



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

ПО БАЗАМ ДАННЫХ

НА ТЕМУ:

Интернет-магазин техники

Студент ИУ7-64Б
(Группа)

Е. В. Власова
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта

Т. Н. Романова
(Подпись, дата) (И.О.Фамилия)

2020 г.

Содержание

Введение	3
1 Аналитический раздел	4
1.1 Формализация задачи	4
1.2 Типы баз данных.....	4
1.3 Вывод	8
2 Конструкторская часть	9
2.1 Проектирование Use-Case диаграммы БД	9
2.2 Проектирование таблиц БД	10
2.4 Ег диаграмма БД.....	13
2.5 Вывод.....	14
2 Технологическая часть	15
3.1 Выбор инструментов разработки.....	15
3.2 Реализация хранения данных	16
3.3 Frontend разработка	19
3.4 Интерфейс программы	20
3.5 Вывод	23
Заключение	24
Список использованной литературы.....	25

Введение

С развитием технологий многие сферы жизни переходят в «онлайн». Главным плюсом интернет-приложений – круглосуточная доступность. Особое место в Сети занимают интернет-магазины одежды, техники, аксессуаров, одежды и так далее. Они охватывают большую аудиторию, чем оффлайн магазины, и предоставляют большее количество ассортимента. С ростом популярности интернета появлялось всё больше сайтов, а это привело к росту конкуренции. Недостаточно просто создания интернет-магазина, надо ещё и сделать его удобным и интуитивно понятным для пользователя.

Цель данного курсового проекта – создание клиент-серверного приложения «Technovibe», которое является интернет-магазином техники. В приложении должна быть реализована авторизация, возможность поиска техники по различным критериям и цене. Пользователь может добавлять технику в список «Желаемое» и оформлять заказ.

Для достижения цели курсового проекта необходимо решить следующие задачи:

- Формализовать задачу, задав необходимый функционал;
- Выбрать технологический стек;
- Спроектировать базу данных, необходимую для решения поставленной цели;
- Реализовать спроектированную базу данных;
- Реализовать клиент-серверное приложение для взаимодействия с созданной базой данных.

1 Аналитический раздел

1.1 Формализация задачи

Пользователь должен иметь возможность регистрироваться на сайте, заходить на свой аккаунт и выходить с него. Должна быть реализация добавления техники в список «Желаемое», возможности просматривать уже добавленные продукты, а также оформления заказа. Необходима реализация поиска техники по фильтрам (по категории, бренду и так далее). У администратора должна быть возможность добавлять/удалять продукты интернет-магазина, а также менять их характеристики. Также администратор должен иметь возможность редактировать профили пользователей и их заказы.

1.2 Типы баз данных

База данных (БД) – совокупность данных, организованных по определённым правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ. Эти данные относятся к определённой предметной области и организованы таким образом, что могут быть использованы для решения многих задач многими пользователями. Выделяют три основных типа организации данных и связей между ними: иерархический (в виде дерева), сетевой и реляционный. [1]

1.2.1 Иерархическая модель БД

В данной модели имеется один главный объект и остальные – подчинённые объекты, находящиеся на разных уровнях иерархии. Взаимосвязи объектов образуют иерархическое дерево с одним корневым объектом.

Иерархическая БД состоит из упорядоченного набора нескольких экземпляров одного типа дерева. Автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

На рисунке 1 показана структуру иерархической базы данных, в самом верху находится родитель или корневой элемент, ниже находятся дочерние элементы, элементы, находящиеся на одном уровне, называются братьями. Чем ниже уровень элемента, тем вложенность этого элемента больше.

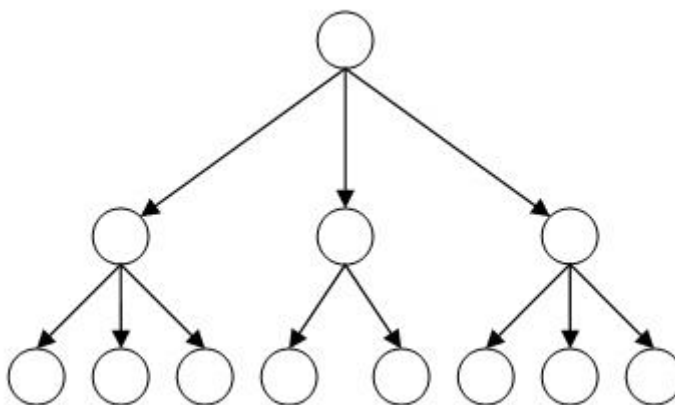


Рисунок 1 Иерархическая модель БД

1.2.2 Сетевая модель БД

Сетевой подход к организации данных является расширением иерархического. В иерархических структурах запись-потомок должна иметь в точности одного предка, в то время как в сетевой структуре данных потомок может иметь любое число предков.

В сетевой модели данных любой объект может быть одновременно и главным, и подчинённым и может участвовать в образовании любого числа взаимосвязей с другими объектами. Сетевая БД состоит из набора записей и набора связей между этими записями.

Связи между записями выполняются в виде указателей, т.е. каждая запись хранит ссылку на другую однотипную запись (или признак конца списка) и ссылки на списки подчинённых записей, связанных с ней групповыми отношениями. Таким образом, в каждой вершине записи хранятся в виде связного списка. Если список организован как однонаправленный, запись имеет ссылку на следующую

однотипную запись в списке; если список двунаправленный – то на следующую и предыдущую однотипные записи.

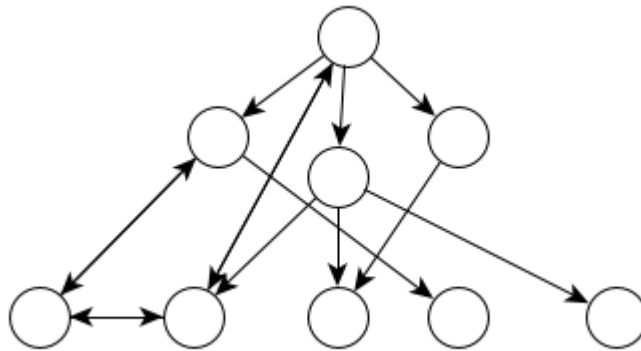


Рисунок 2 Сетевая модель БД

1.2.3 Реляционная модель БД

Главной особенностью реляционных баз данных является, то, что объекты внутри таких баз данных хранятся в виде набора двумерных таблиц. То есть, таблица состоит из набора столбцов, в котором может указываться: название, тип данных (дата, число, строка, текст и т.д.). Еще одной важной особенностью реляционных БД является, то, что число столбцов фиксировано, то есть, структура базы данных известна заранее, а вот число строк или рядов в реляционных базах данных ничем не ограничено.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя (заголовки столбцов являются названиями полей в записях);
- одинаковые строки в таблице отсутствуют;

- порядок следования строк и столбцов может быть произвольным.

Отношение – это плоская таблица, содержащая N столбцов, среди которых нет одинаковых. Столбец отношения соответствует атрибуту сущности. Кортеж – строка отношения (соответствует записи в таблице).[2]

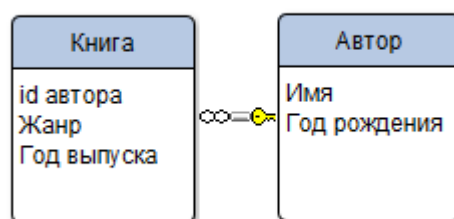


Рисунок 3 пример реляционной модели БД

1.2.4 Сравнение моделей БД

Основным недостатком иерархической модели БД является дублирование данных. Оно вызвано тем, что каждая сущность (атрибут) может относиться только к одной родительской сущности. Также поиск какого-либо элемента данных в такой системе может оказаться довольно трудоемким из-за необходимости последовательно проходить несколько предшествующих иерархических уровней.

Сетевая модель БД является достаточно сложной для проектирования и поддержки. В этой модели не обеспечивается физическая независимость данных, так как наборы организованы с помощью физических ссылок. Также не обеспечивается независимость данных от программ.

Реляционная модель БД отличается наглядностью организации данных, а также скоростью поиска нужной информации, универсальностью и удобством обработки данных, которая осуществляется с помощью декларативного языка запросов SQL.

Исходя из описанного выше сравнения, в данной работе будет использоваться реляционная модель БД.

1.3 Вывод

В данном разделе была дана формализация задачи на курсовой проект. Рассмотрены модели БД, для данной работы выбрала реляционная модель БД.

2 Конструкторская часть

2.1 Проектирование Use-Case диаграммы БД

На рисунке 4 представлена Use-Case диаграмма проекта. С системой могут взаимодействовать три актора: администратор, пользователь и гость. У администратора есть права на изменение, добавление и удаление товаров, заказов и пользователей. Пользователь и гость могут оставить отзыв о товаре, просматривать список товаров и фильтровать их по параметрам, но пользователь также может добавлять товары в список «Желаемое» и оформлять заказ.

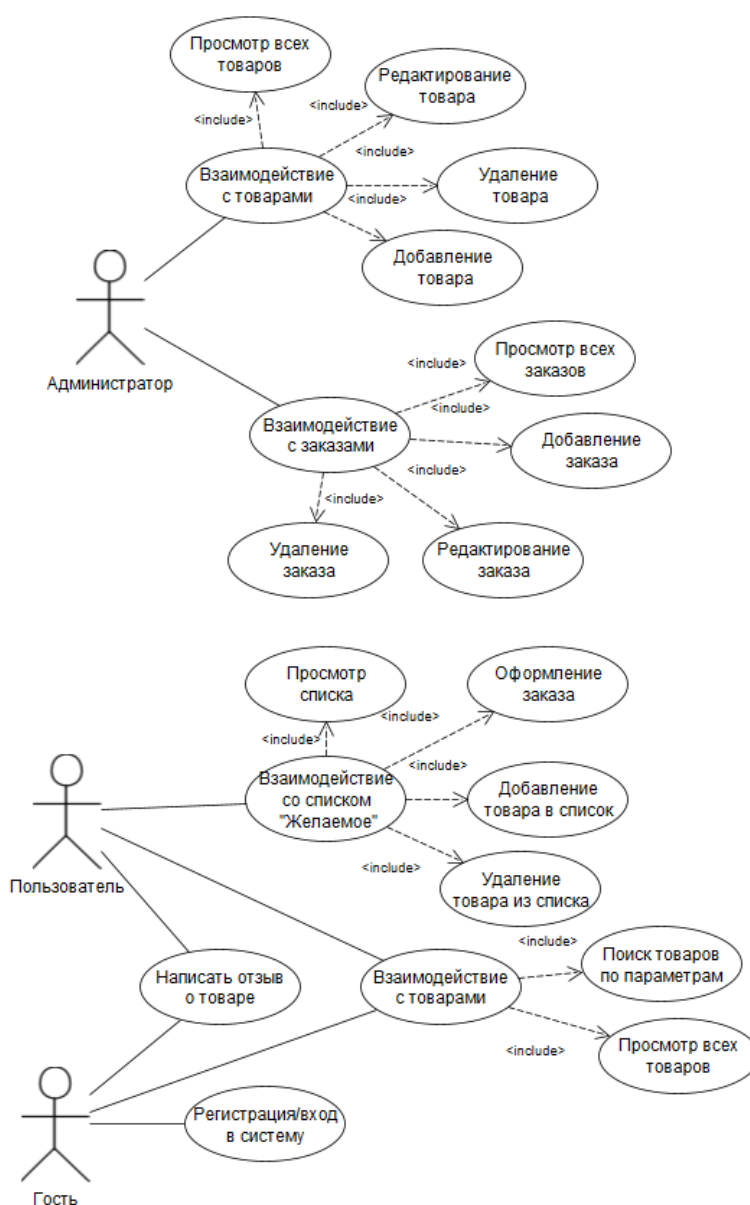


Рисунок 4 Use-case диаграмма

2.2 Проектирование таблиц БД

База данных интернет-магазина состоит из следующих таблиц:

1. Таблица пользователя;
2. Таблица товаров;
3. Таблица заказов;
4. Таблица отзывов.

2.2.1 Представление модели покупателя

Аккаунт покупателя должен содержать информацию, необходимую для его идентификации. Рассмотрим необходимые поля модели:

- First_name – имя пользователя;
- Last_name – фамилия пользователя;
- Email – почта пользователя;
- Password – пароль пользователя.

2.2.2 Представление модели продуктов

В данной таблице хранятся данные о продаваемой технике. Эта модель связана с моделью заказа, моделью фотографии продукта, а также с моделью отзыва.

- Name – название товара;
- Brand – бренд;
- Series – серия продукта;
- Category – категория;
- Price – цена;
- Discount – скидка;
- Color – цвет;

- Weight – вес;
- Country – страна производства;
- Guarantee – гарантия;
- Description – описание продукта.

Так как в работе представлены разные виды техники (ноутбуки и телефоны), то каждая категория должна иметь уникальные поля, помимо уже описанных.

Поля модели телефона:

- Screen – размер экрана;
- Dimensions – габариты;
- ROM – постоянное запоминающее устройство (ПЗУ);
- Bluetooth – версия блютуза.

Поля модели ноутбука:

- CPU_type – тип процессора;
- Cores – количество ядер;
- Memory_frequency – частота памяти;
- RAM – оперативная память (ОЗУ);
- Resolution – разрешение;
- Display_diagonal – диагональ дисплея.

2.2.3 Представление модели заказов

Данная модель содержит данные о статусе заказа.

- total_price – цена заказа;
- status – статус заказа;

- comments – комментарий к заказу.

Для того, чтобы описать товар в заказе, требуется дополнительная модель, в которой будет указана цена товара и его количество.

- Number – количество единиц данного товара;
- Price_pet_item – цена за одну единицу товара;
- Total_price – цена за все единицы товара.

2.2.4 Представление модели отзывов

Данная модель содержит отзыв пользователя о каком-либо товаре, а также его данные.

- Name – имя пользователя;
- Email – почта пользователя;
- Text - отзыв.

2.4 Диаграммы БД

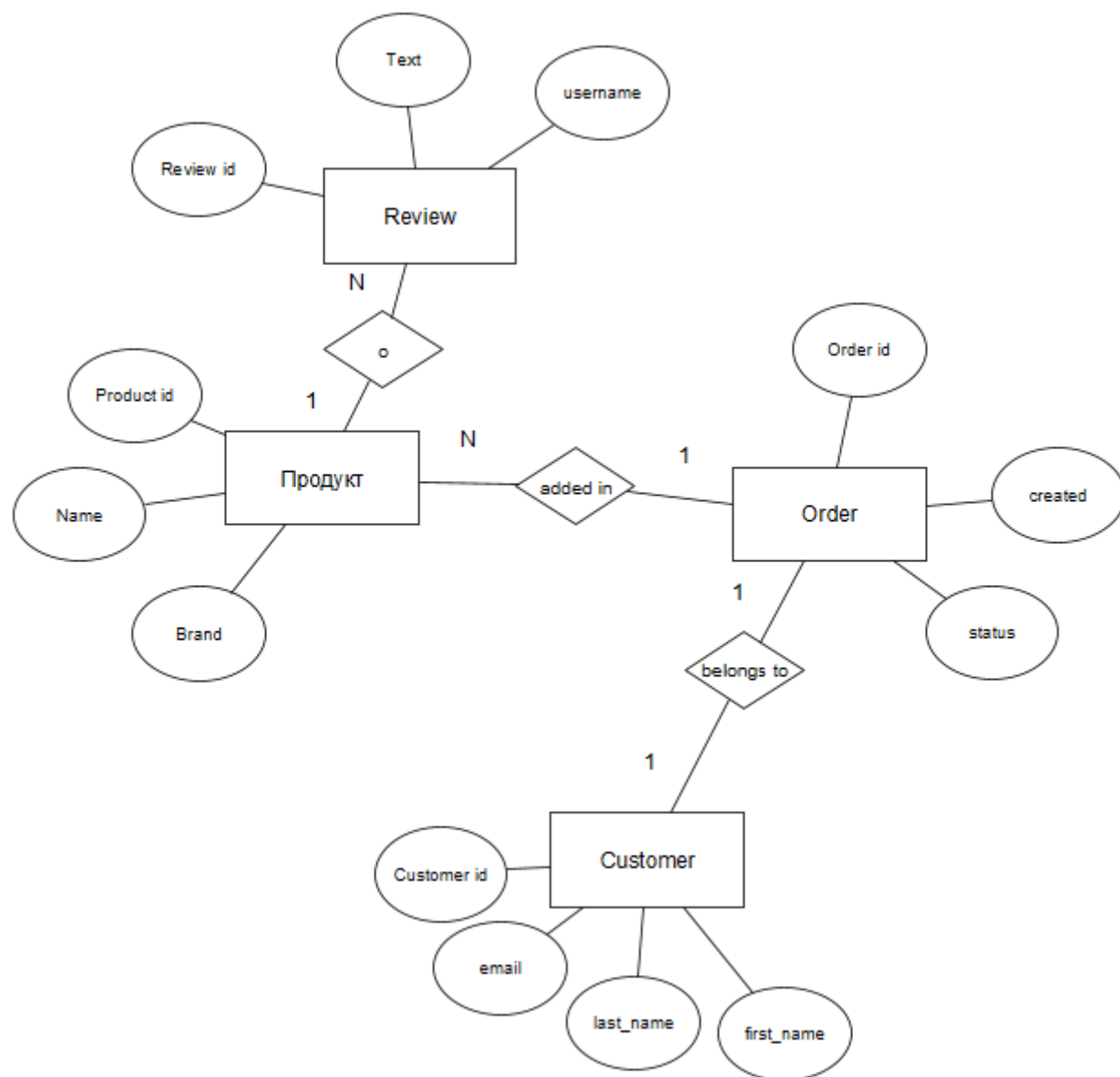


Рисунок 5 ER Диаграмма БД

На диаграмме 6 представлена диаграмма БД.

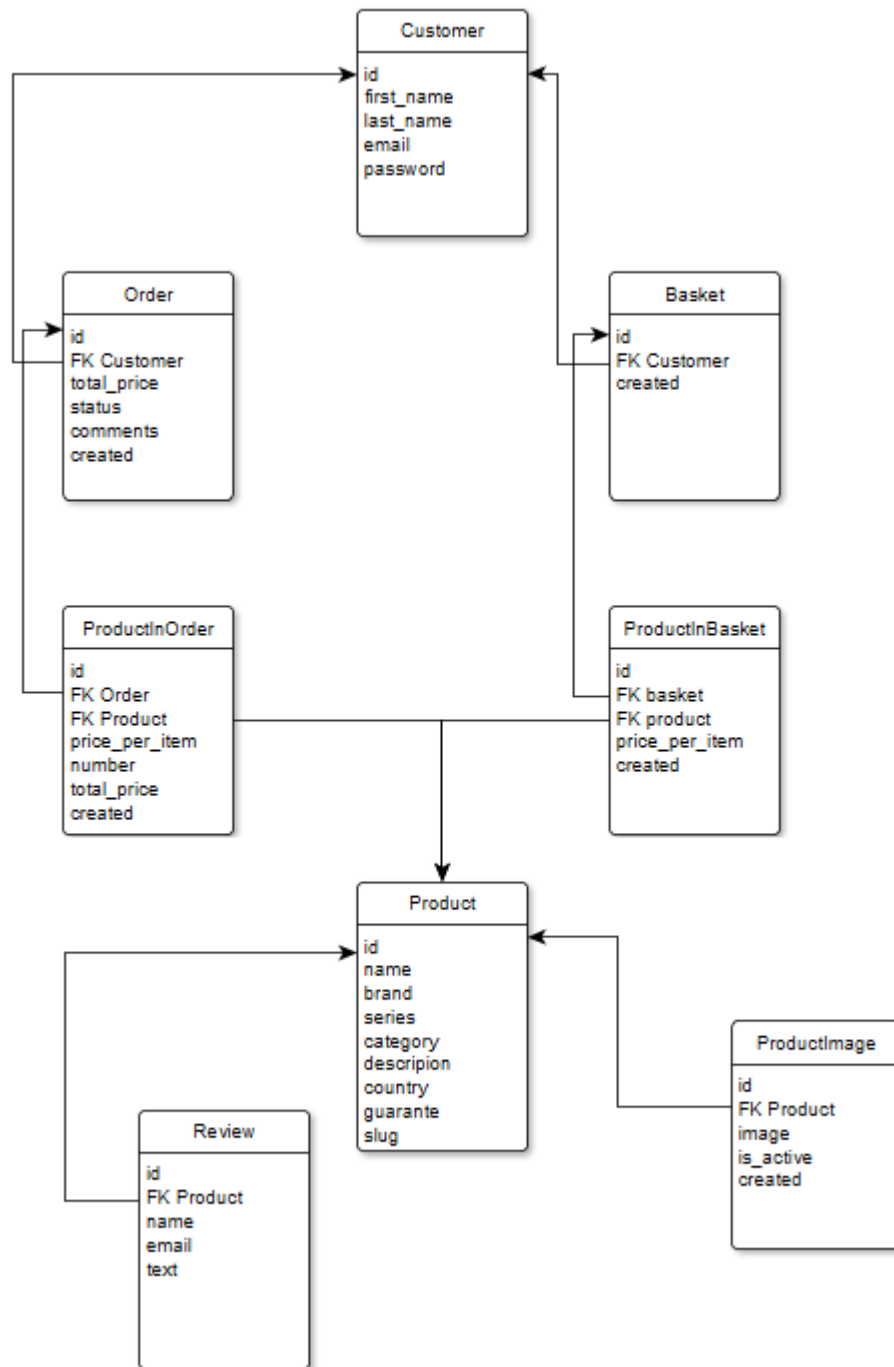


Рисунок 6 Диаграмма БД

2.5 Вывод

Была разработана модель приложения, реализующая интернет-магазин с товаром, пользователями и заказами.

2 Технологическая часть

3.1 Выбор инструментов разработки

Для реализации поставленных задач необходимо выбрать систему управления базами данных (СУБД), язык программирования и фреймворк.

3.1.1 Выбор СУБД

Самыми распространёнными реляционными системами управления базами данных являются:

- SQLite;
- MySQL;
- PostgreSQL.

В данной работе используется PostgreSQL, ориентирующаяся на полное соответствие стандартам и расширяемость. PostgreSQL отлично справляется с одновременной обработкой нескольких заданий. Также к её преимуществам можно отнести:

- Полная SQL-совместимость;
- Расширяемость;
- Объектно-ориентированность.

3.1.2 Выбор ЯП

Самыми распространёнными языками программирования для приложений являются:

- C++;
- C#;
- Python.

В данной работе используется Python, являющийся динамическим языком программирования. Его преимуществами являются

кроссплатформенность, отсутствие этапа компиляции (за счёт чего уменьшается время разработки), а также лаконичность и легко читаемый код.

3.1.3 Выбор фреймворка

Самыми популярным веб фреймворками Python являются Flask и Django. В отличие от Flask, Django поддерживает встроенные формы, которые интегрируются с ORM. Также он доступен со встроенным ORM и системой миграции, которая может управлять базами данных, в то время как Flask нуждается в установке дополнительных инструментов. Django предоставляет приложение аутентификации, которое предоставляет реализацию по умолчанию для пользовательского управления и привилегий. Исходя из описанных преимуществ, в данной работе будет использоваться фреймворк Django.

3.2 Реализация хранения данных

Для работы с базой данных требуется объявить таблицы БД. В Django есть встроенная модель User, которую можно расширить, добавив свои поля. Расширение встроенной модели представлено в Листинге 1.

Листинг 1: Класс «Покупатель»

```
class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='user')
    first_name = models.CharField(max_length = 20, null = True, default = None)
    last_name = models.CharField(max_length = 20, null = True, default = None)
    email = models.EmailField()
    phone = models.CharField(max_length = 15, blank = True, null = True, default = None)
    address = models.CharField(max_length = 40, blank = True, null = True, default = None)
    objects = CustomerManager()
```

Класс «Товар» содержит общие характеристики, имеющиеся у всех продуктов. Реализация данного класса представлена в листинге 2.

Листинг 2: Класс «Товар»


```

class Product(models.Model):
    name = models.CharField(max_length = 30, null = True, default = None)
    copies = models.IntegerField(default = 0)
    brand = models.CharField(max_length=40, null = True, default = None)
    series = models.CharField(max_length=30, null = True, default = None)
    category_choices = [('Laptops', 'Laptop'), ('Phones', 'Phone')]
    category = models.CharField(max_length=10, choices = category_choices)
    price = models.IntegerField(default = 0)
    discount = models.IntegerField(default = 0)
    is_active = models.BooleanField(default=True)
    color = models.CharField(max_length=10, null = True, default = None)
    weight = models.CharField(max_length=10, null = True, default = None)
    country = models.CharField(max_length=15, null = True, default = None)
    guarantee = models.CharField(max_length=10, null = True, default = None)
    description = models.TextField(blank = True, null = True, default = None)

```

Так как в проекте существует разные виды товара («Телефон» и «Ноутбук»), то для компактного их представления требуется создание двух классов, которые будут содержать специфичные характеристики для каждого вида. Реализация таких классов представлена в листингах 3 и 4.

Листинг 3: Класс «Телефон»

```

class ProductPhone(Product):
    product = models.ForeignKey(Product, on_delete=models.CASCADE, related_name = 'product_phone')
    screen = models.CharField(max_length = 25, null = True, default = None)
    dimensions = models.CharField(max_length = 25, null = True, default = None)
    os = models.CharField(max_length = 15, null = True, default = None)
    ROM = models.CharField(max_length = 10, null = True, default = None)
    bluetooth = models.IntegerField(default = 0)

```

Листинг 4: Класс «Ноутбук»

```

class ProductLaptop(Product):
    product = models.ForeignKey(Product, on_delete=models.CASCADE, related_name = 'product_laptop')
    os = models.CharField(max_length = 15, null = True, default = None)
    cpu = models.CharField(max_length = 15, null = True, default = None)
    cores = models.IntegerField(default = 0)

```

```
memory_freq = models.CharField(max_length = 20, null = True, default = None)
RAM = models.CharField(max_length = 20, null = True, default = None)
resolution = models.CharField(max_length = 40, null = True, default = None)
display_diagonal = models.CharField(max_length = 30, null = True, default = None)
```

Реализация класса «Отзыв», содержащего данные об оставленном на товар отзыве, представлена в листинге 5.

Листинг 5: «Класс «Отзыв»»

```
class Reviews(models.Model):
    email = models.EmailField()
    name = models.CharField("Имя", max_length=100)
    text = models.TextField("Сообщение", max_length=5000)
    product = models.ForeignKey(Product, verbose_name="Продукт", on_delete=models.CASCADE)
    created = models.DateTimeField(auto_now_add = True, auto_now = False)
```

Листинг 6: Класс «Заказ»

```
class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE, default = None)
    total_price = models.DecimalField(max_digits = 10, decimal_places = 2, default = 0)
    status = models.ForeignKey(Status, on_delete=models.CASCADE)
    commets = models.TextField(blank = True, null = True, default = None)
```

Листинг 7: Класс «Товар в заказе»

```
class ProductInOrder(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete= models.CASCADE)
    number = models.IntegerField(default = 0)
    is_active = models.BooleanField(default=True)
    price_per_item = models.PositiveIntegerField(default = 1)
    total_price = models.DecimalField(max_digits = 10, decimal_places = 2, default = 0)
    created = models.DateTimeField(auto_now_add = True, auto_now = False)
    updated = models.DateTimeField(auto_now_add = False, auto_now = True)
```

3.3 Frontend разработка

Пользовательский интерфейс представляет собой вёрстку проекта.

HTML отображает язык разметки гипертекста. «Язык разметки» означает, что HTML использует теги для идентификации различных типов контента и целей, которые каждый преследует на веб-странице.

CSS — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML).

JavaScript – это логический язык программирования, который можно использовать для изменения содержимого веб-сайта. Общее использование JavaScript включает в себя окна подтверждения, призывы к действию и добавление новых идентификаторов к существующей информации.

Шаблоны Django предоставляют теги, которые повторяют некоторые структуры языка программирования. Шаблон содержит переменные, которые будут заменены значениями при выполнении шаблона, и теги, которые управляют логикой шаблона. Переменные выглядят таким образом: `{{ variable }}`. Когда шаблон встречает переменную, он вычисляет ее и заменяет результатом. Названия переменных могут состоять из букв, цифр и нижнего подчеркивания (`"_"`).[3]

Листинг 8: Вывод товаров

```
{% for product in product_list %}
<div class="col-md-4 col-xs-6">
  <div class="product">
    <div class="product-img">
      <a href = "{{ product.get_absolute_url }}"><img src= {{ product.productimage.first.image.url }}
width="250" height ="240" alt=""></a>
      {% if product.discount > 0 %}
        <div class="product-label">
          <span class="sale">Скидка</span>
        </div>
      {% endif %}
    </div>
  </div>
</div>
```

```

<div class="product-body">
    <p class="product-category">Category</p>
    <h3 class="product-name"><a href="{{ product.get_absolute_url }}"> {{ product.name }}</a></h3>
    <h4 class="product-price">{{ product.price }}₽ {% if product.discount > 0 %} <del class="product-
old-price">{{ product.discount }}₽</del>{% endif %}</h4>

    <div class="product-btns">
        <button class="add-to-wishlist"><i class="fa fa-heart-o"></i><span class="tooltip">В
желаемое</span></button>
    </div>
</div>
<div class="add-to-cart">
    <button class="add-to-cart-btn"><i class="fa fa-shopping-cart"></i>В корзину</button> </div>
</div>
{% endfor %}

```

3.4 Доступ к данным

В листинге 9 представлено добавление товара в заказ. Если такой товар уже существует, то увеличивается его количество, иначе товар просто добавляется в заказ. Если заказа нет, то он создаётся. Аналогично реализовано удаление из заказа.

Листинг 9 Доступ к данным

```

def add_to_order(request, slug):
    item = get_object_or_404(Product, slug = slug)
    order_item, created = ProductInOrder.objects.get_or_create(product=item,
customer_id=request.user.id)
    order_qs = Order.objects.filter(customer_id=request.user.id)
    if order_qs.exists():
        order = order_qs[0]
        if ProductInOrder.objects.filter(product__slug=item.slug).exists():
            messages.info(request, "Вы добавили этот товар в заказ:")
            order_item.number += 1
            if order_item.order != order_qs[0]:

```

```

        order_item.order = order_qs[0]
        order_item.save()
        price = order_qs[0].total_price
        order_qs.update(total_price = price + order_item.price_per_item)
        return redirect("product_detail", slug=slug)
    else:
        messages.info(request, "Вы добавили этот товар в заказ:")
        order_item.order = order_qs[0]
        price = order_qs[0].total_price
        order_qs.update(total_price = price + order_item.price_per_item)
        order_item.number = 1
        order_item.save()
        return redirect("product_detail", slug=slug)
    else:
        status = Status.objects.all()
        ordered_date = timezone.now()
        messages.info(request, "Вы добавили этот товар в заказ:")
        neworder = Order.objects.create(customer_id=request.user.id, created =
ordered_date, status_id=3)
        price = neworder.total_price + order_item.price_per_item
        neworder.total_price = price
        neworder.save()
        order_item.order = neworder
        order_item.number = 1
        order_item.save()
        return redirect("product_detail", slug=slug)

```

3.5 Интерфейс программы

Незарегистрированные пользователи могут просматривать и фильтровать товары, но не могут добавлять их в желаемое или заказывать. Разница шапки главной страницы показана на рисунках 6 и 7.

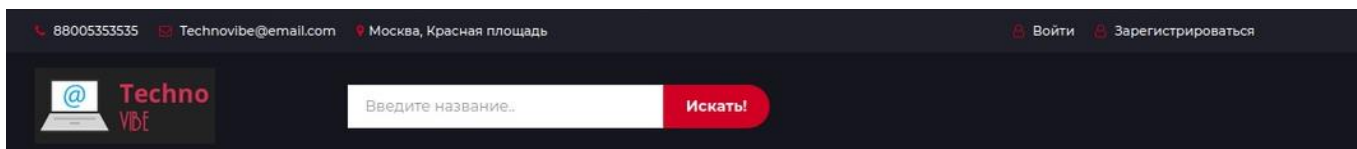


Рисунок 6 Шапка страницы для незарегистрированного пользователя

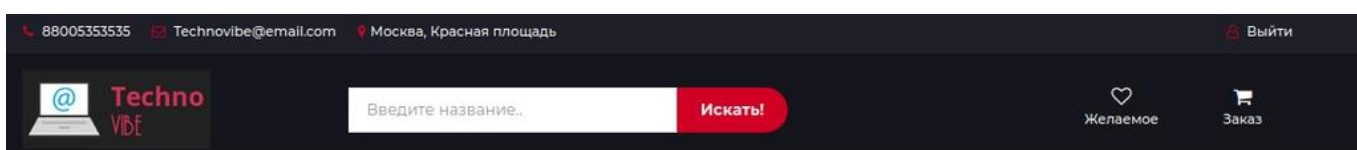


Рисунок 7 Шапка страницы для зарегистрированного пользователя

На рисунках 8 и 9 представлены формы для входа в аккаунт и регистрации.

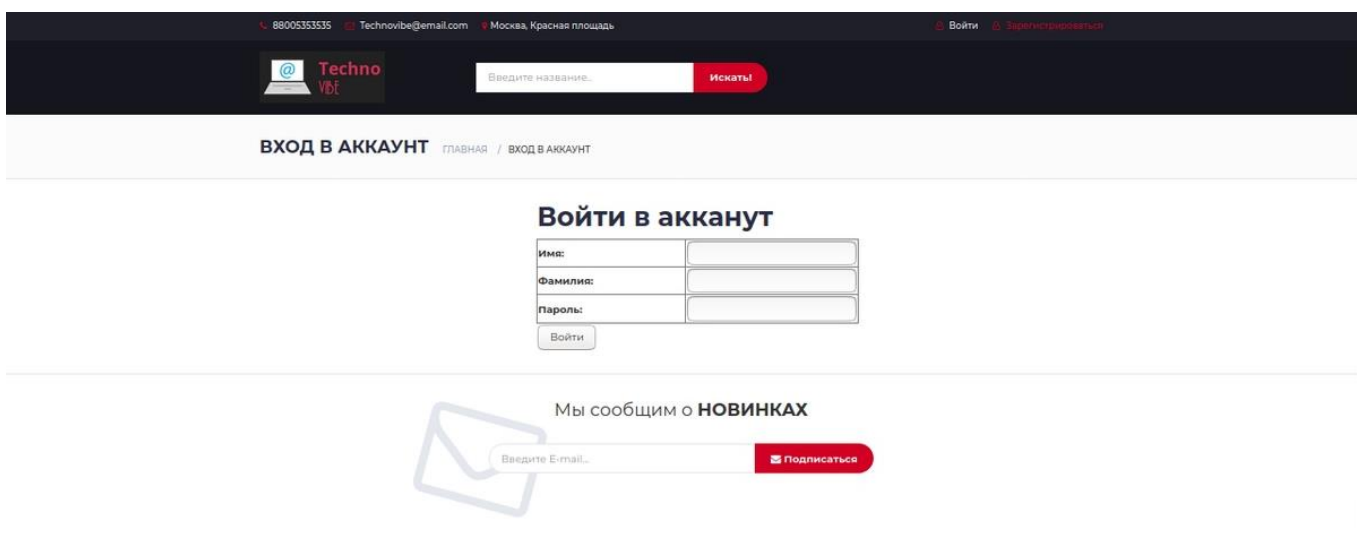


Рисунок 8 Форма входа в аккаунт

88005353535 Technovibe@gmail.com Москва, Красная площадь Войти Зарегистрироваться

Techno VIBE

Введите название... Искать

РЕГИСТРАЦИЯ ГЛАВНАЯ / РЕГИСТРАЦИЯ

Зарегистрироваться

Имя:	<input type="text"/>
Фамилия:	<input type="text"/>
Почта:	<input type="text"/>
Пароль:	<input type="password"/>
Повторите пароль:	<input type="password"/>

Зарегистрироваться

Мы сообщим о **НОВИНКАХ**

Введите E-mail... Подписаться

Рисунок 9 Форма регистрации

В проекте реализована фильтрация товаров по разным параметрам. Пример показан на рисунке 10: выводится все телефоны фирмы Apple

Категории

☐ Laptops (add)

☒ Phones (add)

БРЕНД

☒ Apple (578)

☐ Lenovo (578)


☐ Samsung (578)

Найти

ЦЕНА

+ -

Скидка




PHONES

APPLE IPHONE 7 32GB

25990P ~~30000P~~

♡



PHONES

APPLE IPHONE X 64GB

34990P

♡

Рисунок 10 Результат фильтра по параметрам

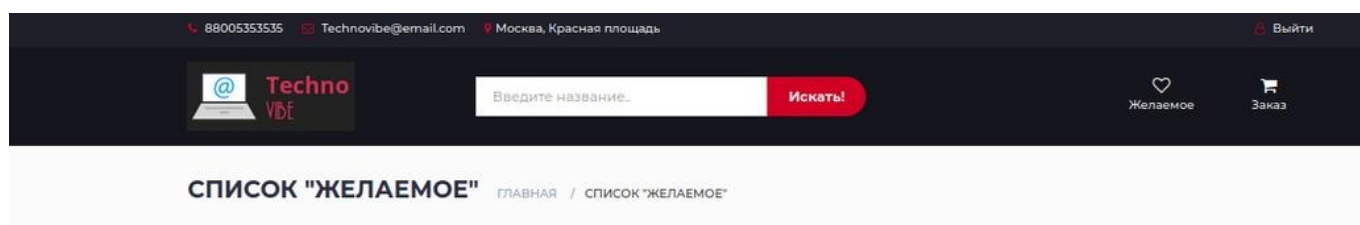


Рисунок 11 Список «Желаемое»

Рассмотрим вид административной панели. На рисунке 12 представлен список заказов, доступный со страницы администратора. Администратор может изменять данные заказа, что показано на рисунке 13.

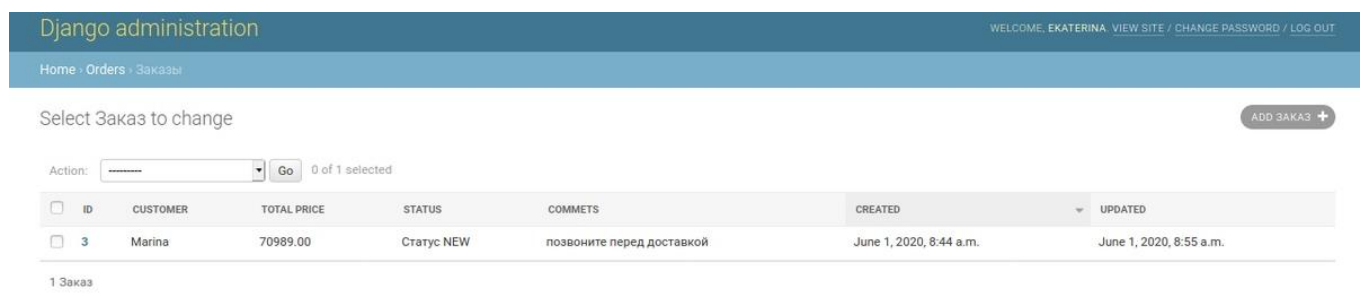


Рисунок 12 Список заказов с административной панели

Change Заказ

HISTORY

Customer:

Marina

Total price:

34990.00

Status:

Статус NEW

Comments:

позвоните перед доставкой

ТОВАРЫ В ЗАКАЗЕ

PRODUCT	NUMBER	IS ACTIVE	PRICE PER ITEM	TOTAL PRICE	DELETE?
<div>Apple iPhone X 64GB</div> <div>Apple iPhone X 64GB</div>	<div>1</div> <div></div> <div></div>	<input checked="" type="checkbox"/>	<div>34990</div> <div></div> <div></div>	<div>34990.00</div> <div></div> <div></div>	<input type="checkbox"/>

+

Add another Товар в заказе

Delete

Save and add another

Save and continue editing

SAVE

Рисунок 13 Изменение заказа с панели администратора

3.5 Вывод

В данном разделе приведены листинги моделей БД. Показаны примеры реализованного интерфейса и его функций.

Заключение

В результате проведённой работы:

- Спроектирована БД для данного проекта;
- Проведён анализ инструментов для данной задачи, в результате которого были выбран язык программирования Python, фреймворк Django и система управления БД PostgreSQL;

- Реализована структура базы данных;
- Разработано клиент-серверное приложение «Technovibe», являющееся интернет-магазином. Реализованы такие возможности, как регистрация и вход в аккаунт; просмотр всего списка товаров; поиск по заданным параметрам; добавление товара в список желаемого и удаление из него; оформление заказа.

Список использованной литературы

1. Карпова И. П. Основы баз данных. 2009
2. Грошев А.С. Базы данных. 2005
3. Документация Django [Электронный ресурс] URL: <https://docs.djangoproject.com/en/3.0/> (Дата обращения: 20.05.2020)
4. Вильям Винсет С. Django for Beginners: Build websites with Python and Django. 2020

5. Руководство по веб-фреймворку Django [Электронный ресурс]
URL: <https://metanit.com/python/django/> (Дата обращения: 29.05.2020)
6. Грубер М. Понимание SQL. 2012
7. Коннолли Т., Бегг К. Базы данных. 2003
8. Документация Python [Электронный ресурс] URL:
<https://www.python.org/doc/> (Дата обращения: 31.05.2020)
9. Документация PostgreSQL [Электронный ресурс] URL:
<https://postgrespro.ru/docs/postgresql> (Дата обращения: 01.06.2020)