

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

ФАКУЛЬТЕТ ИКТ

Отчет по лабораторной работе №2 по курсу «Основы Web-
программирования»

Тема: РЕАЛИЗАЦИЯ WEB-СЕРВИСОВ СРЕДСТВАМИ Django
REST framework, React, Semantic UI React

Выполнил:

Арлаков Денис

Студент группы К3340

Проверил:

Говоров А. И.

Санкт-Петербург

2020

Цель работы: овладеть практическими навыками и умениями реализации web-сервисов средствами Django REST framework, React, Semantic UI React.

Оборудование: компьютерный класс.

Программное обеспечение: Python 3.6, Django REST framework, React, Semantic UI React.

Задачи работы:

- Анализ предметной области
- Определение функциональных требований
- Проектирование серверной части
- Проектирование интерфейсов, а именно:
 - Создание интерфейса для отображения продуктов для покупки.
 - Реализация возможности добавлять/изменять продукты и их описание, чтобы это отображалось в интерфейсе всех продуктов.
 - Добавление возможности аутентификации.

Ход работы:

1. Была создана модель данных:

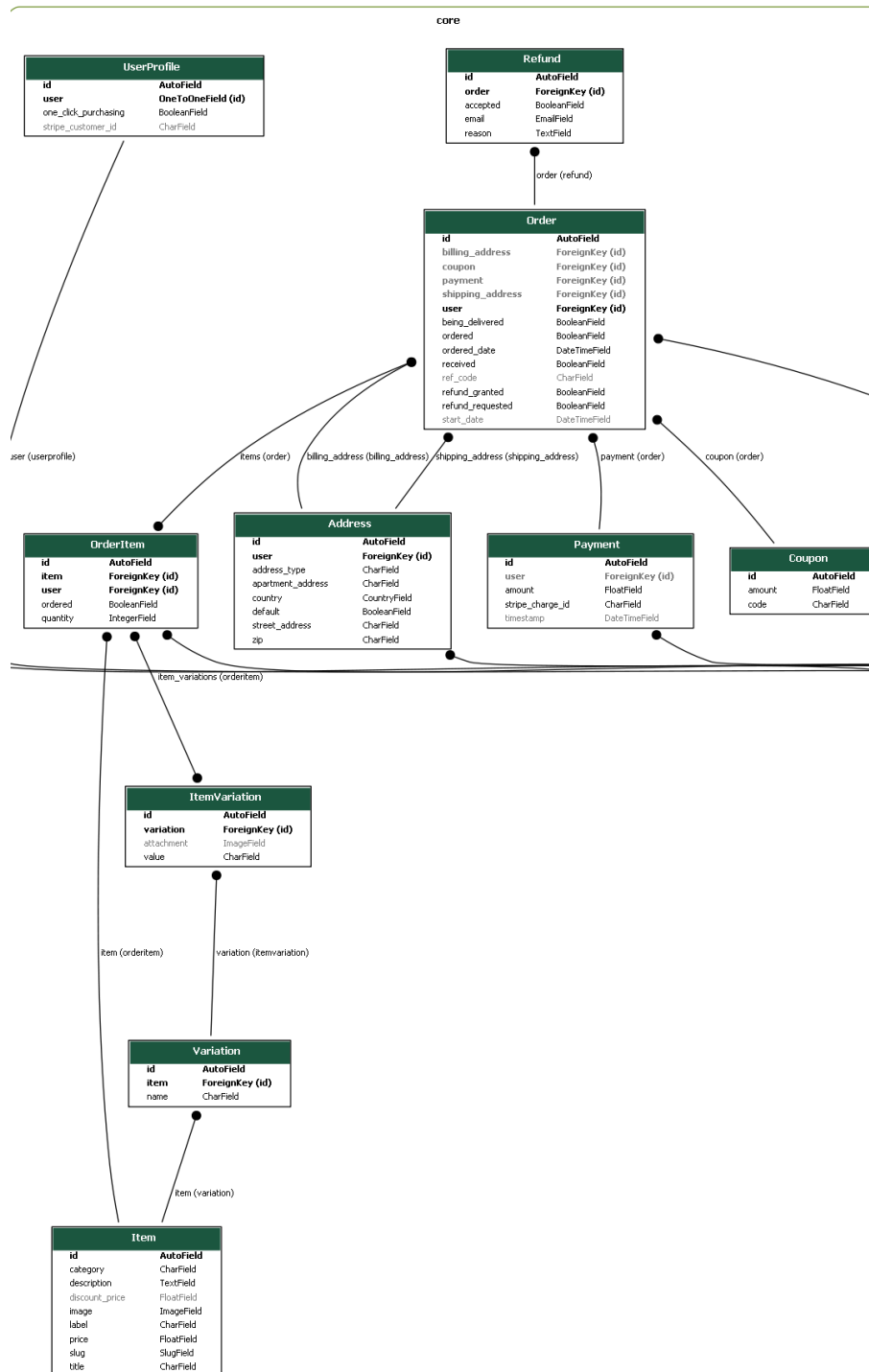


Рисунок 1 - Диаграмма базы данных интернет-магазина в полном размере.

2. Данная модель данных реализована в файле models.py

```
from django.db.models.signals import post_save
from django.conf import settings
```

```

from django.db import models
from django.db.models import Sum
from django.shortcuts import reverse
from django_countries.fields import CountryField

CATEGORY_CHOICES = (
    ('S', 'Shirt'),
    ('SW', 'Sport wear'),
    ('OW', 'Outwear')
)

LABEL_CHOICES = (
    ('P', 'primary'),
    ('S', 'secondary'),
    ('D', 'danger')
)

ADDRESS_CHOICES = (
    ('B', 'Billing'),
    ('S', 'Shipping'),
)

class UserProfile(models.Model):
    user = models.OneToOneField(
        settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    stripe_customer_id = models.CharField(max_length=50, blank=True, null=True)
    one_click_purchasing = models.BooleanField(default=False)

    def __str__(self):
        return self.user.username

class Item(models.Model):
    title = models.CharField(max_length=100)
    price = models.FloatField()
    discount_price = models.FloatField(blank=True, null=True)
    category = models.CharField(choices=CATEGORY_CHOICES, max_length=2)
    label = models.CharField(choices=LABEL_CHOICES, max_length=1)
    slug = models.SlugField()
    description = models.TextField()
    image = models.ImageField()

    def __str__(self):
        return self.title

```

```

def get_absolute_url(self):
    return reverse("core:product", kwargs={
        'slug': self.slug
    })

def get_add_to_cart_url(self):
    return reverse("core:add-to-cart", kwargs={
        'slug': self.slug
    })

def get_remove_from_cart_url(self):
    return reverse("core:remove-from-cart", kwargs={
        'slug': self.slug
    })

class Variation(models.Model):
    item = models.ForeignKey(Item, on_delete=models.CASCADE)
    name = models.CharField(max_length=50) # size

    class Meta:
        unique_together = (
            ('item', 'name')
        )

    def __str__(self):
        return self.name

class ItemVariation(models.Model):
    variation = models.ForeignKey(Variation, on_delete=models.CASCADE)
    value = models.CharField(max_length=50) # S, M, L
    attachment = models.ImageField(blank=True)

    class Meta:
        unique_together = (
            ('variation', 'value')
        )

    def __str__(self):
        return self.value

class OrderItem(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE)
    ordered = models.BooleanField(default=False)

```

```

item = models.ForeignKey(Item, on_delete=models.CASCADE)
item_variations = models.ManyToManyField(ItemVariation)
quantity = models.IntegerField(default=1)

def __str__(self):
    return f"{self.quantity} of {self.item.title}"

def get_total_item_price(self):
    return self.quantity * self.item.price

def get_total_discount_item_price(self):
    return self.quantity * self.item.discount_price

def get_amount_saved(self):
    return self.get_total_item_price() - self.get_total_discount_item_price()

def get_final_price(self):
    if self.item.discount_price:
        return self.get_total_discount_item_price()
    return self.get_total_item_price()

class Order(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE)
    ref_code = models.CharField(max_length=20, blank=True, null=True)
    items = models.ManyToManyField(OrderItem)
    start_date = models.DateTimeField(auto_now_add=True)
    ordered_date = models.DateTimeField()
    ordered = models.BooleanField(default=False)
    shipping_address = models.ForeignKey(
        'Address', related_name='shipping_address', on_delete=models.SET_NULL, blank=True, null=True)
    billing_address = models.ForeignKey(
        'Address', related_name='billing_address', on_delete=models.SET_NULL, blank=True, null=True)
    payment = models.ForeignKey(
        'Payment', on_delete=models.SET_NULL, blank=True, null=True)
    coupon = models.ForeignKey(
        'Coupon', on_delete=models.SET_NULL, blank=True, null=True)
    being_delivered = models.BooleanField(default=False)
    received = models.BooleanField(default=False)
    refund_requested = models.BooleanField(default=False)
    refund_granted = models.BooleanField(default=False)

    ...

1. Item added to cart

```

```

2. Adding a billing address
(Failed checkout)
3. Payment
(Preprocessing, processing, packaging etc.)
4. Being delivered
5. Received
6. Refunds
'''

def __str__(self):
    return self.user.username

def get_total(self):
    total = 0
    for order_item in self.items.all():
        total += order_item.get_final_price()
    if self.coupon:
        total -= self.coupon.amount
    return total

class Address(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE)
    street_address = models.CharField(max_length=100)
    apartment_address = models.CharField(max_length=100)
    country = CountryField(multiple=False)
    zip = models.CharField(max_length=100)
    address_type = models.CharField(max_length=1, choices=ADDRESS_CHOICES)
    default = models.BooleanField(default=False)

    def __str__(self):
        return self.user.username

    class Meta:
        verbose_name_plural = 'Addresses'

class Payment(models.Model):
    stripe_charge_id = models.CharField(max_length=50)
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.SET_NULL, blank=True, null=True)
    amount = models.FloatField()
    timestamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.user.username

```

```

class Coupon(models.Model):
    code = models.CharField(max_length=15)
    amount = models.FloatField()

    def __str__(self):
        return self.code

class Refund(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    reason = models.TextField()
    accepted = models.BooleanField(default=False)
    email = models.EmailField()

    def __str__(self):
        return f"{self.pk}"

def userprofile_receiver(sender, instance, created, *args, **kwargs):
    if created:
        userprofile = UserProfile.objects.create(user=instance)

post_save.connect(userprofile_receiver, sender=settings.AUTH_USER_MODEL)

```

3. Полученные интерфейсы в панели Django Admin

CORE		
Addresses	+ Add	Change
Coupons	+ Add	Change
Item variations	+ Add	Change
Items	+ Add	Change
Order items	+ Add	Change
Orders	+ Add	Change
Payments	+ Add	Change
Refunds	+ Add	Change
User profiles	+ Add	Change
Variations	+ Add	Change

Рисунок 2 - интерфейсы в панели Django Admin

– class Item(models.Model) – отвечает за каждый продукт. Здесь задаётся название, цена, цена со скидкой, категория, лэйбл, описание, slug(Slug - это тип поля в Django для создания понятных URL на латинице.), изображение.



Change item

Title:	<input type="text" value="Cap"/>
Price:	<input type="text" value="10,0"/>
Discount price:	<input type="text" value="5,0"/>
Category:	<input type="text" value="Sport wear"/>
Label:	<input type="text" value="primary"/>
Slug:	<input type="text" value="Cap_Star"/>
Description:	<div>The best cap in the world!</div>
Image:	Currently: blue_cap.jpg Change: <input type="button" value="Выберите файл"/> Файл не выбран

Рисунок 3 – вид сущности Item в панели администратора

– class Variation(models.Model) – отвечает за добавление вариативности к продукту, например выбор цвета или размера, а так же изображение.

Change variation

Item:  

Name:

ITEM VARIATIONS	
VALUE	ATTACHMENT
Red <input type="text" value="Red"/>	Currently: red_cap.jpg <input type="button" value="Clear"/> Change: <input type="button" value="Выберите файл"/> Файл не выбран
Blue <input type="text" value="Blue"/>	Currently: blue_cap_1x1N1k.jpg <input type="button" value="Clear"/> Change: <input type="button" value="Выберите файл"/> Файл не выбран
<input type="text"/>	<input type="button" value="Выберите файл"/> Файл не выбран




 Add another item variation

Рисунок 4 – вид сущности Variation в панели администратора

– class Order(models.Model) – является корзиной, в которой отображаются все добавленные продукты. Здесь мы можем добавлять промо-код, адрес доставки, почтовый адрес.


Change order

User: Way29  



Ref code:

Items:




1 of Shirt
2 of Shirt
1 of Shirt
1 of Shirt
1 of Pants
3 of Shirt
2 of Shirt
1 of Shirt
1 of Can









Hold down "Control", or "Command" on a Mac, to select more than one.

Ordered date: **Date:** 2020-06-27 Today 
Time: 13:31:11 Now 
Note: You are 3 hours ahead of server time.

☐ Ordered

Shipping address: -----   

Billing address: -----   

Payment: -----   






Coupon: -----   

Рисунок 5 – вид сущности Order в панели администратора

– class UserProfile(models.Model) – сущность профиля юзера для связи с библиотекой django-allauth. Имеет свой уникальный Stripe customer id.

Change user profile

User: waytonine  

Stripe customer id: ch_1FEJb9AOiJzDcnKnzD4W9K3C

☐ One click purchasing

Рисунок 6 – вид сущности UserProfile в панели администратора

- class Address(models.Model) – позволяет добавить адрес для получения товара, а так же почтовый адрес. Так же можно отметить возможность использовать адрес по умолчанию для ускорения процесса оплаты в будущем.

Change address

User:	waytonine	✎ +
Street address:	Привокзальная Площадь 3/3	
Apartment address:	542	
Country:	Russia	
Zip:	188661	
Address type:	Billing	
<input checked="" type="checkbox"/> Default		

Рисунок 7 – вид сущности Address в панели администратора

- class Payment(models.Model) – позволяет оплачивать выбранные товары.

Change payment

Stripe charge id:	ch_1FEJb9AOIjzDcnKnzD4W9K3C	
User:	waytonine	✎ + ✖
Amount:	360,0	

Delete

Рисунок 8 – вид сущности Payment в панели администратора

- class Coupon(models.Model) – позволяет добавлять скидочные промо-коды.

Change coupon

Code:	COUPON2020
Amount:	100,0

Рисунок 9 – вид сущности Coupon в панели администратора

4. Созданные интерфейсы с использованием React

Для реализации клиентской части мною была выбрана JavaScript-библиотека React, а так же для ускорения процесса разработки библиотека Semantic UI React.

Проект разделен на следующие главные интерфейсы: ProductList, ProductDetail, Signup, Login.

– Интерфейс ProductList.

Данный интерфейс отображает все товары доступные для приобретения. Из панели администратора их можно добавлять, удалять и изменять.

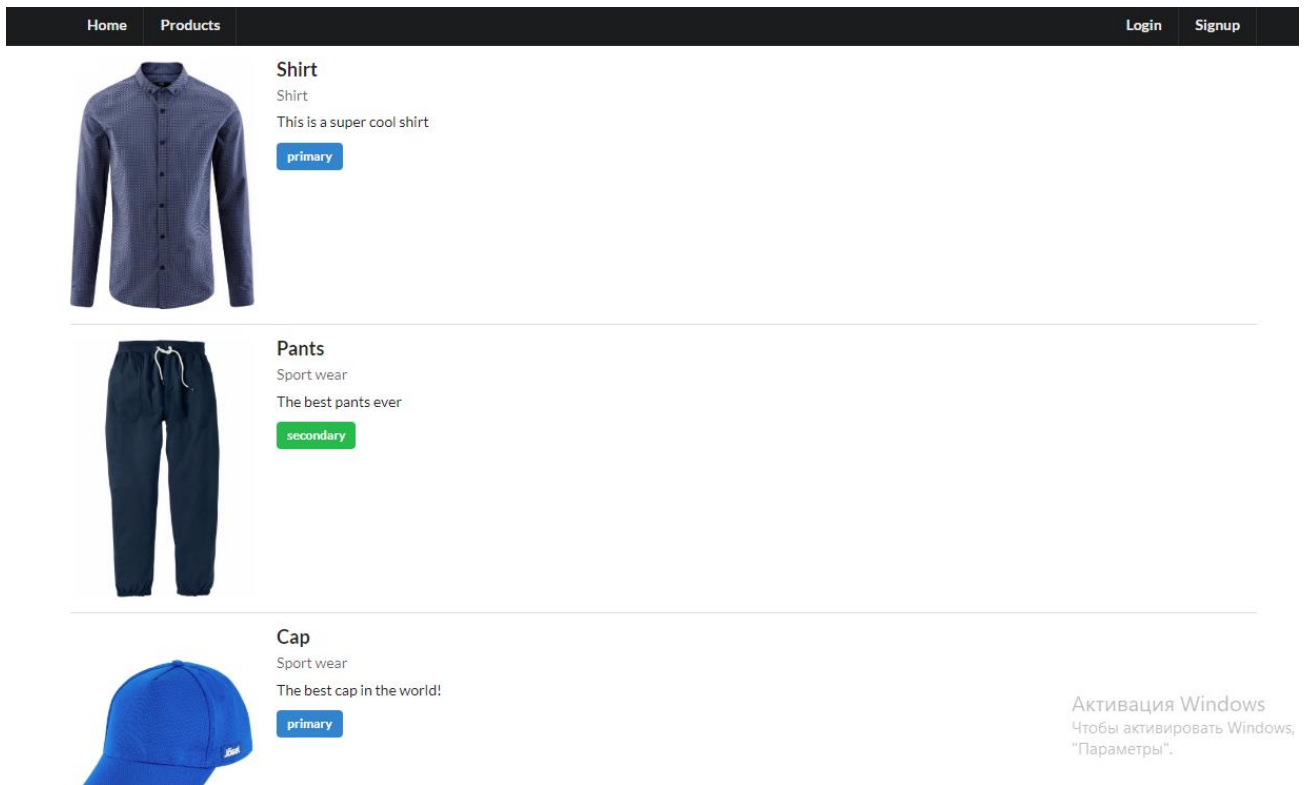


Рисунок 10 – интерфейс ProductList.

– Интерфейс ProductDetail

Данный интерфейс отображает каждый товар в подробностях. Здесь можно посмотреть фото крупным планом, выбрать необходимый цвет и размер товара. Так же в данном интерфейсе есть возможность добавить товар в корзину.

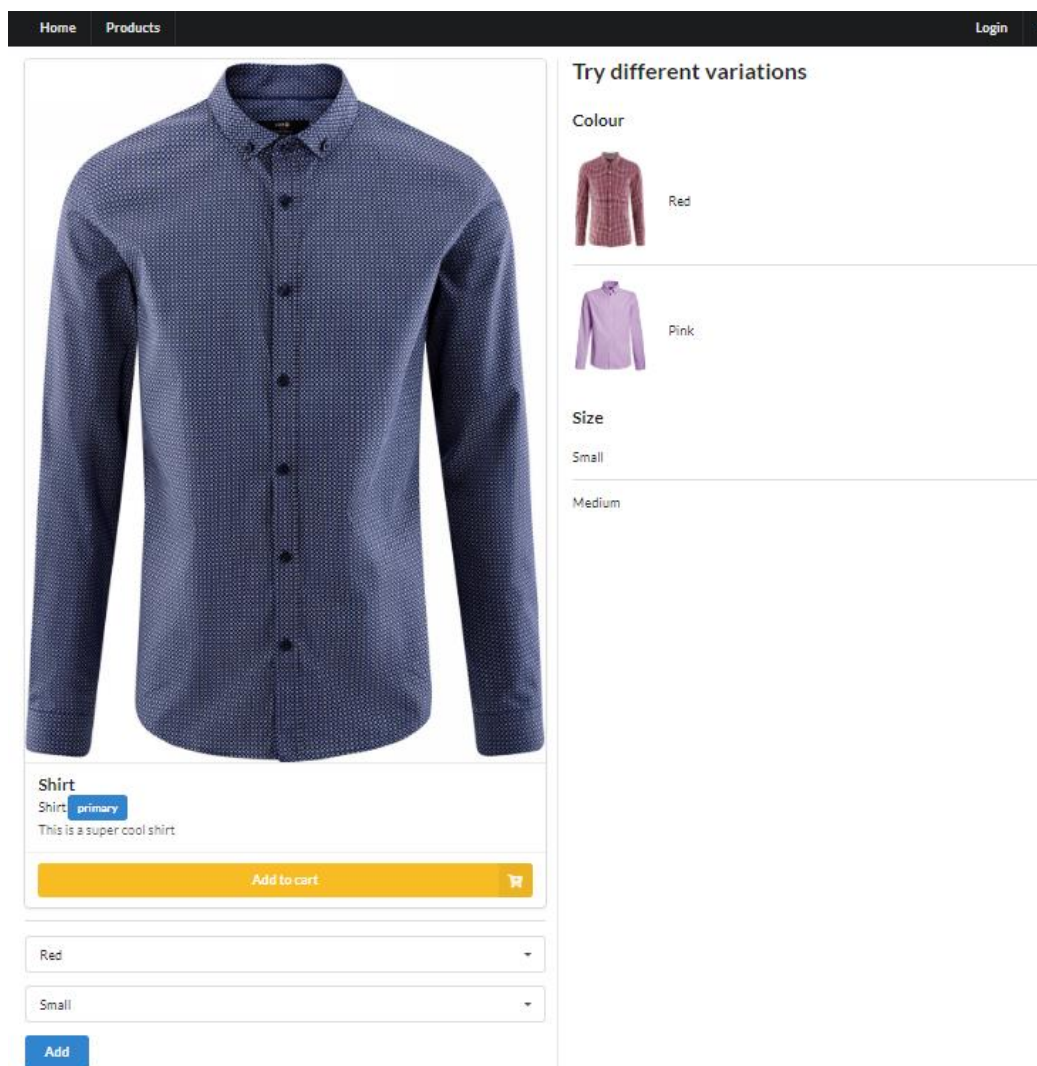
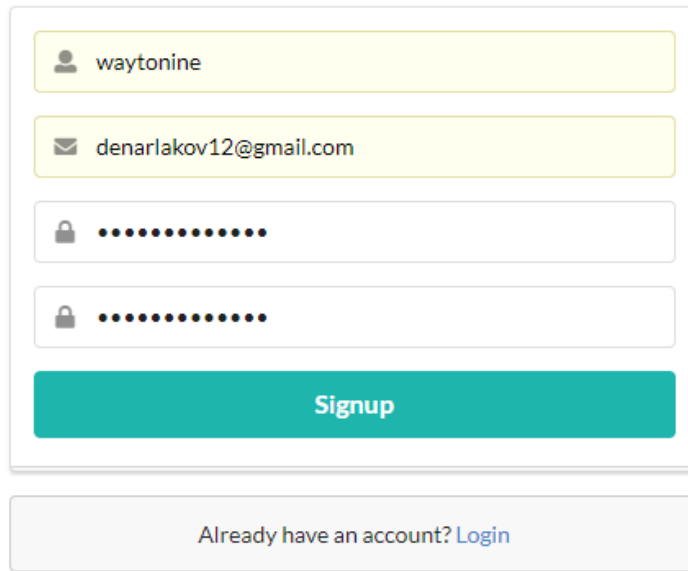


Рисунок 11 – интерфейс ProductDetail.

– Интерфейс Signup

Данный интерфейс отвечает за регистрацию новых пользователей.

Signup to your account



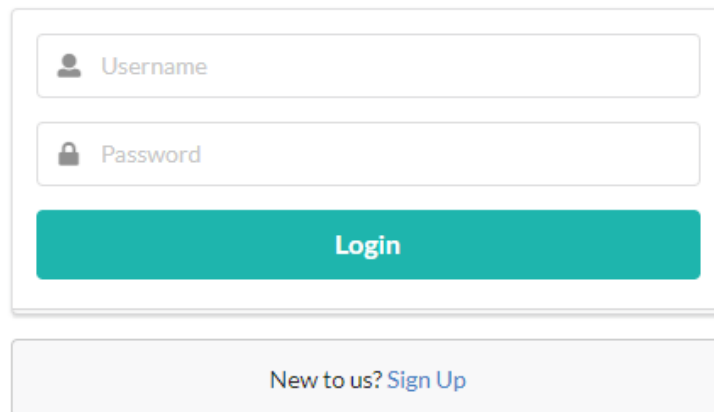
The image shows a 'Signup to your account' form. It has a teal title at the top. Below the title are four input fields: the first contains 'waytonine' with a user icon, the second contains 'denarlakov12@gmail.com' with an email icon, and the next two are password fields with lock icons and masked dots. A teal 'Signup' button is below the password fields. At the bottom, a light gray box contains the text 'Already have an account? [Login](#)'.

Рисунок 12 – интерфейс Signup.

– Интерфейс Login

Данный интерфейс отвечает за авторизацию уже зарегистрированных пользователей.

Log-in to your account



The image shows a 'Log-in to your account' form. It has a teal title at the top. Below the title are two input fields: the first is labeled 'Username' with a user icon, and the second is labeled 'Password' with a lock icon. A teal 'Login' button is below the password field. At the bottom, a light gray box contains the text 'New to us? [Sign Up](#)'.

Рисунок 13 – интерфейс Login.

5. Вывод:

В ходе выполнения лабораторной работы были получены навыки создания web-приложения с помощью web-фреймворка Django REST языка программирования Python, библиотеки React языка программирования JavaScript и Semantic UI React.