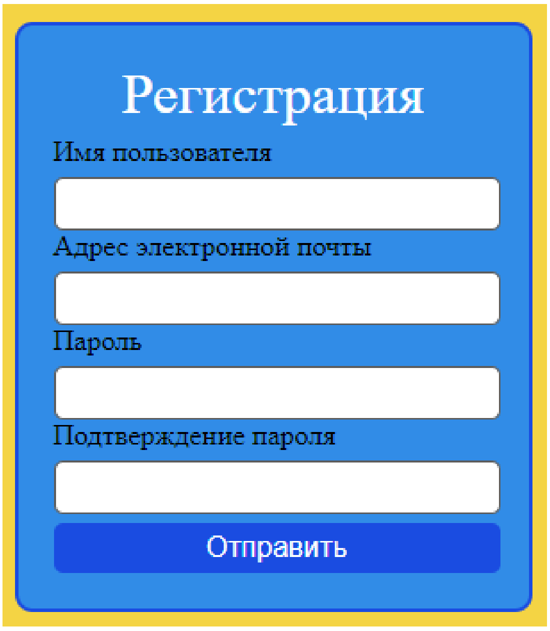
The page use only login.html



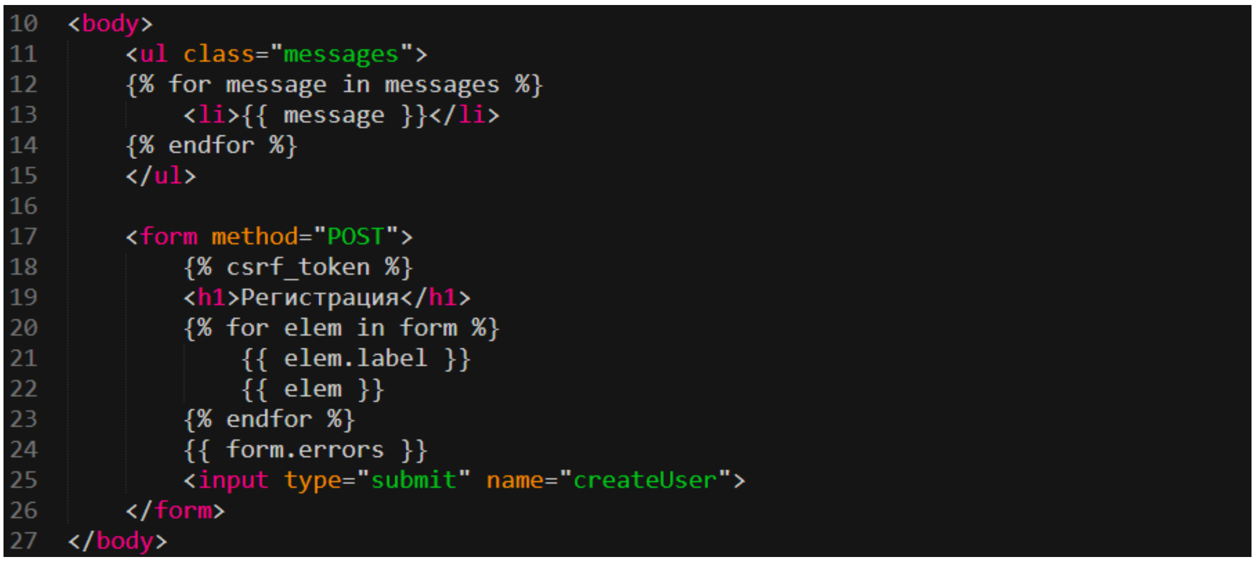
```
<body>
    <ul class="messages">
    {% for message in messages %}
        <li>{{ message }}</li>
    {% endfor %}
    </ul>

    <form method="POST">
        {% csrf_token %}
        <h1>Вход</h1>
        <input type="text" name="username" placeholder="Имя пользователя"><br>
        <input type="password" name="password" placeholder="Пароль"><br>
        {{ form.errors }}
        <input type="submit" value="Войти"><br>
        <a href="{% url 'register' %}">регистрироваться</a>
    </form>
</body>
```

Register page:

There are only register form and message box if user give invalid data.



The page use only register.html.



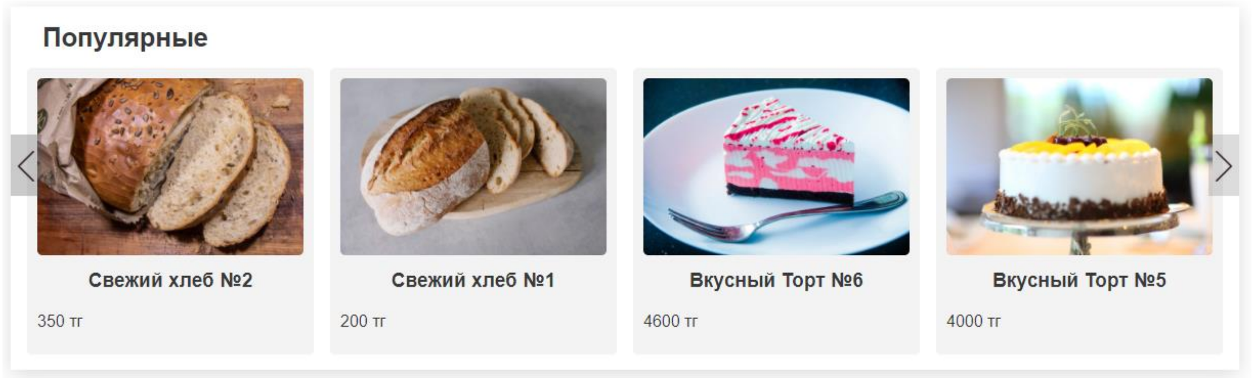
```
<body>
    <ul class="messages">
    {% for message in messages %}
        <li>{{ message }}</li>
    {% endfor %}
    </ul>

    <form method="POST">
        {% csrf_token %}
        <h1>Регистрация</h1>
        {% for elem in form %}
            {{ elem.label }}
            {{ elem }}
        {% endfor %}
        {{ form.errors }}
        <input type="submit" name="createUser">
    </form>
</body>
```

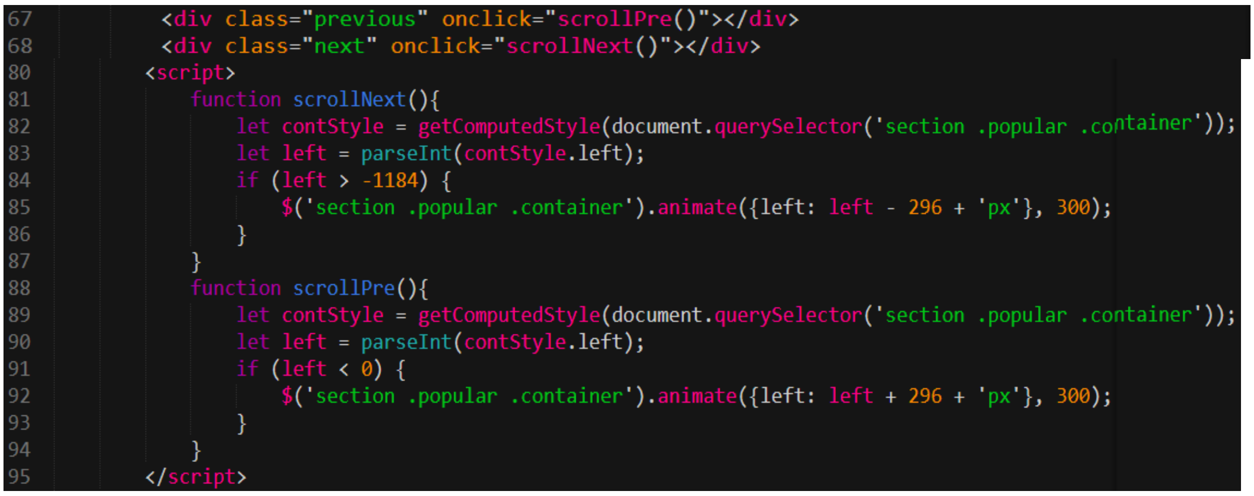
5.0 Programming

5.1 JavaScript

<u>5.1.1 Home page:</u>

In index.html, there are block “Популярные” which contain 8 product blocks and 2 buttons. User can observe only 4 product blocks at once, but product blocks manage to move to left side when user presses the button with right direction. After that user is able to see the 5th product block while the 1st product block hides in the left side of the main block. To watch previous product blocks, user presses the button with left direction.



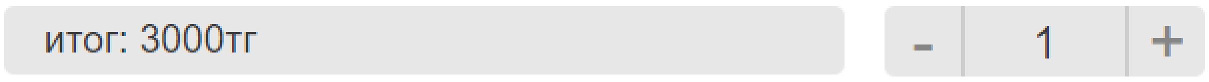


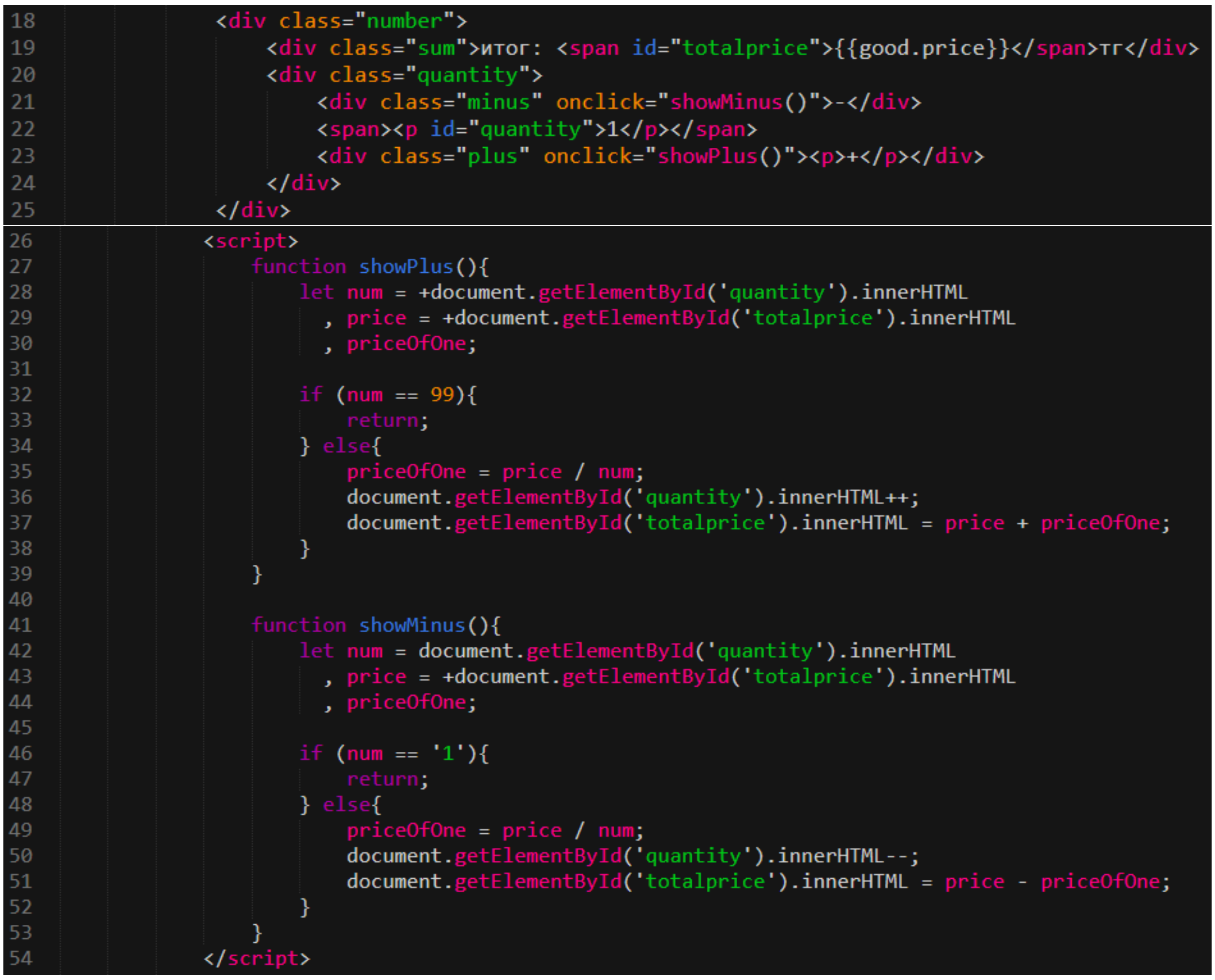
```
        <div class="previous" onclick="scrollPre()"></div>
        <div class="next" onclick="scrollNext()"></div>
      <script>
          function scrollNext(){
              let contStyle = getComputedStyle(document.querySelector('section .popular .container'));
              let left = parseInt(contStyle.left);
              if (left > -1184) {
                  $('section .popular .container').animate({left: left - 296 + 'px'}, 300);
              }
          }
          function scrollPre(){
              let contStyle = getComputedStyle(document.querySelector('section .popular .container'));
              let left = parseInt(contStyle.left);
              if (left < 0) {
                  $('section .popular .container').animate({left: left + 296 + 'px'}, 300);
              }
          }
      </script>
```

<u>5.1.2 Product page:</u>

In details.html, there are two different functions which are written by JS:

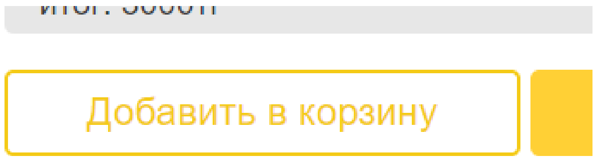
The first is to count number of product and to output its total price. There are label panel which represents total price and counting panel which contains minus and plus button and output label.

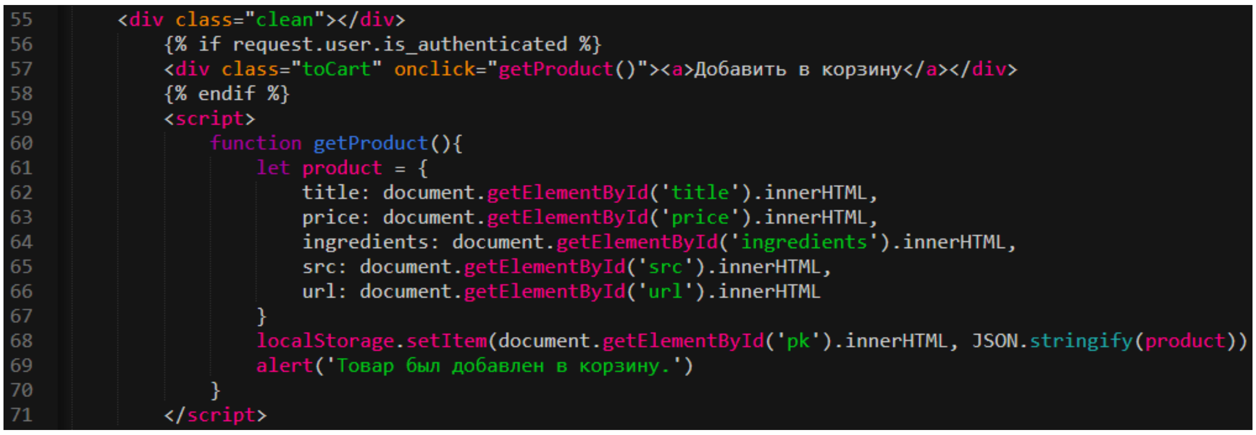




```
        <div class="number">
            <div class="sum">итог: <span id="totalprice">{{good.price}}</span>тг</div>
            <div class="quantity">
                <div class="minus" onclick="showMinus()">-</div>
                <span><p id="quantity">1</p></span>
                <div class="plus" onclick="showPlus()"><p>+</p></div>
            </div>
        </div>
        <script>
            function showPlus(){
                let num = +document.getElementById('quantity').innerHTML
                  , price = +document.getElementById('totalprice').innerHTML
                  , priceOfOne;

                if (num == 99){
                    return;
                } else{
                    priceOfOne = price / num;
                    document.getElementById('quantity').innerHTML++;
                    document.getElementById('totalprice').innerHTML = price + priceOfOne;
                }
            }

            function showMinus(){
                let num = document.getElementById('quantity').innerHTML
                  , price = +document.getElementById('totalprice').innerHTML
                  , priceOfOne;

                if (num == '1'){
                    return;
                } else{
                    priceOfOne = price / num;
                    document.getElementById('quantity').innerHTML--;
                    document.getElementById('totalprice').innerHTML = price - priceOfOne;
                }
            }
        </script>
```

The second is to save all data of the product into local store of user's browser in order to put block of chosen product in user's account then. There is the button "Добавить в корзину" to execute the JS code.

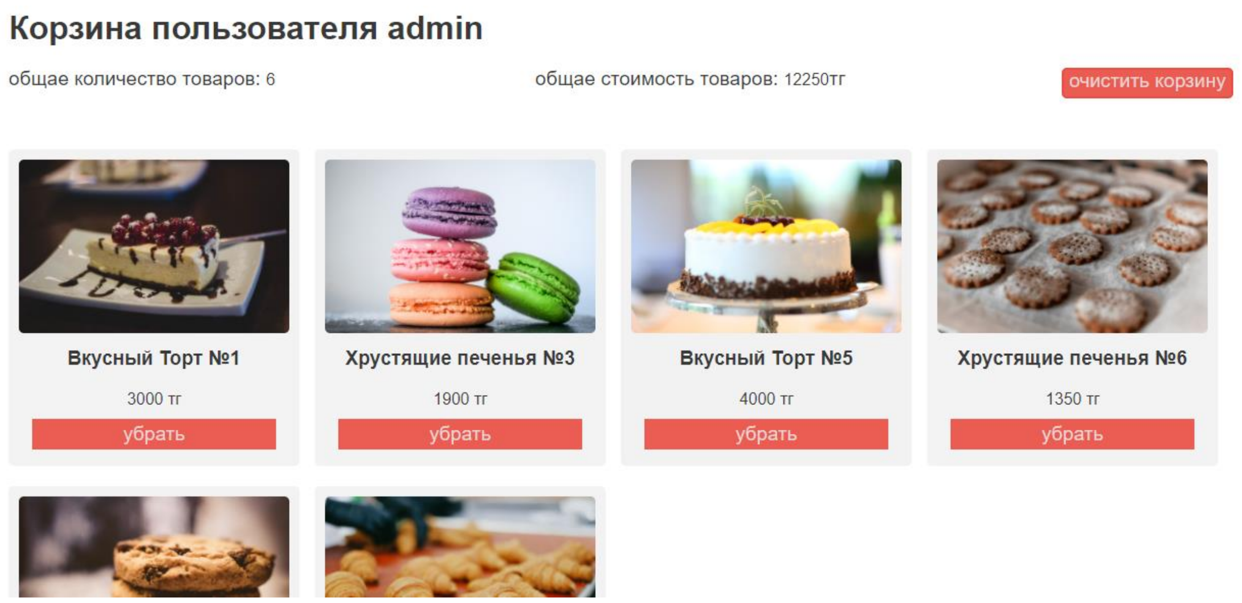




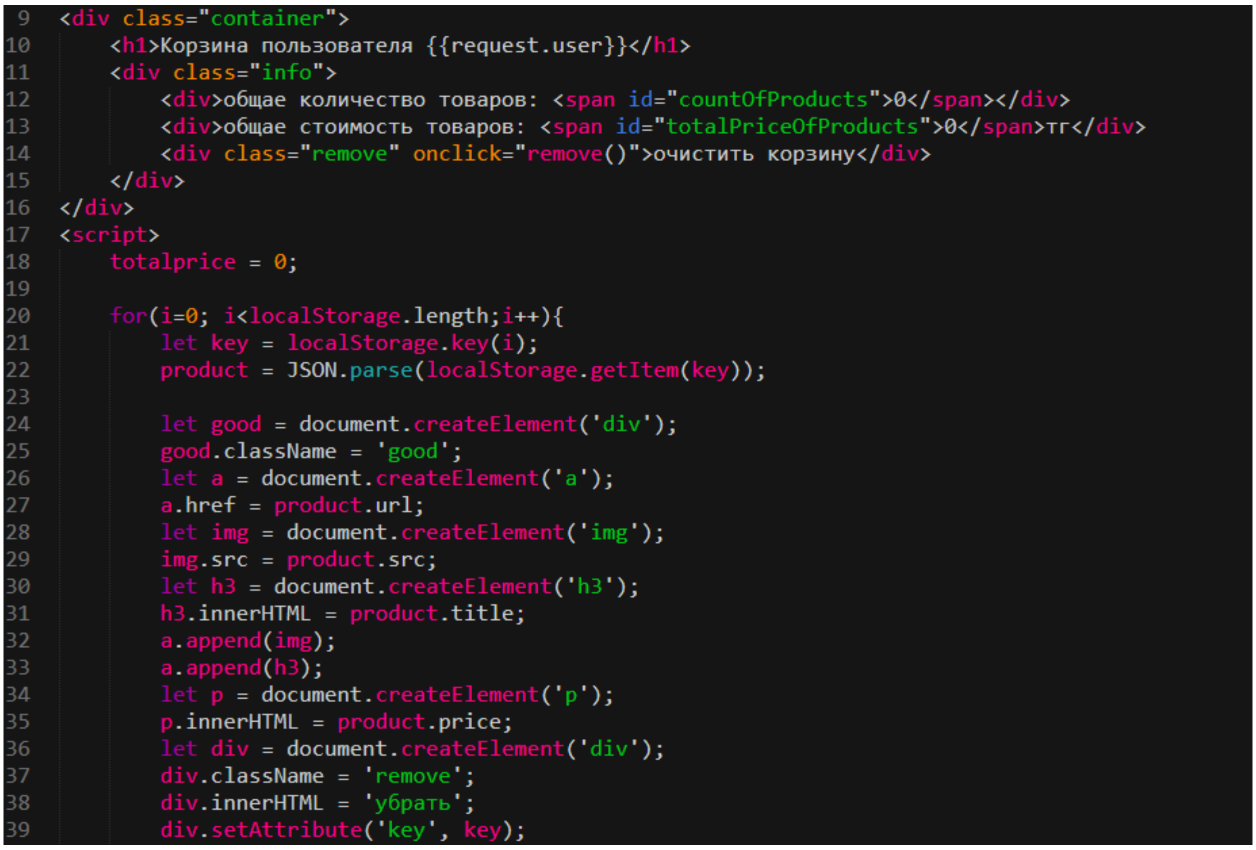
```
    <div class="clean"></div>
        {% if request.user.is_authenticated %}
        <div class="toCart" onclick="getProduct()"><a>Добавить в корзину</a></div>
        {% endif %}
        <script>
            function getProduct(){
                let product = {
                    title: document.getElementById('title').innerHTML,
                    price: document.getElementById('price').innerHTML,
                    ingredients: document.getElementById('ingredients').innerHTML,
                    src: document.getElementById('src').innerHTML,
                    url: document.getElementById('url').innerHTML
                }
                localStorage.setItem(document.getElementById('pk').innerHTML, JSON.stringify(product))
                alert('Товар был добавлен в корзину.')
            }
        </script>
```

5.1.3 Account page:

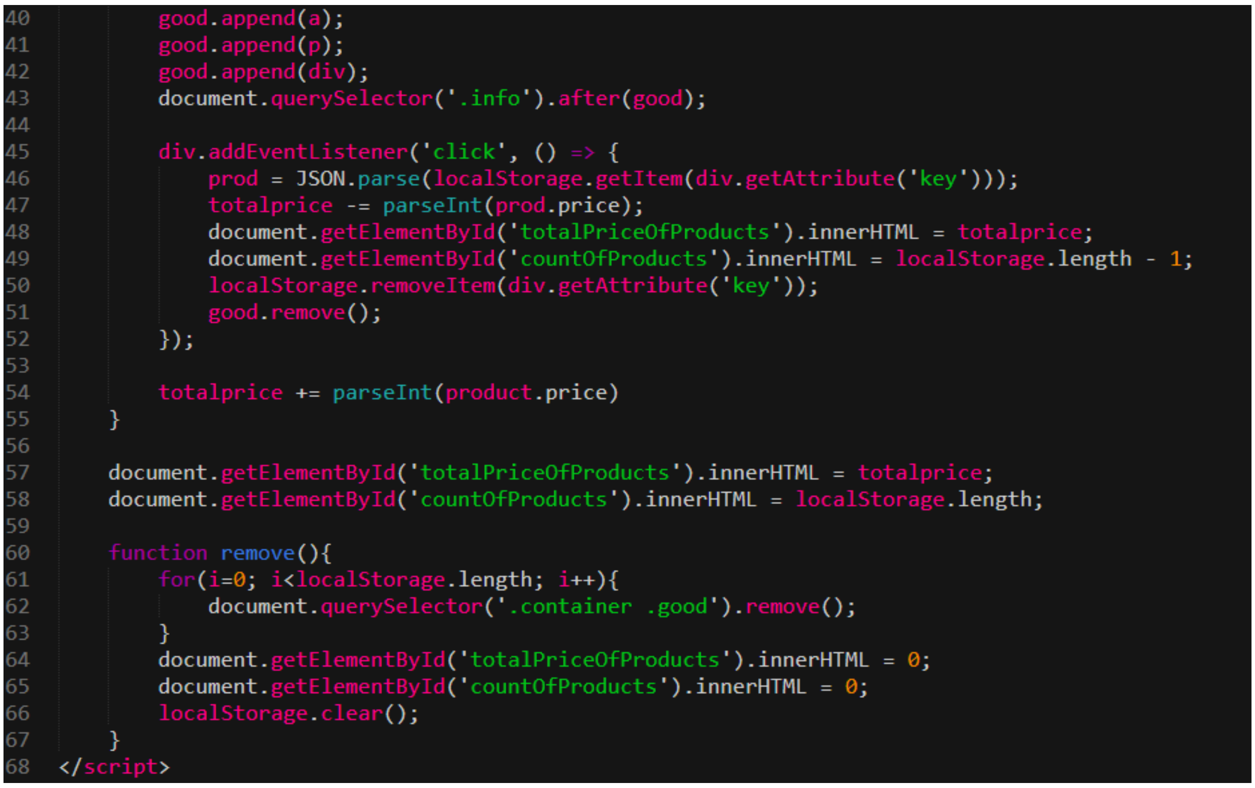
In account.html, there are 2 output labels, remove button to clean whole page and product blocks.



At the first, data about products is received from local store and represented like simple product blocks with extra button that is used to remove targeted block. Secondly, number of products and their total price are counted and are represented in page.



```
<div class="container">
    <h1>Корзина пользователя {{request.user}}</h1>
    <div class="info">
        <div>общае количество товаров: <span id="countOfProducts">0</span></div>
        <div>общае стоимость товаров: <span id="totalPriceOfProducts">0</span>тг</div>
        <div class="remove" onclick="remove()">очистить корзину</div>
    </div>
</div>
<script>
    totalprice = 0;

    for(i=0; i<localStorage.length;i++){
        let key = localStorage.key(i);
        product = JSON.parse(localStorage.getItem(key));

        let good = document.createElement('div');
        good.className = 'good';
        let a = document.createElement('a');
        a.href = product.url;
        let img = document.createElement('img');
        img.src = product.src;
        let h3 = document.createElement('h3');
        h3.innerHTML = product.title;
        a.append(img);
        a.append(h3);
        let p = document.createElement('p');
        p.innerHTML = product.price;
        let div = document.createElement('div');
        div.className = 'remove';
        div.innerHTML = 'убрать';
        div.setAttribute('key', key);
```



```
        good.append(a);
        good.append(p);
        good.append(div);
        document.querySelector('.info').after(good);

        div.addEventListener('click', () => {
            prod = JSON.parse(localStorage.getItem(div.getAttribute('key')));
            totalprice -= parseInt(prod.price);
            document.getElementById('totalPriceOfProducts').innerHTML = totalprice;
            document.getElementById('countOfProducts').innerHTML = localStorage.length - 1;
            localStorage.removeItem(div.getAttribute('key'));
            good.remove();
        });

        totalprice += parseInt(product.price)
    }

    document.getElementById('totalPriceOfProducts').innerHTML = totalprice;
    document.getElementById('countOfProducts').innerHTML = localStorage.length;

    function remove(){
        for(i=0; i<localStorage.length; i++){
            document.querySelector('.container .good').remove();
        }
        document.getElementById('totalPriceOfProducts').innerHTML = 0;
        document.getElementById('countOfProducts').innerHTML = 0;
        localStorage.clear();
    }
</script>
```

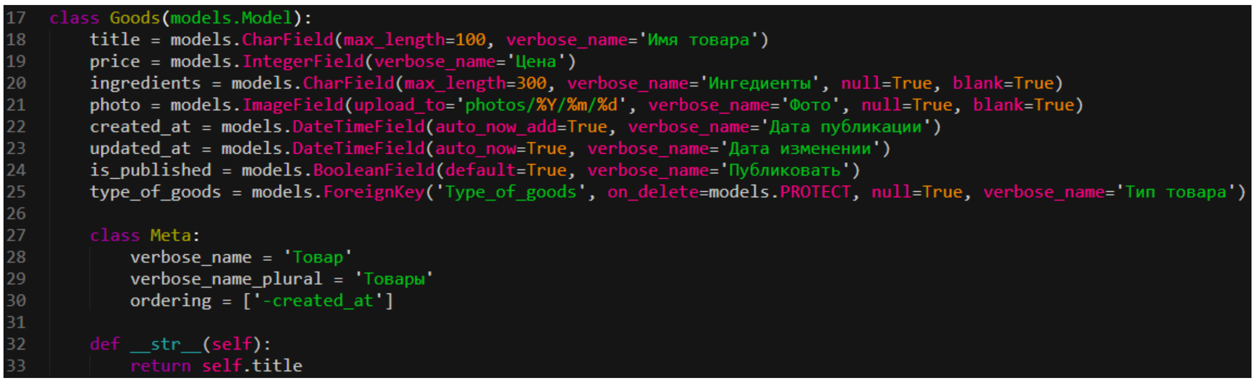
5.2 Python and Django

WARNING, Modules that I used in my code will be not shown in the section. Because I don’t think that it is useful information. Moreover, I will not cover all aspects of programming because same of them are similar to each other or too easy to notice or too many and complex to explain in several sentences.

5.2.0.1 Models of database

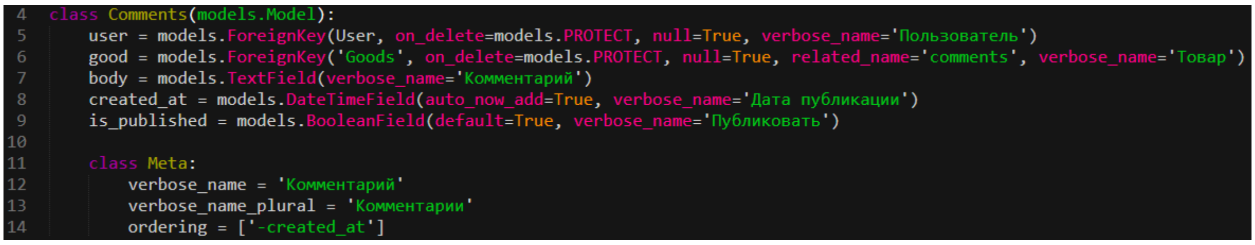
First of all, we should create models of tables that will be used to save, change, receive and delete data. It is possible by creating class with table name and using special classes of Django. Firstly, in our class, we declare fields of our class that are objects of other classes which are used to create fields of tables and their options. Secondly, to tune whole table, we create class Meta in our class. In the class Meta, we declare special fields that can be option of table as table name in admin page, ordering and others. In addition, there are special functions that can help to tune the table. One of them is function “__str__”. It is the function that can be in any python class and its job is output the data when an object is written in the functions as “print” or “str”. When this table is primary and it connects with secondary table, this is needed in admin page to observe value of foreign key that is an object of primary table.

Class “Goods” and Table “Goods”:



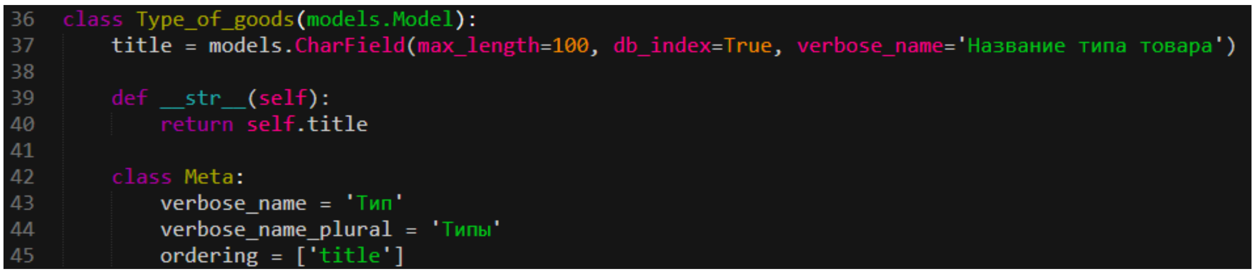
```python
class Goods(models.Model):
    title = models.CharField(max_length=100, verbose_name='Имя товара')
    price = models.IntegerField(verbose_name='Цена')
    ingredients = models.CharField(max_length=300, verbose_name='Ингедиенты', null=True, blank=True)
    photo = models.ImageField(upload_to='photos/%Y/%m/%d', verbose_name='Фото', null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='Дата публикации')
    updated_at = models.DateTimeField(auto_now=True, verbose_name='Дата изменении')
    is_published = models.BooleanField(default=True, verbose_name='Публиковать')
    type_of_goods = models.ForeignKey('Type_of_goods', on_delete=models.PROTECT, null=True, verbose_name='Тип товара')

    class Meta:
        verbose_name = 'Товар'
        verbose_name_plural = 'Товары'
        ordering = ['-created_at']

    def __str__(self):
        return self.title
```

Class “Comments” and Table “Comments”:



```python
class Comments(models.Model):
    user = models.ForeignKey(User, on_delete=models.PROTECT, null=True, verbose_name='Пользователь')
    good = models.ForeignKey('Goods', on_delete=models.PROTECT, null=True, related_name='comments', verbose_name='Товар')
    body = models.TextField(verbose_name='Комментарий')
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='Дата публикации')
    is_published = models.BooleanField(default=True, verbose_name='Публиковать')

    class Meta:
        verbose_name = 'Комментарий'
        verbose_name_plural = 'Комментарии'
        ordering = ['-created_at']
```

Class “Type_of_goods” and Table “Type_of_goods”



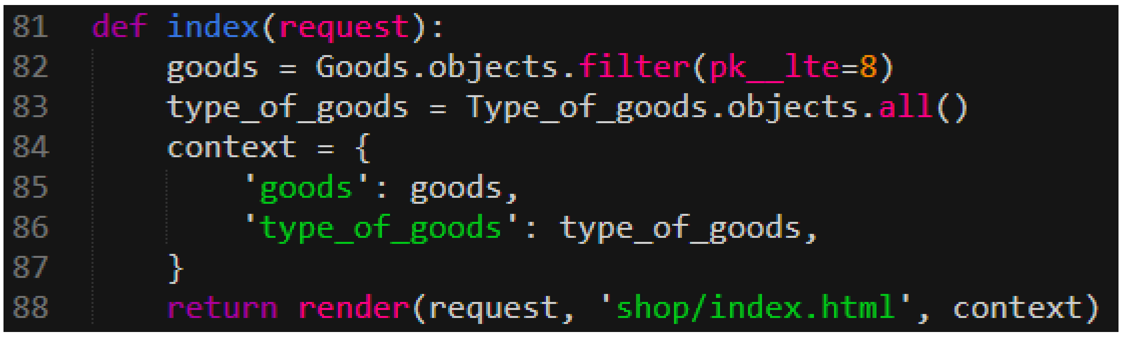
```python
class Type_of_goods(models.Model):
    title = models.CharField(max_length=100, db_index=True, verbose_name='Название типа товара')

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = 'Тип'
        verbose_name_plural = 'Типы'
        ordering = ['title']
```

In the website there is user table. But we don’t create it duo to Django that creates it instead of us.

It relates to next part of programming. All views of pages are written in view.py as functions. To connect a view with its link, there is urls.py.

5.2.1 Home page:

In view.py, index.html and the context are needed to render view. Context is a dictionary that contains keys which store data from database and are used in the template as a variables that output data in the page.



```python
def index(request):
    goods = Goods.objects.filter(pk__lte=8)
    type_of_goods = Type_of_goods.objects.all()
    context = {
        'goods': goods,
        'type_of_goods': type_of_goods,
    }
    return render(request, 'shop/index.html', context)
```

In urls.py, array urlpatterns has a function path that connect our function index with a link is called ‘’. Our page is accessible by link ‘’. It means that this page is the main and is created when guests open our site.

```
urlpatterns = [
    path('', index, name='home'),
    ...
]
```

5.2.2 Shop page:

Firstly, in forms.py, the form SortFilterForm is needed to create sorting and filtering forms in the page. It is created like a model. There are 2 fields are called sort and filt. They are both select inputs and have default values which save in arrays SORT_CHOICES and FILT_CHOICES.

```
SORT_CHOICES = [
    ('-updated_at', 'от нового до старого'),
     ('updated_at', 'от старого до нового'),
    ('price', 'сначало дешевые'),
    ('-price', 'сначало дорогие'),
]
FILT_CHOICES = [('0', 'все товары'),] + [(ToG.title, ToG.title) for ToG in Type_of_goods.objects.all()]

class SortFilterForm(forms.Form):
    sort = forms.ChoiceField(choices=SORT_CHOICES, widget=forms.Select(attrs={'onchange': 'submit();'}))
    filt = forms.ChoiceField(choices=FILT_CHOICES, widget=forms.Select(attrs={'onchange': 'submit();'}))
```

Secondly, in views.py, index.html and the context are needed to render view. In addition, we use our form to receive data from user, and then by data that the form receives the context of view alters too. It means that the values of key goods, products data are changed or change their order.

```
def shop(request):
    goods = Goods.objects.all()
    if request.method == 'GET':
        form = SortFilterForm(request.GET)
        if form.is_valid():
            if request.GET.get('filt') == '0':
                goods = Goods.objects.all().order_by(request.GET.get('sort'))
            else:
                goods = Goods.objects.filter(type_of_goods__title=request.GET.get('filt')).order_by(request.GET.get('sort'))
    else:
        form = SortFilterForm()
    type_of_goods = Type_of_goods.objects.all()
    context = {
        'goods': goods,
        'type_of_goods': type_of_goods,
        'form': form,
    }
    return render(request, 'shop/shop.html', context)
```

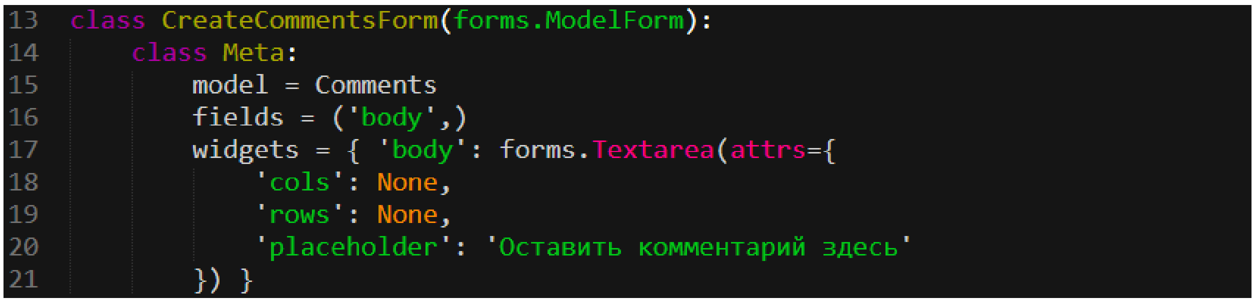
In the end, in urls.py, we put our view into function path with his link that will be used to open the page by tag ‘a’ and search form above user’s browser. The link is ‘shop/’ and the attribute name that is ‘shop’ is used in tags ‘a’ to write link. Because if we want change link of the page we don’t change whole reference of tags ‘a’ into right link. For example,

```
<a href="{% url 'shop' %}">
```

```
urlpatterns = [
    ...
    path('shop/', shop, name='shop'),
    ...
]
```

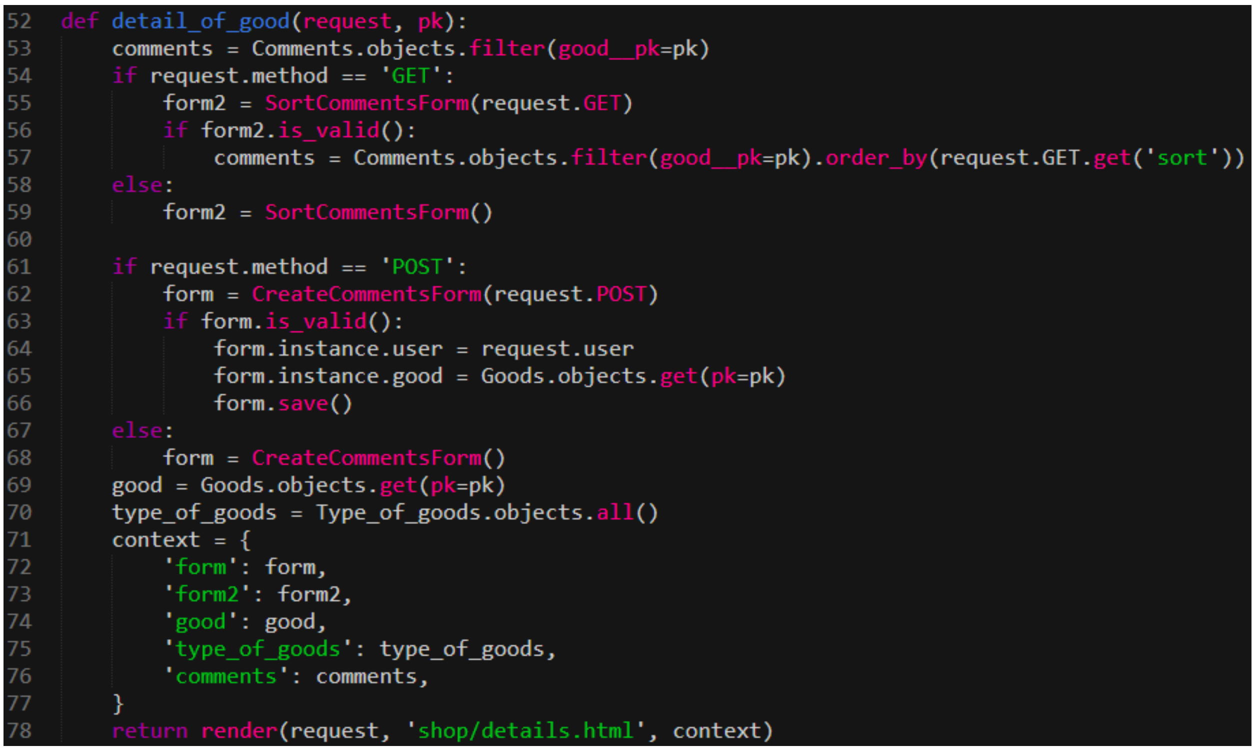
5.2.3 Product page:

In forms.py, we should create new form “CreateCommentsForm” to save data from guest into database in table “Comments”. This is the form that relates to model “Comments”:



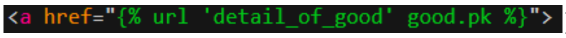
```
class CreateCommentsForm(forms.ModelForm):
    class Meta:
        model = Comments
        fields = ('body',)
        widgets = { 'body': forms.Textarea(attrs={
            'cols': None,
            'rows': None,
            'placeholder': 'Оставить комментарий здесь'
        }) }
```

In views.py, we add 2 forms. First one is to send comment by guest and saves the data into database in table ‘Comments’. Second one is known form ‘SortFilterForm’. But in this case, we use only sorting system. This page is about one product, so we have to know which product we need. That’s why; we give extra attribute ‘pk’ on function ‘detail_of_good’. Then we can get that record which its ‘id’ equals to ‘pk’. In addition we get all comment that are related to the product and put into the context. In the end, view renders whole data.



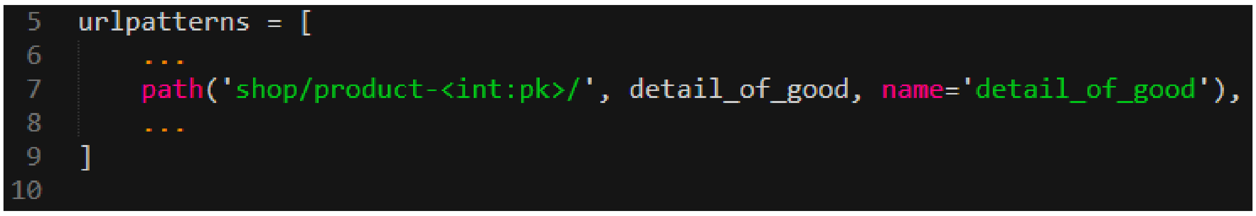
```
def detail_of_good(request, pk):
    comments = Comments.objects.filter(good__pk=pk)
    if request.method == 'GET':
        form2 = SortCommentsForm(request.GET)
        if form2.is_valid():
            comments = Comments.objects.filter(good__pk=pk).order_by(request.GET.get('sort'))
    else:
        form2 = SortCommentsForm()

    if request.method == 'POST':
        form = CreateCommentsForm(request.POST)
        if form.is_valid():
            form.instance.user = request.user
            form.instance.good = Goods.objects.get(pk=pk)
            form.save()
    else:
        form = CreateCommentsForm()
    good = Goods.objects.get(pk=pk)
    type_of_goods = Type_of_goods.objects.all()
    context = {
        'form': form,
        'form2': form2,
        'good': good,
        'type_of_goods': type_of_goods,
        'comments': comments,
    }
    return render(request, 'shop/details.html', context)
```

By the way, how attribute ‘pk’ contains useful value. It is possible by tag ‘a’ that we saw in previous part about shop page. We put data into link like this:



```
<a href="{% url 'detail_of_good' good.pk %}">
```

In that image, we see that link template has extra data good.pk. By this feature, we get attribute ‘pk’. And value of ‘pk’ can be used to create and recognize links.



```
urlpatterns = [
    ...
    path('shop/product-<int:pk>/', detail_of_good, name='detail_of_good'),
    ...
]
```

5.2.4 Register page:

In forms.py, we have to create a form “CreateUserForm” that is connected with User model to save data in its table. We put name of fields that exist in User model and are needed to get data from guests. If we look at field “fields”, there are only username and email. Because fields “password” and “password2” are created automatically.

```python
class CreateUserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'email']
```

In view.py, we use a form “CreateUserForm” to create new user. Obviously, in every form, we check to validation. After that the form saves the data into database in table ‘User’. Then it redirects new user to home page and output a message. In the end, view renders whole data.

```python
def registerPage(request):
    form = CreateUserForm()

    if request.method == 'POST':
        form = CreateUserForm(request.POST)
        if form.is_valid():
            form.save()
            user = form.cleaned_data.get('username')
            messages.success(request, user + ' успешно был зарегистрирован!')
            return redirect('home')

    context = {'form': form}
    return render(request, 'users/register.html', context)
```

The register page is accessible by link ‘register/’.

```python
urlpatterns = [
    ...
    path('register/', registerPage, name='register'),
]
```

5.2.5 Login page:

In view.py, we use special function of Django to find user account by value of inputs that are called ‘username’ and ‘password’ and are filled by guests. If that kind of user exists, website gives permit to activate user account and redirects to home page

```python
def loginPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.info(request, 'Имя пользователья или пароль не правильны!')
    return render(request, 'users/login.html')
```

The login page is accessible by link ‘login/’.

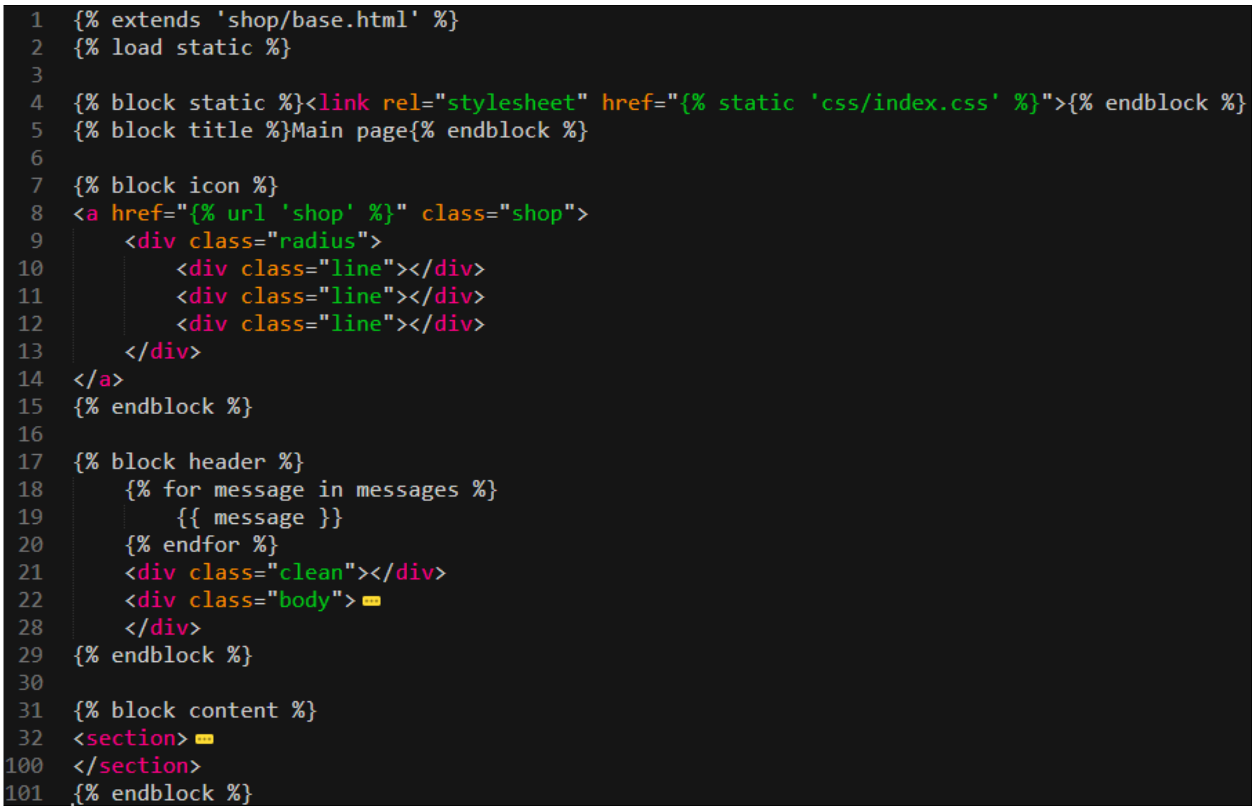
```
urlpatterns = [
    ...
    path('login/', loginPage, name='login'),
    ...
]
```

5.3 HTML and templates

There are 7 templates that are written in HTML. They are base.html, index.html, shop.html, special.html, details.html, account.html, login.html and register.html. And the interesting one is base.html. Because all other html files except login.html and register.html are connected with it. This is the technology of Django when a page is created by more than one html file. It is done by special Django tags like `{% extends 'shop/base.html' %}` `{% load static %}` `{% block title %}{% endblock %}` and others. In the first example we see symbols ‘{%%}’ that means it is Django tag, tag name ‘extends’ and file path ‘shop/base.html’. It is needed to import main file ‘base.html’ that would be body of template. In ‘base.html’, There are header, footer, and some html tags that is needed to create true template like “DOCTYPE”, “html”, “head”, “body” and others. Its short version is shown below:

```
{% load static %}
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    {% block static %}{% endblock %}
    <script src="{% static 'javascript/jquery-3.6.0.min.js' %}"></script>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    {% block icon %}{% endblock %}

    <header>
        <div class="head">⋯
        </div>

        {% block header %}{% endblock %}

    </header>

    {% block content %}{% endblock %}

    {% block footer %}
    <div class="clean"></div>
    <footer>⋯
    </footer>
    {% endblock %}
</body>
</html>
```

There are some Django tags like “block icon” or “block content”. But they are one type of tags “block” with the any name. It is very handsome tag that is used to combine html files and their fragments. For example this is short version of “index.html” that has tag “extends” that imports “base.html”, tag “load static” that imports some images, CSS and JS files and this tag “block” with values inside. Moreover, there aren’t any tags like “DOCTYPE”, “html”, “head” and “body”. Because main template is “base.html” and tags “block” with empty value in “base.html” are filled with tags “block” with value in “index.html”:



```
{% extends 'shop/base.html' %}
{% load static %}

{% block static %}<link rel="stylesheet" href="{% static 'css/index.css' %}">{% endblock %}
{% block title %}Main page{% endblock %}

{% block icon %}
<a href="{% url 'shop' %}" class="shop">
    <div class="radius">
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
    </div>
</a>
{% endblock %}

{% block header %}
    {% for message in messages %}
        {{ message }}
    {% endfor %}
    <div class="clean"></div>
    <div class="body">
    </div>
{% endblock %}

{% block content %}
<section>
</section>
{% endblock %}
```

If we look at this code, we can see some other tags like `{% for message in messages %}` and `{{ message }}` with symbols “{{ }}” that means to output value from variable. Firstly we should know that there are too many Django tags that their functionals equal to their python version. It means that first tag is similar to `for message in messages:` and second one equals to `print(message)`. Secondly, variables like messages is a key of a context that we render with html file in file “view.py”. So with that tags, we can create some kind of a system of rendering templates.

6.0.0 Testing

6.1.0 Test plan

6.1.1 Registration

№	Test objective	Data type	Data used	Expected outcome	Result
T1	Ensure that website checks the login for uniqueness During registration	Typical – register a new user with the login that doesn’t exist in a database.	Username – “example”	New user will be added to the User table	Successfull
T2		Erroneous – try to register user with login which already exists.	Username – “beksultan”	“Введите правильный адрес электронной почты.” message.	Successfull
T3	Ensure that password confirmation is efficient during registration	Typical – register a new user with the same password in both “Password” and “Password2” fields.	Password – “H786fty”, Password2 – “H786fty”	New user will be added to the User table	Successfull
T4		Erroneous – try to register a new user with different passwords in two fields.	Password – “H786fty”, Password2 – “H78s8asd9”	“Введенные пароли не совпадают.” message.	Successfull

6.1.2 Authorization

№	Test objective	Data type	Data used	Expected outcome	Result
T5	Ensure that the system checks the correctness of the login during authorization	Typical – authrize with an existing login	Username – “beksultan”	Redirecting to home page with activating user account	Successfull
T6		Erroneous – authorize with correct login and password	Username – “fakelogin”	“Имя пользователья или пароль не правильны!” message	Successfull
T7	Ensure that the system checks the correctness of the login during authorization	Typical – authorize with an correct login and password.	Username – “beksultan”, password – “1234567b”	Redirecting to home page with activating user account	Successfull
T8		Erroneous – authorize with a wrong password.	Username – “beksultan”, password – “123123123”	“Имя пользователья или пароль не правильны!” message	Successfull
T9	Ensure that user can log out	Typical – click “Выход” button		Redirecting to login page	Successfull

6.1.3 Sending comment

№	Test objective	Data type	Data used	Expected outcome	Result
T10	Ensure that user can sends comment	Typical – send comment with at list one sympol axcept space.	Body – “Very tasty pancake.”	Sending comment to the page	Successfull
T11		Erroneous – send comment with empty textarea or with only space symbols.	Body – “ ”	Nothing	Successfull

6.1.4 Filtering and sorting product blocks and sorting comments

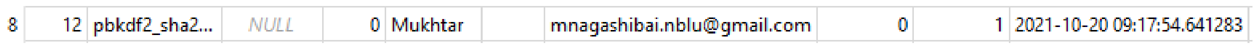
№	Test objective	Data type	Data used	Expected outcome	Result
T12	Ensure that user can filter product blocks	Typical – select one option in the select input	Select input – “Торт”	Filtering product block by chosen option	Successfull
T13	Ensure that user can sort product blocks	Typical – select one option in the select input	Select input – “от нового до старого”	Sorting product block by chosen option	Successfull
T14	Ensure that user can sort comments	Typical – select one option in the select input	Select input – “сначало новые”	Sorting comments by chosen option	Successfull

6.1.5 Saving and deleteing product data in user’s account

№	Test objective	Data type	Data used	Expected outcome	Result
T15	Ensure that user can bring product into his account page.	Typical – click “Добавить в корзину” button		Showing the product block when user open his account page	Successfull
T16	Ensure that user can delete all product blocks from the account	Typical – click “очистить корзину” button		Hiding all product blocks.	Successfull
T17	Ensure that user can delete targeted product block from the account	Typical – click “убрать” button		Hiding the product block.	Successfull

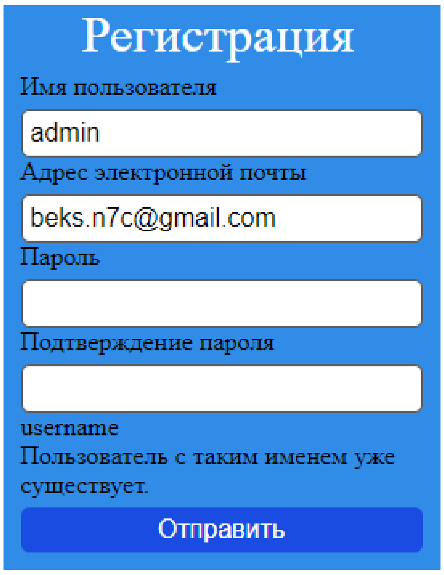
6.2.0 Results of testing

T1, T3: New user will be added to the User table.



```
8   12 pbkdf2_sha2...  NULL   0 Mukhtar      mnagashibai.nblu@gmail.com   0   1 2021-10-20 09:17:54.641283
```

T2: Registration wasn't successful, because username is already taken.



Регистрация

Имя пользователя

admin

Адрес электронной почты

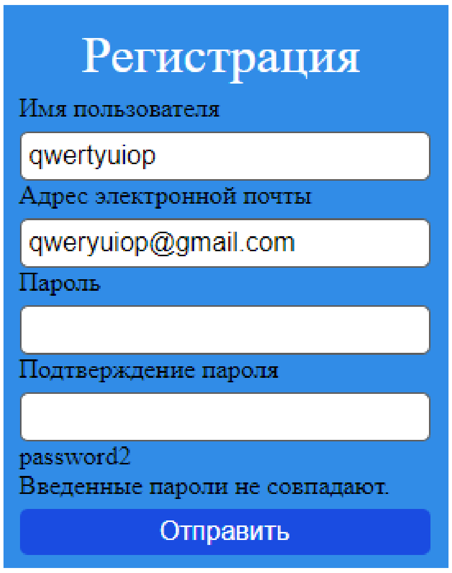
beks.n7c@gmail.com

Пароль

Подтверждение пароля

username
Пользователь с таким именем уже существует.

T4: Registration wasn't successful, because passwords are not the same.



Регистрация

Имя пользователя

qwertyuiop

Адрес электронной почты

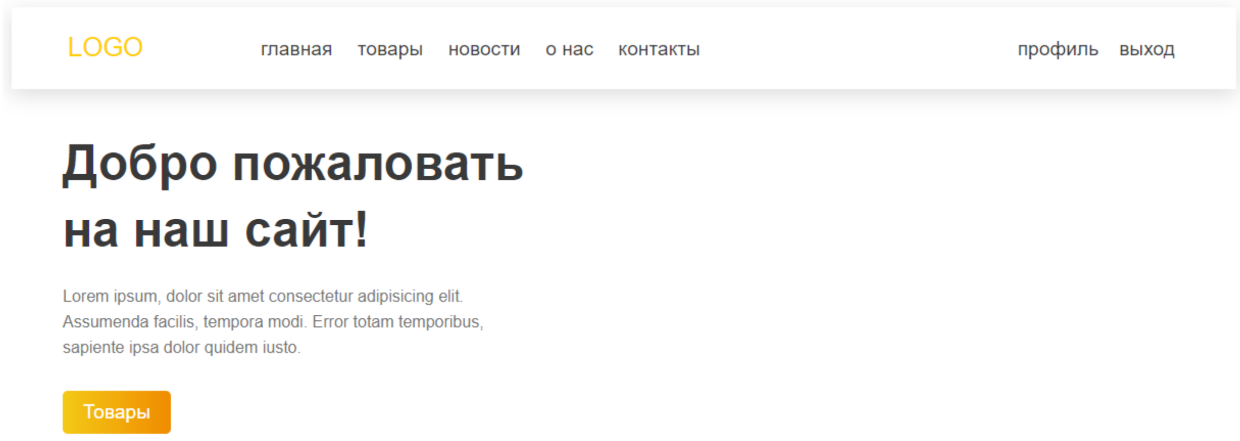
qweryuiop@gmail.com

Пароль

Подтверждение пароля

password2
Введенные пароли не совпадают.

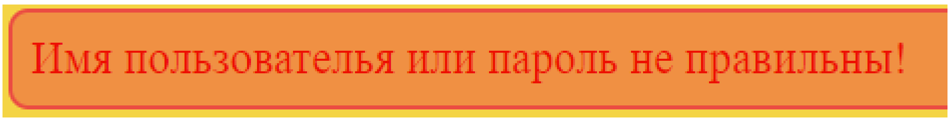
T5, T7: Redirecting to home page with activating user account.



Добро пожаловать на наш сайт!

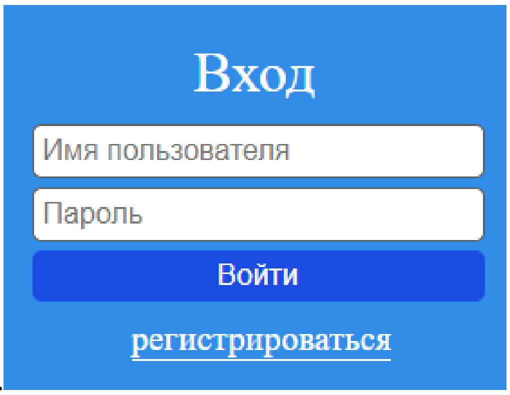
Lorem ipsum, dolor sit amet consectetur adipisicing elit. Assumenda facilis, tempora modi. Error totam temporibus, sapiente ipsa dolor quidem iusto.

T6: Authorization wasn't successful, because the username doesn't exist,
T8: Authorization wasn't successful, because the password is wrong.



Имя пользователья или пароль не правильны!

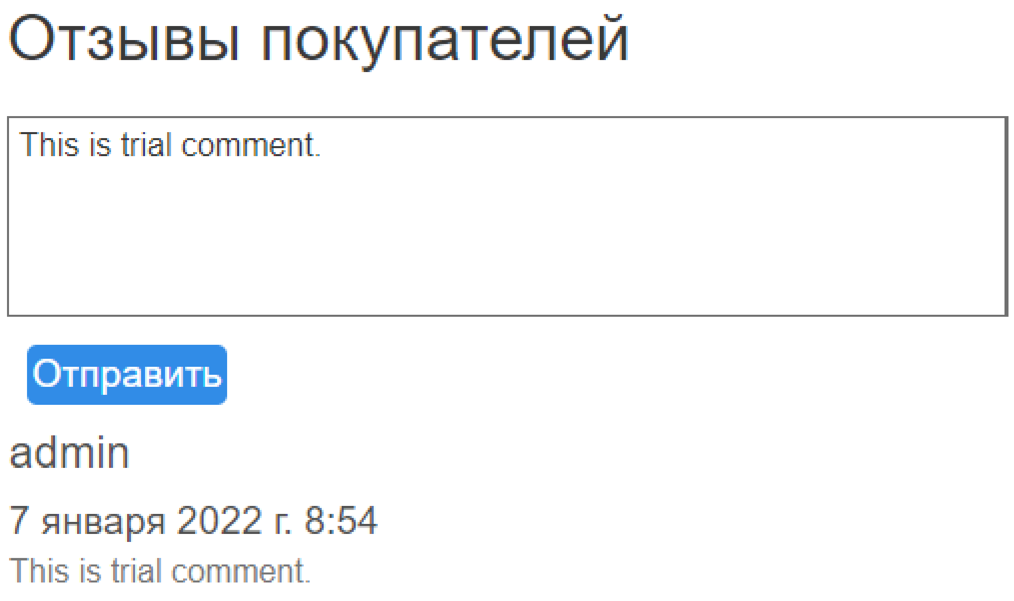
T9: Redirecting to login page



Вход

.

T10: Sending comment to the page.



Отзывы покупателей

This is trial comment.

admin

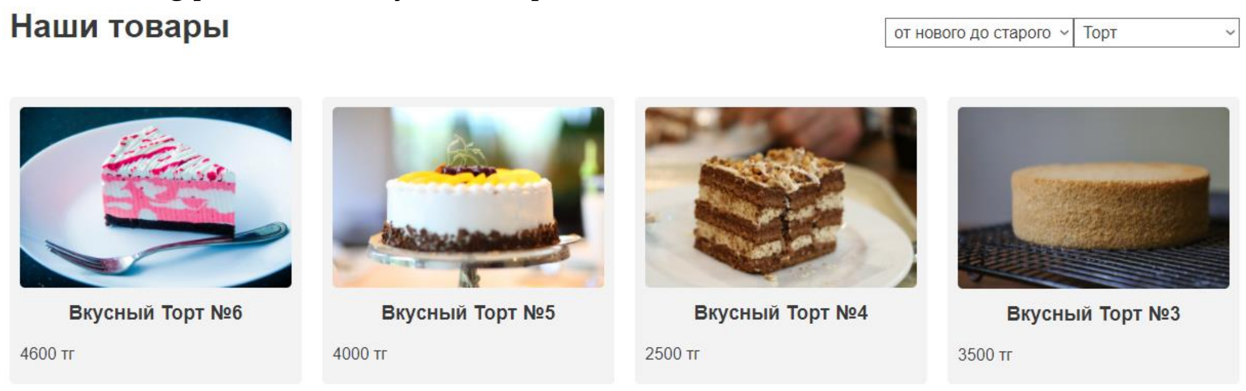
7 января 2022 г. 8:54

This is trial comment.

T11: Comment can’t be sent.

Nothing happens.

T12: Filtering product block by chosen option.



Наши товары

Вкусный Торт №6

4600 тг

Вкусный Торт №5

4000 тг

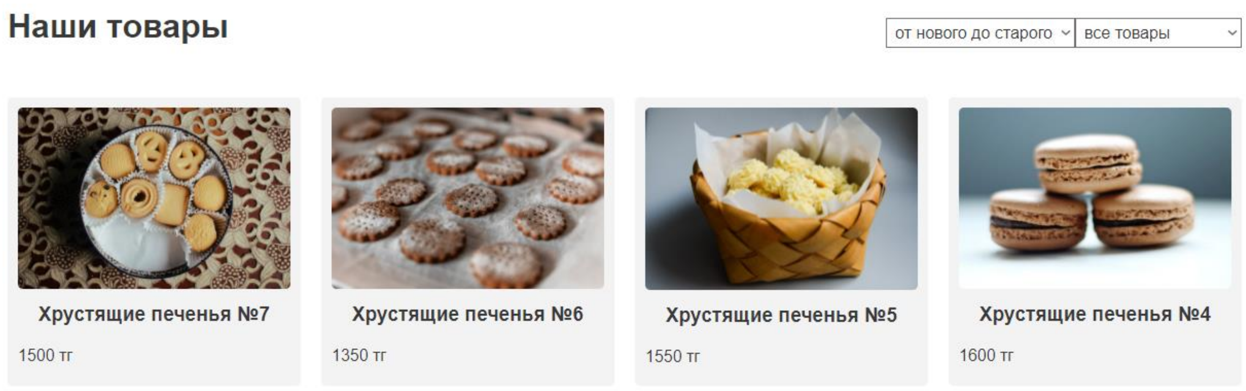
Вкусный Торт №4

2500 тг

Вкусный Торт №3

3500 тг

T13: Sorting product block by chosen option.



Наши товары

Хрустящие печенья №7

1500 тг

Хрустящие печенья №6

1350 тг

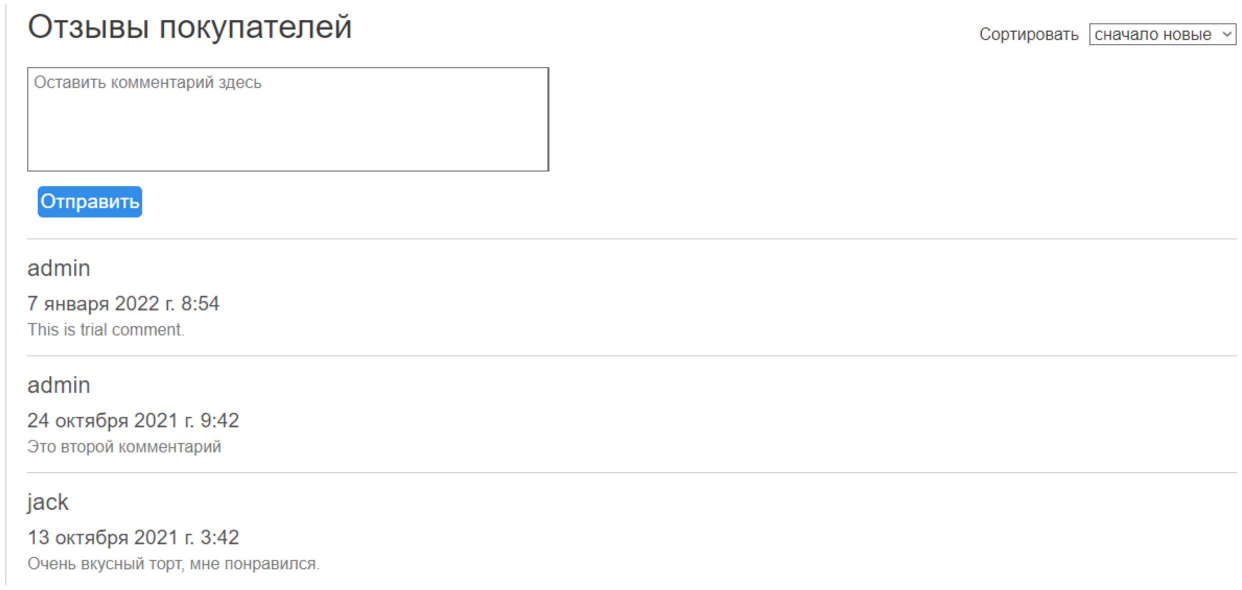
Хрустящие печенья №5

1550 тг

Хрустящие печенья №4

1600 тг

T14: Sorting comments by chosen option.


Отзывы покупателей

Сортировать сначало новые

Оставить комментарий здесь

Отправить

admin

7 января 2022 г. 8:54

This is trial comment.

admin

24 октября 2021 г. 9:42

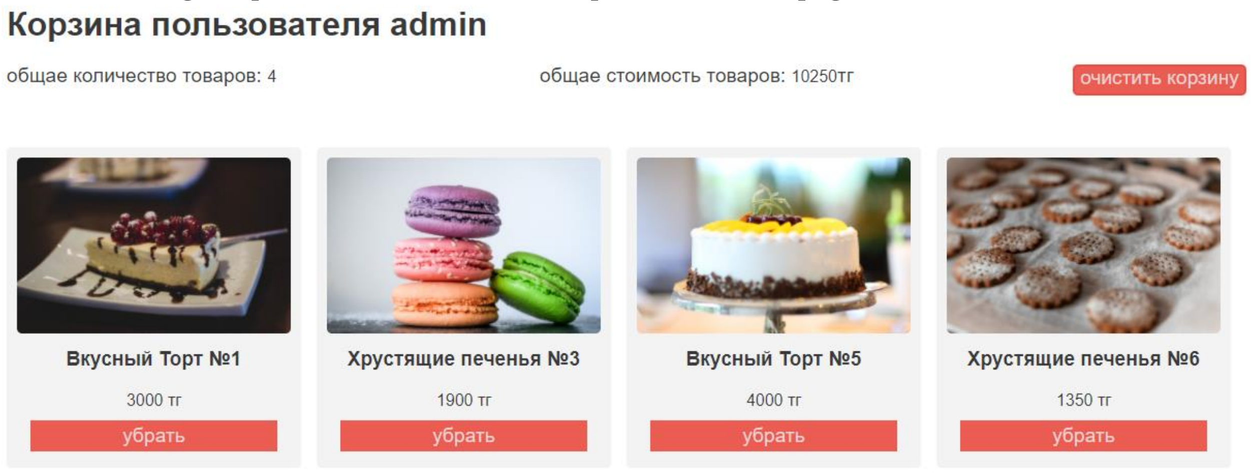
Это второй комментарий

jack

13 октября 2021 г. 3:42

Очень вкусный торт, мне понравился.

T15: Showing the product block when user open his account page.


Корзина пользователя admin

общае количество товаров: 4

общае стоимость товаров: 10250тг

очистить корзину

Вкусный Торт №1

3000 тг

убрать

Хрустящие печенья №3

1900 тг

убрать

Вкусный Торт №5

4000 тг

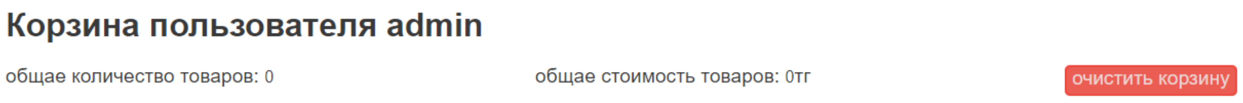
убрать

Хрустящие печенья №6

1350 тг

убрать

T16: Hiding all product blocks.

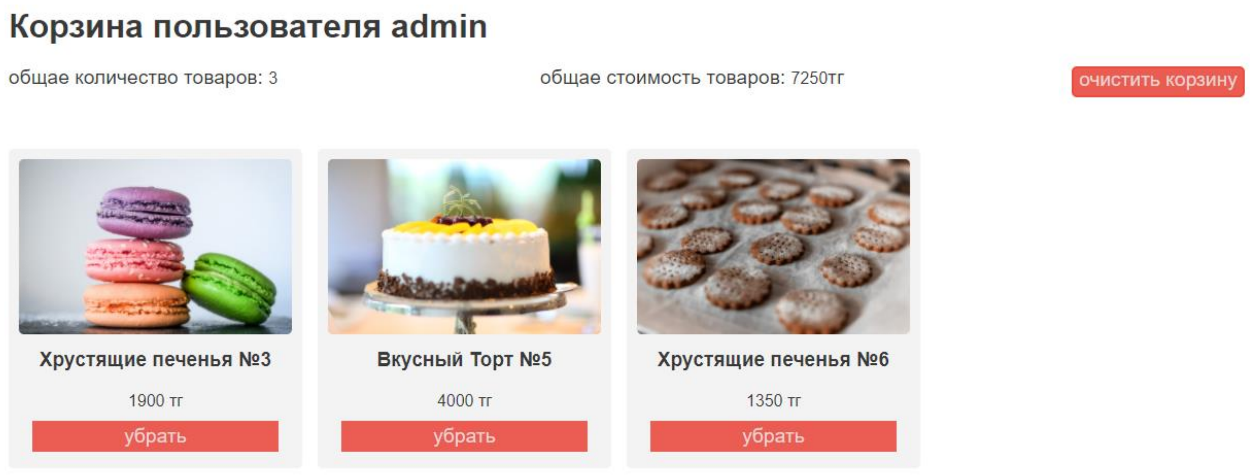

Корзина пользователя admin

общае количество товаров: 0

общае стоимость товаров: 0тг

очистить корзину

T17: Hiding the product block.



Корзина пользователя admin

общае количество товаров: 3

общае стоимость товаров: 7250тг

Хрустящие печенья №3

1900 тг

Вкусный Торт №5

4000 тг

Хрустящие печенья №6

1350 тг

8.0 System maintenance documentation

8.1 List of data files, use, size and location

Filename	Description	Size	Location
base.html	It contain base structure of website and some html fragments that stands in almost all pages. They are header and footer.	3KB	shop/templates/shop /base.html
index.html	Main page: it is used to show general information like popular products, photos of bakery and links with all main pages.	4KB	shop/templates/shop /index.html
shop.html	Shop page: it is used to represent whole variation of goods.	1KB	shop/templates/shop /shop.html
details.html	There are all data about only one product at once. In this page, users can transfer the product to account that is used as cart. In addition, users can send comments under the data.	4KB	shop/templates/shop /details.html
account.html	User's account page: it is used like cart where user carries products that are interesting to him.	3KB	shop/templates/users /account.html
register.html	Registration page: to give account to guests via receiving their personal data.	1KB	shop/templates/users /register.html
login.html	Authorization page: to give permission to users who have accounts and input right data.	1KB	shop/templates/users /login.html

8.2 Listing of databases tables and fields

Table name	Fields	Use
User	id, username, firstname, last name, email, password, is_active, is_superuser, is_staff, date_joined, last_login	It stores data about users.
Goods	id, title, price, ingredients, photo, created_at, updated_at, is_published, type_of_goods	It stores data about products that are shown in shop page.
Type_of_goods	id, title	It stores data about types of products that are used to split products into categories.
Comments	id, user, good, body, created_at, is_published.	It stores data about comments of products that users sent.

User

Fields	Data type	Use
id	integer	Stores id numbers of users
username	varchar	Stores the unique logins of users
password	varchar	Stores the password of users’ account
firlst_name	varchar	Stores firlst name of user
last_name	varchar	Stores last name of user
email	varchar	Stores email address of user
is_active	bool	To know is user active
is_staff	bool	To know is user general user
is_superuser	bool	To know is user admin
date_joined	DATETIME	Stores date of user registration
last_login	DATETIME	Stores date of user last login

Goods

Fields	Data type	Use
id	integer	Stores id numbers of goods
title	varchar	Stores the name of goods
price	integer	Stores the price of goods
ingredients	varchar	Stores the ingredients which goods contain
photo	varchar	Stores the photo of goods
is_published	bool	To give permit to publish goods
created_at	DATETIME	Stores date of publishing
updated_at	DATETIME	Stores date of last changing
Type_of_goods	bigint	Stores the type of goods

Comments

Fields	Data type	Use
id	integer	Stores id numbers of goods
user	varchar	Stores the username
good	integer	Stores the name of good
body	varchar	Stores the comments of user
is_published	bool	To give permit to publish comment
created_at	DATETIME	Stores date of publishing

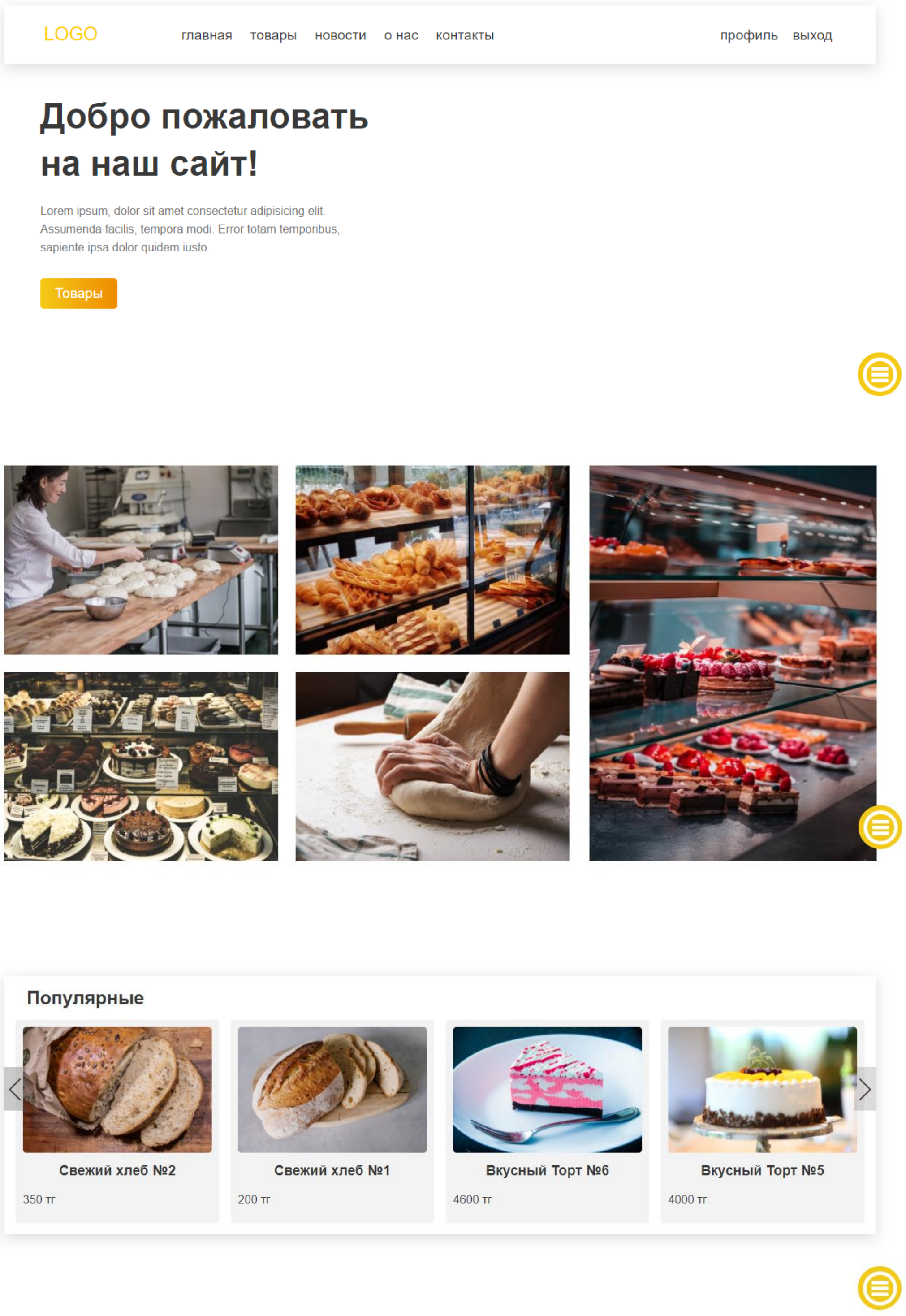
Type_of_goods

Fields	Data type	Use
id	integer	Stores id numbers of types
title	varchar	Stores the name of types

9.0 Documentation

9.1 User manual

1. When you open the website, you will see home page (index.html). On the main page, you can find some general information about the bakery and watch several popular products of the bakery. In addition, the page contains references to other pages such as register, login, shop pages.



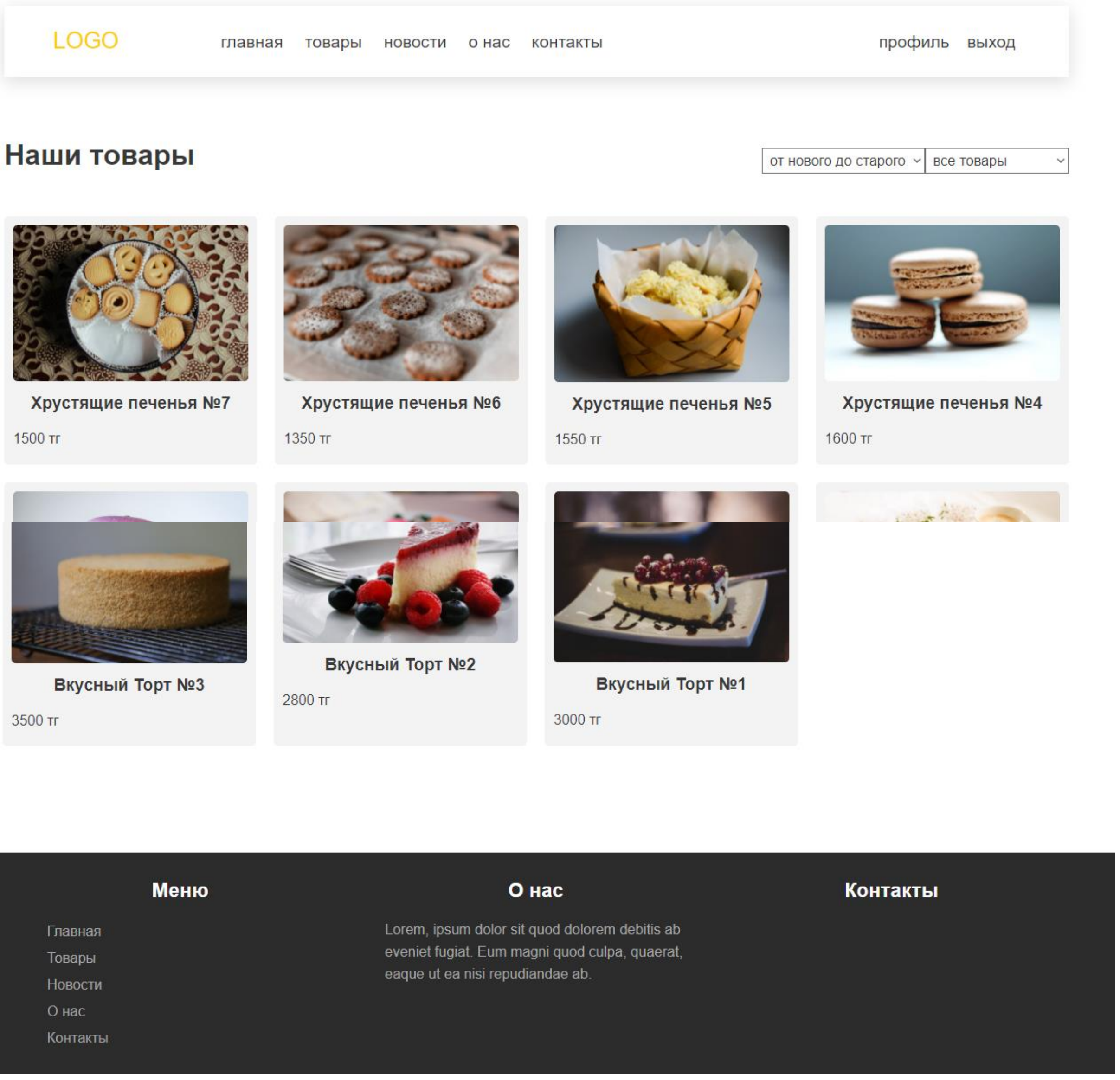
Добро пожаловать на наш сайт!

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Assumenda facilis, tempora modi. Error totam temporibus, sapiente ipsa dolor quidem iusto.

Популярные

Свежий хлеб №2	Свежий хлеб №1	Вкусный Торт №6	Вкусный Торт №5
350 тг	200 тг	4600 тг	4000 тг

2. If you want to watch all products, shop page (shop.html) is used to satisfy this request. You manage to sort and to filter stuffs in order to find needful one.



Наши товары

Хрустящие печенья №7
1500 тг

Хрустящие печенья №6
1350 тг

Хрустящие печенья №5
1550 тг

Хрустящие печенья №4
1600 тг

Вкусный Торт №3
3500 тг

Вкусный Торт №2
2800 тг

Вкусный Торт №1
3000 тг

Меню

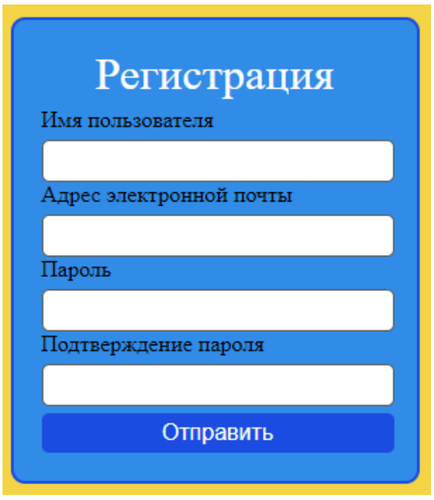
Главная
Товары
Новости
О нас
Контакты

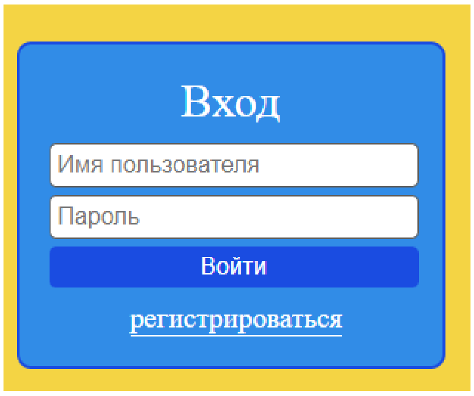
О нас

Lorem, ipsum dolor sit quod dolorem debitis ab eveniet fugiat. Eum magni quod culpa, quaerat, eaque ut ea nisi repudiandae ab.

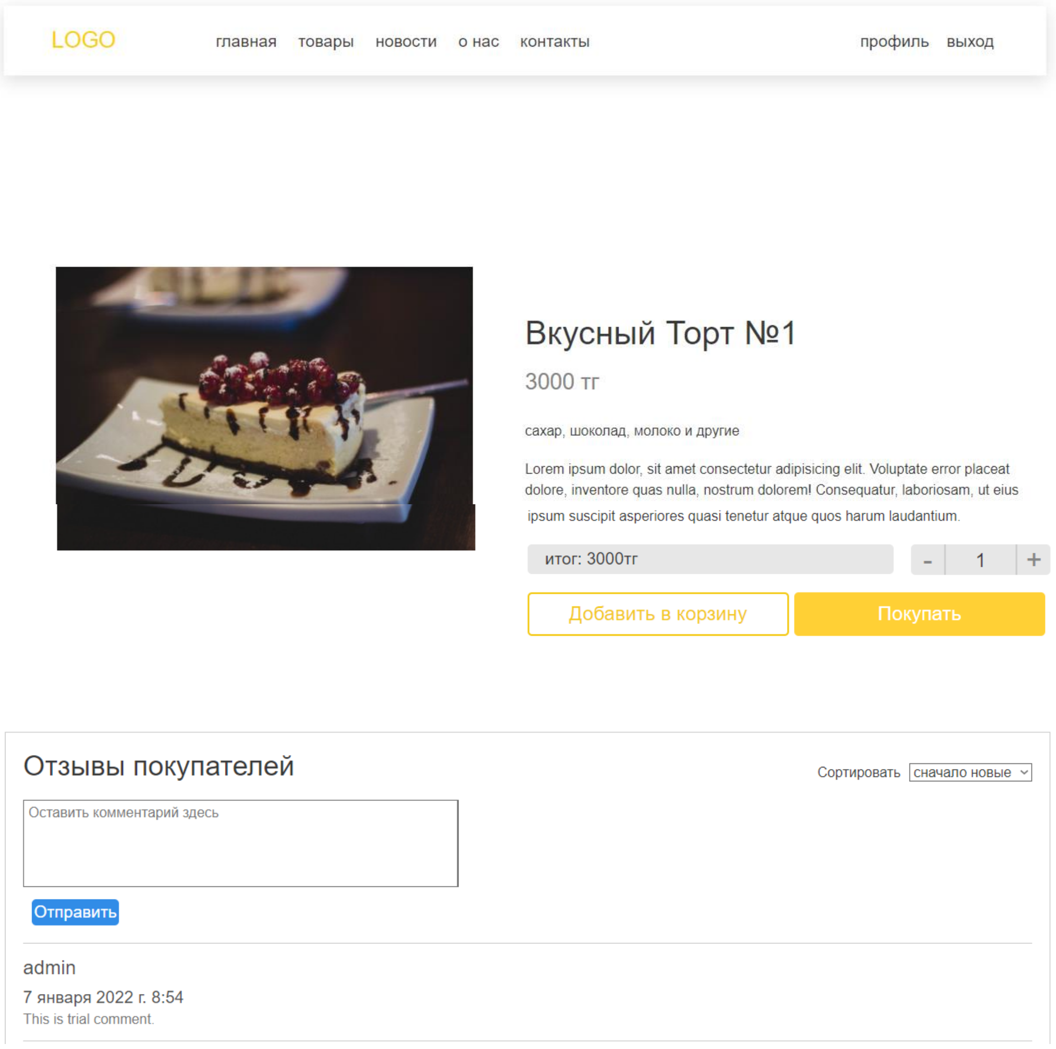
Контакты

3. If you don’t have account or want to authorize, you always can use register and login pages that their links are located at right side of header in any pages.





4. When you authorized, you are able to send comment in product pages where you observe all information about chosen product.



Вкусный Торт №1

3000 тг

сахар, шоколад, молоко и другие

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Voluptate error placeat dolore, inventore quas nulla, nostrum dolorem! Consequatur, laboriosam, ut eius ipsum suscipit asperiores quasi tenetur atque quos harum laudantium.

итог: 3000тг

Отзывы покупателей

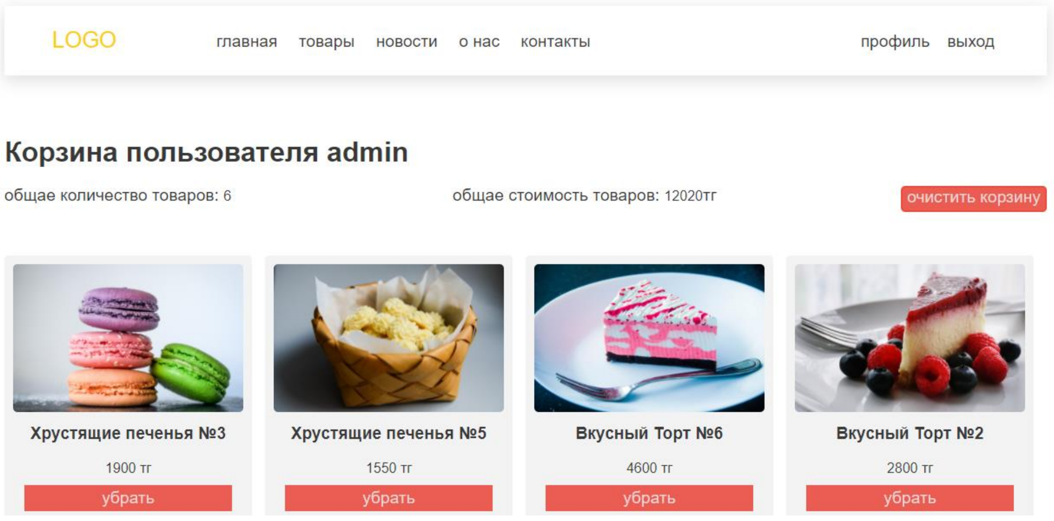
Сортировать: сначало новые

admin

7 января 2022 г. 8:54

This is trial comment.

5. In addition, in product pages, users who authorized can send chosen products to account page which is very useful to collect interesting goods.



Корзина пользователя admin

общае количество товаров: 6

общае стоимость товаров: 12020тг

Хрустящие печенья №3

1900 тг

Хрустящие печенья №5

1550 тг

Вкусный Торт №6

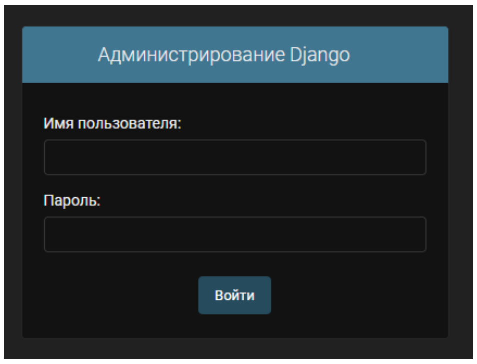
4600 тг

Вкусный Торт №2

2800 тг

9.2 Admin manual

1. You shoud search main page link with extra link '/admin' at the end of the string, and then you authorize in the page that opened.



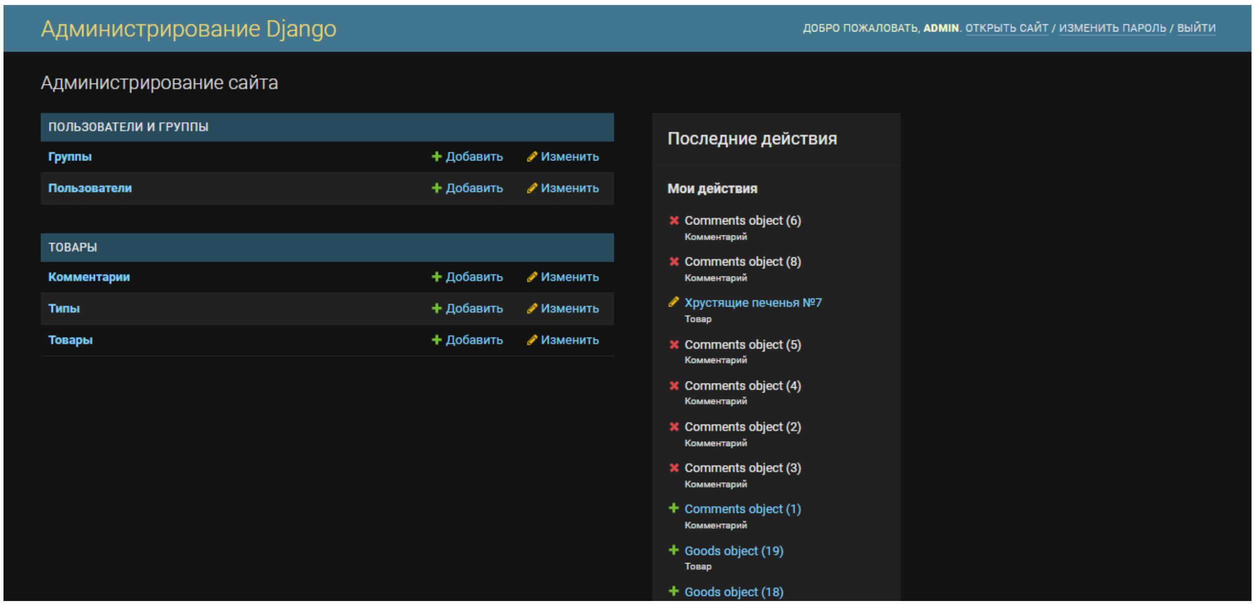
Администрирование Django

Имя пользователя:

Пароль:

Войти

2. After successful login, you see admin page of Django that contains all tables which I made and table which are made by the framework, and it has some other functions.



Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, **ADMIN**. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Администрирование сайта

ПОЛЬЗОВАТЕЛИ И ГРУППЫ		
Группы	+ Добавить	Изменить
Пользователи	+ Добавить	Изменить

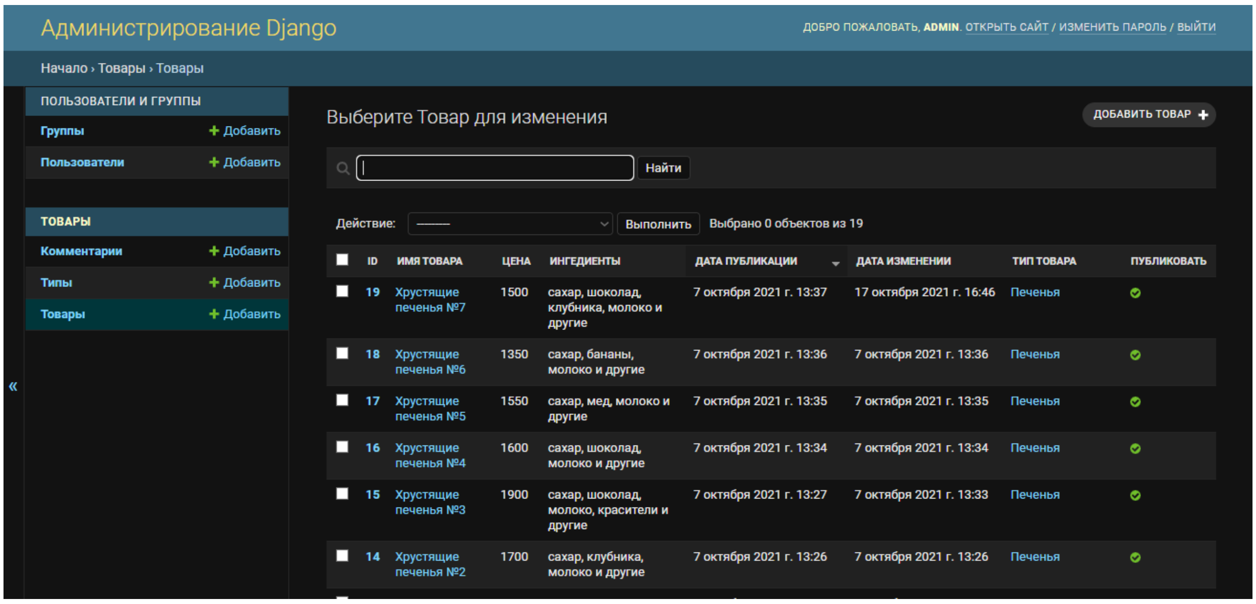
ТОВАРЫ		
Комментарии	+ Добавить	Изменить
Типы	+ Добавить	Изменить
Товары	+ Добавить	Изменить

Последние действия

Мои действия

- ✗ Comments object (6) — Комментарий
- ✎ Хрустящие печенья №7 — Товар
- ✗ Comments object (8) — Комментарий
- ✗ Comments object (5) — Комментарий
- ✗ Comments object (4) — Комментарий
- ✗ Comments object (2) — Комментарий
- ✗ Comments object (3) — Комментарий
- + Comments object (1) — Комментарий
- + Goods object (19) — Товар
- + Goods object (18)

3. By pressing names of tables, you can check all records.



Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, **ADMIN**. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Начало › Товары › Товары

ПОЛЬЗОВАТЕЛИ И ГРУППЫ	
Группы	+ Добавить
Пользователи	+ Добавить

ТОВАРЫ	
Комментарии	+ Добавить
Типы	+ Добавить
Товары	+ Добавить

Выберите Товар для изменения

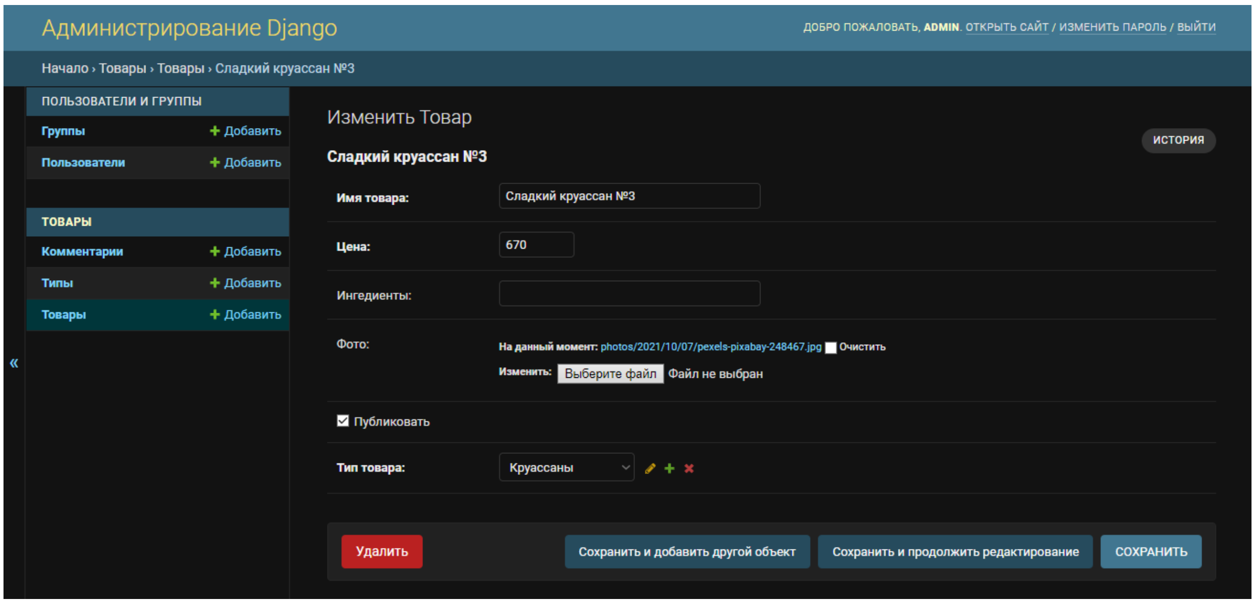
ДОБАВИТЬ ТОВАР +

Найти

Действие: ——— Выполнить Выбрано 0 объектов из 19

	ID	ИМЯ ТОВАРА	ЦЕНА	ИНГЕДИЕНТЫ	ДАТА ПУБЛИКАЦИИ	ДАТА ИЗМЕНЕНИИ	ТИП ТОВАРА	ПУБЛИКОВАТЬ
☐	19	Хрустящие печенья №7	1500	сахар, шоколад, клубника, молоко и другие	7 октября 2021 г. 13:37	17 октября 2021 г. 16:46	Печенья	✓
☐	18	Хрустящие печенья №6	1350	сахар, бананы, молоко и другие	7 октября 2021 г. 13:36	7 октября 2021 г. 13:36	Печенья	✓
☐	17	Хрустящие печенья №5	1550	сахар, мед, молоко и другие	7 октября 2021 г. 13:35	7 октября 2021 г. 13:35	Печенья	✓
☐	16	Хрустящие печенья №4	1600	сахар, шоколад, молоко и другие	7 октября 2021 г. 13:34	7 октября 2021 г. 13:34	Печенья	✓
☐	15	Хрустящие печенья №3	1900	сахар, шоколад, молоко, красители и другие	7 октября 2021 г. 13:27	7 октября 2021 г. 13:33	Печенья	✓
☐	14	Хрустящие печенья №2	1700	сахар, клубника, молоко и другие	7 октября 2021 г. 13:26	7 октября 2021 г. 13:26	Печенья	✓

4. If you want to change or delete one record, you just press its name, and the page will be opened.



Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, ADMIN. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Начало › Товары › Товары › Сладкий круассан №3

ПОЛЬЗОВАТЕЛИ И ГРУППЫ

Группы + Добавить

Пользователи + Добавить

ТОВАРЫ

Комментарии + Добавить

Типы + Добавить

Товары + Добавить

Изменить Товар

ИСТОРИЯ

Сладкий круассан №3

Имя товара: Сладкий круассан №3

Цена: 670

Ингредиенты:

Фото: На данный момент: photos/2021/10/07/pexels-pixabay-248467.jpg Очистить

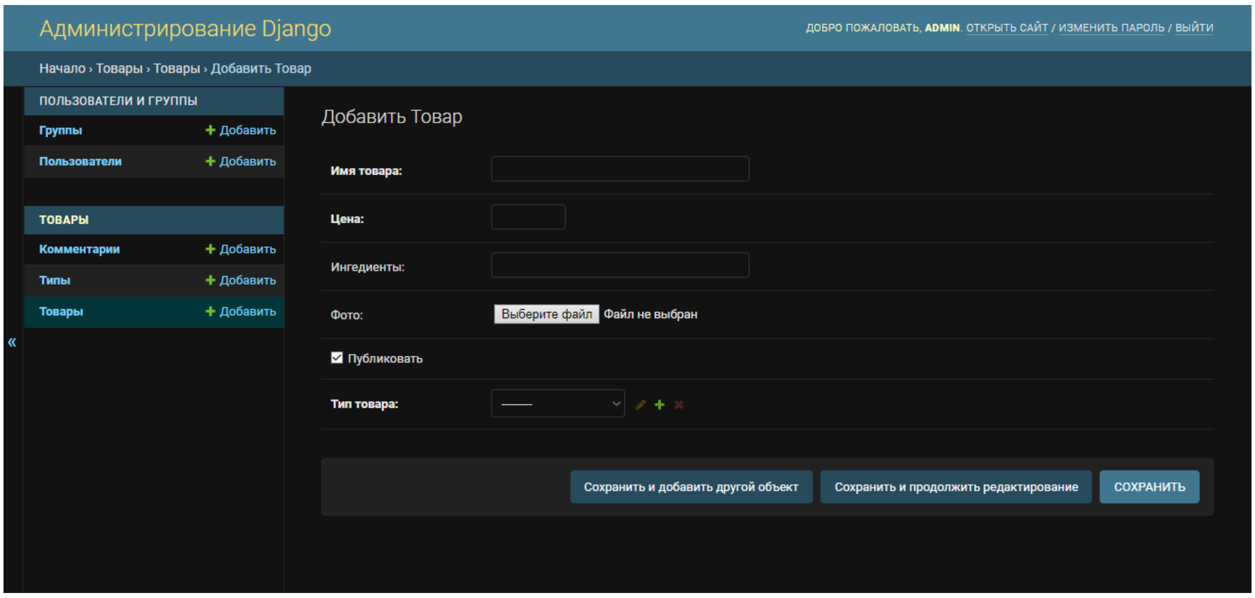
Изменить: Выберите файл Файл не выбран

☑ Публиковать

Тип товара: Круассаны

Удалить | Сохранить и добавить другой объект | Сохранить и продолжить редактирование | СОХРАНИТЬ

5. If you want to create new record, you shoud press the button “Добавить” at top right side of the table page.



Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, ADMIN. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Начало › Товары › Товары › Добавить Товар

ПОЛЬЗОВАТЕЛИ И ГРУППЫ

Группы + Добавить

Пользователи + Добавить

ТОВАРЫ

Комментарии + Добавить

Типы + Добавить

Товары + Добавить

Добавить Товар

Имя товара:

Цена:

Ингредиенты:

Фото: Выберите файл Файл не выбран

☑ Публиковать

Тип товара: ---------

Сохранить и добавить другой объект | Сохранить и продолжить редактирование | СОХРАНИТЬ

10.0 Evaluation

10.1 List of objectives

1. To create convenient and instinctively understandable interface.
2. The design has to be made of modern technology by using HTML, CSS and JavaScript.
3. All backend will be made of program language as python, especially django which is framework of python and is used to create mostly large websites.
4. In process of authorization and registration, the demanded input data is an email and a password.
5. The website should have the cart function which is used by registered users.
6. The website should contain different variation of product.
7. There must be forms to sort and to filter products.
8. In the administration page, only administer can open the page and create, change or delete whole data.

10.2 Evaluation of objectives

№	Evaluation	Evidence
1	The website can be convenient and instinctively understandable.	User Documentation
2	The design has to be made of modern technology by using HTML, CSS and JavaScript.	Development
3	In process of authorization and registration, the demanded input data is an email, username and password.	Testing
4	The website should have the cart function which is used by registered users.	User Documentation
5	There must be forms to sort and to filter products.	User Documentation
6	In the administration page, only administer can open the page and create, change or delete whole data.	User Documentation