

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на учебную практику
по ПМ.11 «Разработка, администрирование и защита баз данных»

Студент Махоткин А.П. Группа № У2335
(Фамилия И. О.)

Руководитель Ефимова Т.Н., преподаватель факультета СПО
Говоров А.И., преподаватель факультета СПО

Тема задания: Проектирование и реализация базы данных.

Сроки прохождения практики: 02.02.2020 -02.07.2020

Место прохождения практики: Факультет СПО

1. Виды работ и требования к их выполнению:

Учебная практика проводится распределенно (понедельно в течение семестра) на базе факультета СПО в лаборатории разработки баз данных. В ходе прохождения практики выполняются следующие виды работ:

- I. Вводный инструктаж по технике безопасности и общим целям, и задачам практики.
- II. Анализ поставленной задачи
- III. Выполнение индивидуального задания: проектирование БД, разработка прототипа веб-приложения.
- IV. Формирование отчета по учебной практике.

2. Виды отчетных материалов и требования к их оформлению:

По результатам прохождения практики составляется отчет, в котором представляются индивидуальное задание, модель базы данных, перечень использованных технологий, программных средств, использованных паттернов (шаблонов) проектирования программ, программный код, описание результатов работы программы. Оформление отчета должно соответствовать Рекомендациям по оформлению технических документов факультета СПО Университета ИТМО.

3. ПЛАН-ГРАФИК

№ эта па	Наименование этапа	Срок завершения этапа	Виды работ	Форма отчетности
1	2	3	4	5
1.	Вводный инструктаж	02.02.2020 – 09.02.2020	Ознакомление с инструкцией по технике безопасности. Ознакомление с целями и задачами производственной практики	Журнал по технике безопасности
2.	Постановка задачи	09.02.2020 – 09.03.2020	Анализ индивидуального задания. Обследование предметной области согласно индивидуальной теме учебной практики.	Отчет по практике: индивидуальное задание Дневник практики
3.	Моделирование базы данных и реализация	10.03.2020 – 31.03.2020	Описание предметной области. Создание диаграммы классов Создание таблиц Заполнение таблиц данными (команды)	Отчет по практике: индивидуальное задание Дневник практики
4.	Реализация модели данных средствами Django ORM	01.04.2020 – 07.04.2020	Создание модели Django в соответствии с моделью данных и настройка связи между таблицами	Отчет по практике: индивидуальное задание Дневник практики
5.		82.04.2020 – 21.04.2020	Реализация элементов инфраструктуры Django, в соответствии с архитектурным паттерном Model-View-Template или сокращенно MVT. Реализация интерфейсов к системе средствами Django Templates или сторонними средствами.	Отчет по практике: индивидуальное задание Дневник практики
6.	Подготовка отчетных материалов	17.06.2020 – 23.06.2020	Формирование отчета о практике	Отчет по практике: индивидуальное задание Дневник практики
7.	Защита результатов практики	24.06.2020 – 02.07.2020	Защита результатов практики в форме устного собеседования и представления результатов практики	

Задание утверждено председателем выпускающей комиссии факультета СПО
 Председатель выпускающей комиссии факультета СПО _____ Королев В.В.
 «___» _____ 20__ г

Дата выдачи задания: _____

Руководитель от факультета _____
 (подпись руководителя)

Задание принял к
 исполнению _____
 (подпись студента)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

Направление подготовки (специальность) 09.02.07 Информационные системы и программирование

О Т Ч Е Т

об учебной практике по профессиональному модулю
ПМ.11 «Разработка, администрирование и защита баз данных»

Тема задания: Разработка и реализация базы данных по предметной области

Обучающийся Махоткин А.П. Группа У2335
(Фамилия И.О.) (номер группы)

Руководитель практики: Ефимова Т.Н., преподаватель факультета СПО
Говоров А.И., преподаватель факультета СПО

Ответственный за практику от университета: Королев В.В. зам. директора факультета СПО

Практика пройдена с оценкой _____

Подписи членов комиссии _____ (_____) (подпись)

_____ (А.И.Говоров) (подпись)

_____ (Т.Н.Ефимова) (подпись)

Дата _____

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ	6
1.1 Формулировка поставленной задачи	6
1.2 Описание предметной области.....	6
2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.....	7
3 ТЕХНОЛОГИИ И ПРОГРАММНЫЕ СРЕДСТВА	8
2.1 Используемые технологии	8
2.2 Программные средства	8
4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	9
ЗАКЛЮЧЕНИЕ	14
СПИСОК ЛИТЕРАТУРЫ.....	15
ПРИЛОЖЕНИЕ А	16
ПРИЛОЖЕНИЕ Б.....	23

ВВЕДЕНИЕ

Целью учебной практики по профессиональному модулю ПМ.11 «Разработка, администрирование и защита баз данных» является углубление знаний и практических умений и получение начального практического опыта по основным видам деятельности «Разработка, администрирование и защита баз данных» и овладение соответствующими общими и профессиональными компетенциями: ОК 1, ОК 2, ОК 3, ОК 4, ОК 5, ОК 6, ОК 7, ОК 8, ОК 9, ОК 10, ОК 11, ПК 11.1., ПК 11.2., ПК 11.3, ПК 11.4., ПК 11.5., ПК 11.6. (см. рабочая программа и фонд оценочных средств по производственной практике).

Учебная практика проводится на базе факультета СПО Университета ИТМО.

Результатом практики является разработка прототипа веб-приложения по заданной предметной области, использующего реляционную базу данных.

Задачи:

1. Спроектировать базу данных.
2. Описать модель данных приложения.
3. Описать методы получения, вставки, редактирования и удаления данных.
4. Описать внешнюю оболочку приложения.
5. Упаковать приложение в Docker.

1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1.1 Формулировка поставленной задачи

Разработать веб-приложение на Django в соответствии с вариантом.

1.2 Описание предметной области

БД “Издательство компьютерной литературы” Описание предметной области: Издательство занимается выпуском литературы по различным областям информатики. Покупатели книг приобретают книги на базе издательства. Когда на базе заканчиваются книги, издается дополнительный тираж.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

В процессе проектирования базы данных была использована методология «сущность-связь» и соответствующая ей нотация.

При анализе предметной области были выделены следующие сущности, обозначенные в инфологической модели на рисунке А.1 в приложении А:

- авторы;
- книги;
- авторы книг;
- категории книг;
- тиражи;
- заказчики;
- заказы;
- книги в заказе;
- отзывы о сайте.

Логическая модель соответствует первой нормальной форме, так как все атрибуты атомарные. Описание логической модели в приложении А в таблицах А.1-А.9.

3 ТЕХНОЛОГИИ И ПРОГРАММНЫЕ СРЕДСТВА

2.1 Использованные технологии

При разработке системы были использованы следующие технологии:

- SQLite;
- Django;
- django-crispy-forms;
- Bootstrap 4.

2.2 Программные средства

При проектировании базы данных был использован MySQL Workbench 8.0 CE – унифицированный визуальный инструмент для разработки и администрирования баз данных.

Для реализации системы были использованы следующие программные средства:

- JetBrains PyCharm 2020.1.2 x64 – IDE для профессиональной разработки на Python;
- Docker – открытая платформа для разработки, доставки и запуска приложений.

4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Для реализации системы была описана модель данных на языке Python с использованием фреймворка Django, представленная в листинге Б.1 приложения Б.

После чего были разработаны пользовательские интерфейсы.

Программный код представлен в приложении Б.

Главная страница представлена на рис. 1.

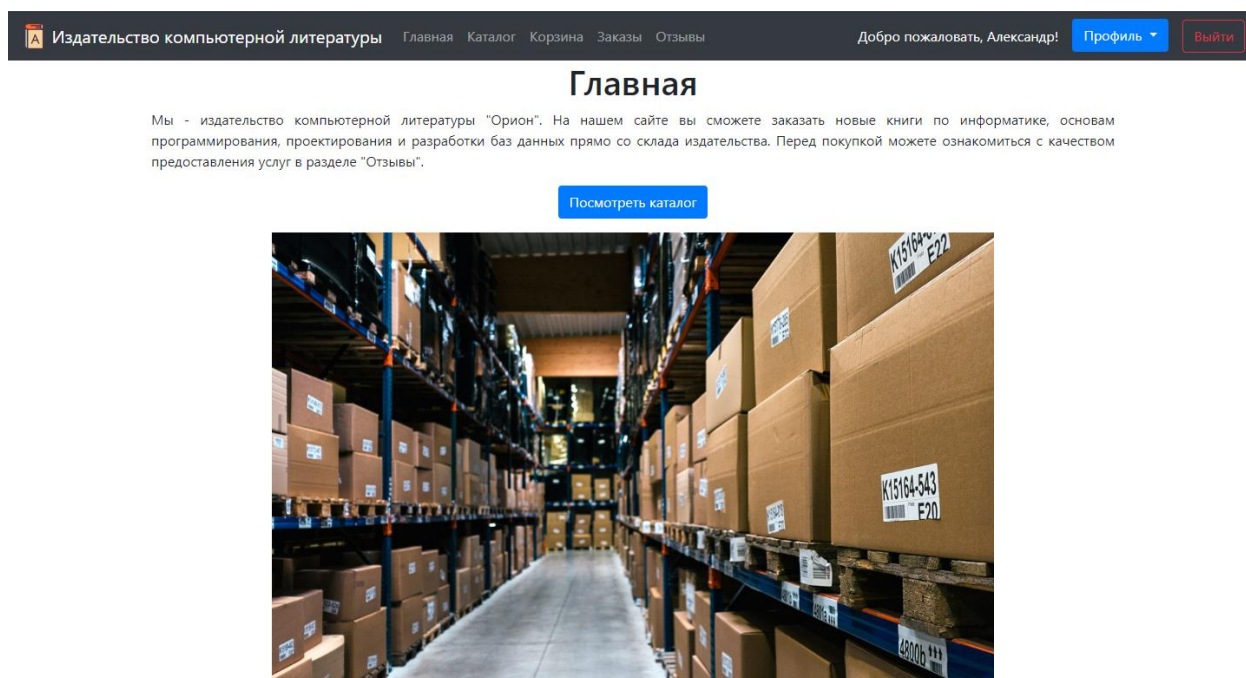


Рисунок 1 – Главная страница

Страница каталога представлена на рис. 2.

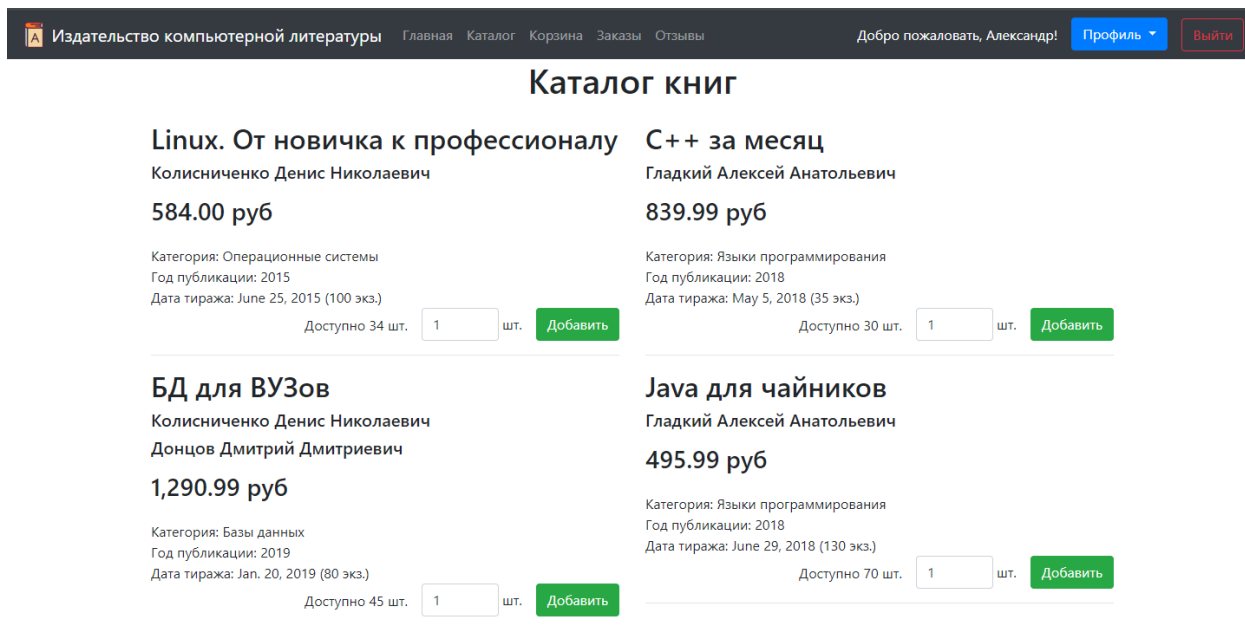


Рисунок 2 – Страница «Каталог книг»

Страница «корзина» представлена на рис. 3.

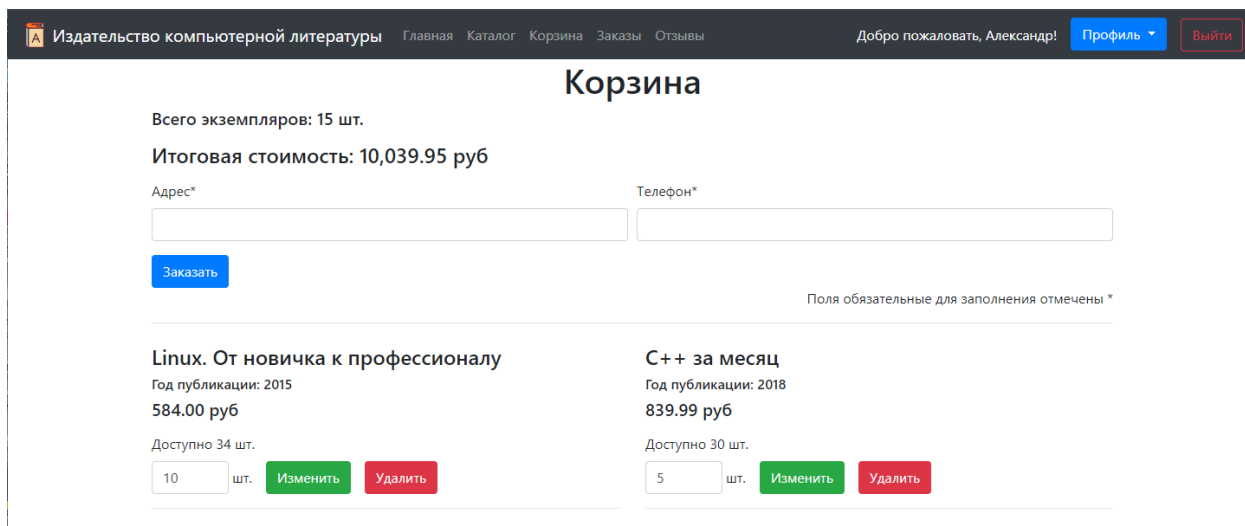


Рисунок 3 – «Корзина» покупателя

Страница «заказы» представляет собой список всех оформленных заказов покупателя с указанием купленных книг, их количества, цену во время покупки за экземпляр, а также суммарное количество экземпляров книг и итоговую стоимость заказов (рис. 4).

Заказы

Заказ от 03-07-2020 15:15:43

Название книги	Цена за экз.	Количество
Java для чайников	495.99 руб	30 шт.
Linux. От новичка к профессионалу	584.00 руб	50 шт.

Всего экземпляров: 80 шт.

Стоимость заказа: 44,079.70 руб

Заказ от 01-07-2020 22:22:01

Название книги	Цена за экз.	Количество
Java для чайников	495.99 руб	10 шт.
Linux. От новичка к профессионалу	584.00 руб	10 шт.

Всего экземпляров: 20 шт.


Стоимость заказа: 10,799.90 руб

Заказ от 01-07-2020 21:21:10

Название книги	Цена за экз.	Количество
Linux. От новичка к профессионалу	584.00 руб	1 шт.

Рисунок 4 – Страница «Заказы»

На странице «отзывы» заказчик может оставлять своё мнение о сервисе (рис. 5).


Издательство компьютерной литературы

[Главная](#)
[Каталог](#)
[Корзина](#)
[Заказы](#)
[Отзывы](#)

Добро пожаловать, Александр!
[Профиль](#)
[Выйти](#)

Отзывы

Текст*

Оценка (из 10)*

Опубликовать

Поля обязательные для заполнения отмечены *

June 29, 2020

Александр

Отличный сервис, быстрая доставка!

Оценка: 10 / 10

Рисунок 5 – Страница отзывов

Пользователю доступно редактирование данных своего профиля на сайте, а также смена пароля. Страницы «редактирование профиля» и «изменение пароля» представлены на рис. 6, 7.

Издательство компьютерной литературы

Главная

Каталог

Корзина

Заказы

Отзывы

Добро пожаловать, Александр!

Профиль

Выйти

Редактирование профиля

Имя*

Александр

Фамилия*

Махоткин

Эл. почта*

admin@admin.com

Адрес

Телефон

Сохранить

Поля обязательные для заполнения отмечены *

Рисунок 6 – Страница редактирования профиля

Рисунок 7 – Страница изменения пароля

Действия пользователя сопровождаются сообщениями вверху страницы. Пример сообщения представлен на рис. 8.

Каталог книг

Linux. От новичка к профессионалу	C++ за месяц
Колисниченко Денис Николаевич	Гладкий Алексей Анатольевич

Рисунок 8 – Пример сообщения

Пользователь без аккаунта на сайте не может делать заказы и оставлять отзывы. На главной странице ему будет предложено зарегистрироваться для совершения покупок.

Также на сайте предусмотрена защита от задержки обновления количества книг на складе.

ЗАКЛЮЧЕНИЕ

В ходе выполнения индивидуального задания были углублены знания и получен начальный опыт по разработке, администрированию и защиты баз данных, а также по созданию веб-приложения на языке Python с помощью фреймворка Django. Был разработано веб-приложение по заданной предметной области, использующего реляционную базу данных.

В ходе разработки была спроектирована база данных. Готовый прототип был упакован в Docker, успешно запущен и протестирован.

СПИСОК ЛИТЕРАТУРЫ

1. Всё, что вы хотели знать о фреймворке Django и его библиотеках [Электронный ресурс] // Django.Fun URL: <https://django.fun/> (дата обращения: 24.06.2020).
2. Django documentation [Электронный ресурс] // Django URL: <https://docs.djangoproject.com/en/3.0/> (дата обращения: 23.06.2020).
3. Introduction Bootstrap [Электронный ресурс] // Bootstrap URL: <https://getbootstrap.com/docs/4.0/> (дата обращения: 23.06.2020).
4. Get Docker [Электронный ресурс] // docker docs URL: <https://docs.docker.com/get-docker/> (дата обращения: 20.06.2020).
5. Django rest framework [Электронный ресурс] // YouTube URL: <https://www.youtube.com/playlist?list=PLF-NY6ldwAWqP4S95brtPHZ5fTCxilgei> (дата обращения: 12.06.2020).

ПРИЛОЖЕНИЕ А

На рисунке А.1 представлена инфологическая схема базы данных.

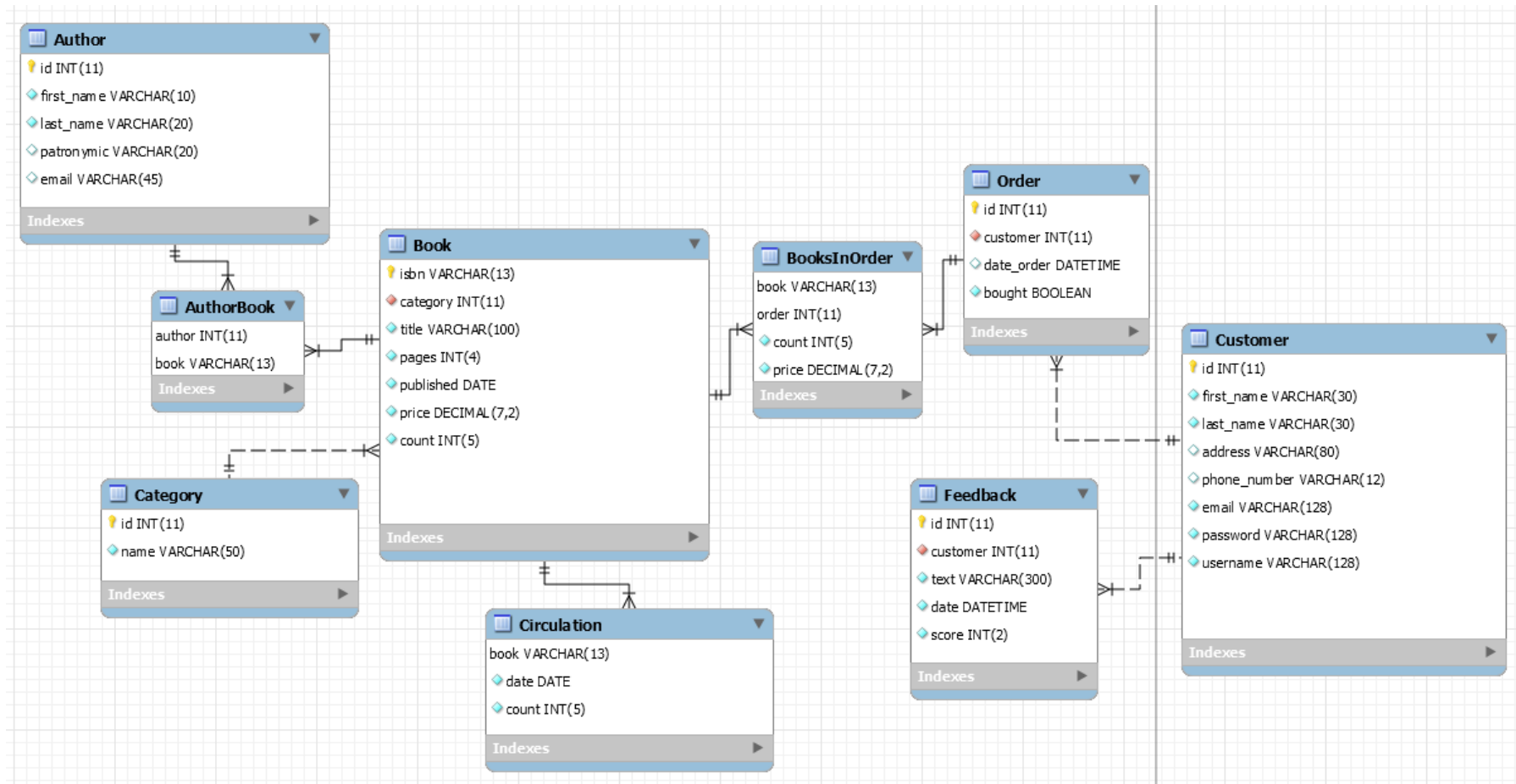


Рисунок А.1 - Инфологическая схема базы данных

В таблицах А.1-А.9 представлены описания всех полей таблиц.

Таблица А.1 – Автор

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
id	INT(11)	+	+	-	Уникален, число	Идентификатор
first_name	VARCHAR(10)	+	-	-	Строковое значение	Имя
last_name	VARCHAR(20)	+	-	-	Строковое значение	Фамилия
patronymic	VARCHAR(20)	-	-	-	Строковое значение	Отчество
email	VARCHAR(45)	-	-	-	Строковое значение	Адрес эл. почты

Таблица А.2 – Автор-книга

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
author	INT(11)	+	+	+	Уникален, число	Автор книги
book	VARCHAR(13)	+	+	+	Уникален, строковое значение	Книга

Таблица А.3 – Книга

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
isbn	VARCHAR(13)	+	+	-	Уникален, строковое значение	Уникальный isbn-номер книги
category	INT(11)	+	-	+	Число	Категория книги
title	VARCHAR(100)	+	-	-	Строковое значение	Название книги
pages	INT(4)	+	-	-	Число	Кол-во страниц
published	DATE	+	-	-	Дата	Дата публикации
price	DECIMAL(7,2)	+	-	-	Дробное значение	Цена
count	INT(5)	+	-	-	Число	Количество на складе

Таблица А.4 – Категория книги

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
id	INT(11)	+	+	-	Уникален, число	Идентификатор
name	VARCHAR(50)	+	-	-	Строковое значение	Название категории

Таблица А.5 – Тираж

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
id	INT(11)	+	+	-	Уникален, число	Идентификатор
book	VARCHAR(13)	+	-	+	Строковое значение	Книга
date	DATE	+	-	-	Дата	Дата тиража
count	INT(5)	+	-	-	Число	Количество

Таблица А.6 – Заказчик

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
id	INT(11)	+	+	-	Уникален, число	Идентификатор
first_name	VARCHAR(30)	+	-	-	Строковое значение	Имя
last_name	VARCHAR(30)	+	-	-	Строковое значение	Фамилия
address	VARCHAR(80)	-	-	-	Строковое значение	Адрес заказчика
phone_number	VARCHAR(12)	-	-	-	Строковое значение	Номер телефона
email	VARCHAR(128)	+	-	-	Строковое значение	Адрес эл. почты
password	VARCHAR(128)	+	-	-	Строковое значение	Пароль
username	VARCHAR(128)	+	-	-	Строковое значение	Имя пользователя

Таблица А.7 – Заказ

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
id	INT(11)	+	+	-	Уникален, число	Идентификатор
customer	INT(11)	+	-	+	Число	Заказчик
date_order	DATETIME	-	-	-	Время и дата	Дата заказа
bought	BOOLEAN	+	-	-	Принимает значения только true или false	Статус покупки

Таблица А.8 – Книги в заказе

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
book	VARCHAR(13)	+	+	+	Уникален, строковое значение	Книга
order	INT(11)	+	+	+	Уникален, число	Заказ
count	INT(5)	+	-	-	Число	Количество книг в заказе
price	DECIMAL(7,2)	+	-	-	Дробное число	Цена книги на момент покупки

Таблица А.9 – Отзывы

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения
id	INT(11)	+	+	-	Уникален, число	Идентификатор
customer	INT(11)	+	-	+	Число	Заказчик
text	VARCHAR(300)	+	-	-	Строковое значение	Текст отзыва
date	DATETIME	+	-	-	Время и дата	Дата отзыва
score	INT(2)	+	-	-	Число	Оценка

ПРИЛОЖЕНИЕ Б

В листинге Б.1-Б.5 представлен программный код разработки.

Листинг Б.1 – Модель данных

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.

class Category(models.Model):
    name = models.CharField(max_length=50, null=False)

class Author(models.Model):
    first_name = models.CharField(max_length=10, null=False)
    last_name = models.CharField(max_length=20, null=False)
    patronymic = models.CharField(max_length=20, null=True)
    email = models.CharField(max_length=45, null=True)

class Customer(AbstractUser):
    address = models.CharField(max_length=80, null=True)
    phone_number = models.CharField(max_length=12, null=True)

class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE,
null=False)
    date_order = models.DateTimeField(auto_now=False, null=True)
    bought = models.BooleanField(null=False, default=False)

class Book(models.Model):
    isbn = models.CharField(max_length=13, primary_key=True)
    authors = models.ManyToManyField(Author)
    category = models.ForeignKey(Category, on_delete=models.CASCADE,
null=False)
    title = models.CharField(max_length=100, null=False)
    pages = models.DecimalField(max_digits=4, decimal_places=0,
null=False)
    published = models.DateField(null=False)
    price = models.DecimalField(max_digits=7, decimal_places=2,
null=False)
    count = models.DecimalField(max_digits=5, decimal_places=0,
null=False, default=0)
    in_orders = models.ManyToManyField(Order, through='BooksInOrder')

class BooksInOrder(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE,
null=False)
    order = models.ForeignKey(Order, on_delete=models.CASCADE,
null=False)
```

```

        count      =      models.DecimalField(max_digits=5,      decimal_places=0,
null=False)
        price      =      models.DecimalField(max_digits=7,      decimal_places=2,
null=False)

class Feedback(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE,
null=False)
    text = models.CharField(max_length=300, null=False)
    date = models.DateTimeField(auto_now=True, null=False)
    score    =      models.DecimalField(max_digits=2,      decimal_places=0,
null=False, default=10)

class Circulation(models.Model):
    book      =      models.ForeignKey(Book,      on_delete=models.CASCADE,
null=False)
    date = models.DateField(null=False)
    count      =      models.DecimalField(max_digits=5,      decimal_places=0,
null=False)

```

Листинг Б.2 – Файл view.py

```

import datetime

from django.contrib.auth import authenticate
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.contrib import auth, messages
from django.utils.html import strip_tags

from .forms import *
from .models import *

# Create your views here.

def orders(request):
    customer_orders = Order.objects.filter(customer_id=request.user.id, bought=True).order_by('-date_order')

    for order in customer_orders:
        sum_price = 0
        sum_count = 0
        books_in_order = BooksInOrder.objects.filter(order=order).values('book')
        books = Book.objects.filter(isbn__in=books_in_order)
        order.books = books
        for book in books:
            try:
                book_in_order = BooksInOrder.objects.get(order_id=order, book_id=book.isbn)
                book.selected_count = book_in_order.count
                book.bought_price = book_in_order.price
                sum_price += book.selected_count * book.bought_price

```



```

        sum_count += book.selected_count
    except:
        book.selected_count = 0
    order.sum_price = sum_price
    order.sum_count = sum_count

context = {
    'customer_orders': customer_orders,
}
return render(request, 'orders.html', context)

def feedback(request):
    if request.method == 'POST':
        form = FeedbackForm(request.POST)

        if form.is_valid():
            text = strip_tags(form.cleaned_data['text'])
            score = int(form.cleaned_data['score'])
            add_feedback(request, text, score)
            messages.success(request, 'Отзыв опубликован')

        return HttpResponseRedirect(request.path_info)

    form = FeedbackForm()
    form.fields['score'].initial = 10

    feedback_data = Feedback.objects.all().order_by('-date')

    context = {
        'form': form,
        'feedback_data': feedback_data,
    }
    return render(request, 'feedback.html', context)

def add_feedback(request, text, score):
    fb = Feedback()
    fb.customer_id = request.user.id
    fb.text = text
    fb.score = score
    fb.date = datetime.date.today()
    fb.save()
    return

def cart(request):
    if request.POST.get('btn_remove', None) is not None:
        remove_book(request.POST['isbn'], request)
        return HttpResponseRedirect(request.path_info)

    if request.POST.get('btn_edit', None) is not None:
        remove_book(request.POST['isbn'], request)
        status = add_book(request.POST['isbn'], int(request.POST['count']), request)
        if status == 'out':
            messages.error(request, "Книга закончилась на складе")
        elif status == 'less':

```

```

        messages.warning(request, "Изменено на меньше, так как от-
сутствует на складе")
    else:
        messages.success(request, "Количество успешно изменено")
        return HttpResponseRedirect(request.path_info)

if request.method == 'POST':
    form = OrderForm(request.POST)
    if form.is_valid():
        customer = Customer.objects.get(id=request.user.id)
        customer.address = strip_tags(form.cleaned_data['ad-
dress'])
        customer.phone_number = form.cleaned_data['phone']
        customer.save()

        order = get_order_or_create(request)
        count_changes = update_books(order)
        if count_changes > 0:
            messages.warning(request, 'Количество доступных книг
на складе изменилось, некоторые книги могут '
                                'отсутствовать. Проверьте
заказ и нажмите кнопку заказа повторно.')
            return HttpResponseRedirect(request.path_info)

        make_order(order)
        messages.success(request, 'Заказ успешно оформлен')
        return HttpResponseRedirect(request.path_info)

form = OrderForm()
customer = Customer.objects.get(id=request.user.id)
form.fields['address'].initial = customer.address
form.fields['phone'].initial = customer.phone_number

order = get_order_or_create(request)
books_in_order = BooksInOrder.objects.filter(order_id=order).val-
ues('book')
books = Book.objects.filter(isbn__in=books_in_order)

sum_price = 0
sum_count = 0

for book in books:
    try:
        book.selected_count = BooksInOrder.objects.get(or-
der_id=order, book_id=book.isbn).count
        sum_price += book.selected_count * book.price
        sum_count += book.selected_count
    except:
        book.selected_count = None

context = {
    'books': books,
    'sum_price': sum_price,
    'sum_count': sum_count,
    'form': form,
}
return render(request, 'cart.html', context)

```

```

def make_order(order):
    books_in_order = BooksInOrder.objects.filter(order_id=order)
    for book_in_order in books_in_order:
        book = Book.objects.get(isbn=book_in_order.book.isbn)
        book.count -= book_in_order.count
        book.save()
    order.bought = True
    order.date_order = datetime.datetime.now()
    order.save()
    return

def update_books(order):
    books_in_order = BooksInOrder.objects.filter(order_id=order)
    count_changes = 0
    for book in books_in_order:
        new_count = get_optimal_count(book.book.isbn, book.count)
        if book.count != new_count:
            count_changes += 1
    return count_changes

def catalog(request):
    if request.POST.get('btn_add', None) is not None:
        status = add_book(request.POST['isbn'], int(request.POST['count']), request)
        if status == 'out':
            messages.error(request, "Книга закончилась на складе")
        elif status == 'less':
            messages.warning(request, "Добавлено меньше, так как отсутствует на складе")
        else:
            messages.success(request, "Книга успешно добавлена")
        return HttpResponseRedirect(request.path_info)

    if request.POST.get('btn_remove', None) is not None:
        remove_book(request.POST['isbn'], request)
        return HttpResponseRedirect(request.path_info)

    books = Book.objects.all()

    if request.user.is_authenticated:
        order = get_order_or_create(request)
        books_in_order = BooksInOrder.objects.filter(order_id=order)
        for book in books:
            for bio in books_in_order:
                if book.isbn == bio.book_id:
                    book.selected_count = bio.count
            book_circ = Circulation.objects.filter(book=book).order_by('-date').first()
            book.circulation = book_circ
        context = {
            'books': books,
        }
    return render(request, 'catalog.html', context)

```

```

def get_order_or_create(request):
    try:
        order = Order.objects.get(customer_id=request.user.id,
bought=False)
    except:
        order = Order()
        order.customer_id = request.user.id
        order.save()
    return order

def add_book(isbn, count, request):
    order = get_order_or_create(request)

    try:
        book_in_order = BooksInOrder.objects.get(book_id=isbn, or-
der_id=order.id)
    except:
        book_in_order = BooksInOrder()
        book_in_order.book_id = isbn
        book_in_order.order = order
        book_in_order.price = Book.objects.get(isbn=isbn).price

    book_in_order.count = get_optimal_count(isbn, count)
    if book_in_order.count == 0:
        return 'out'
    else:
        book_in_order.save()
        if book_in_order.count != count:
            return 'less'
        return 'success'

def get_optimal_count(isbn, count):
    have_count = Book.objects.get(isbn=isbn).count
    if have_count < count:
        count = have_count
    return count

def remove_book(isbn, request):
    order = get_order_or_create(request)
    book_in_order = BooksInOrder.objects.get(order=order,
book_id=isbn)
    if book_in_order is not None:
        book_in_order.delete()
    return

def registration(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            if form.cleaned_data['password'] ==
form.cleaned_data['password2']:
                user = Customer.objects.create_user(
username=strip_tags(form.cleaned_data['username']),

```

```

        email=strip_tags(form.cleaned_data['email']),
        password=form.cleaned_data['password'],
    )
    user.first_name =
strip_tags(form.cleaned_data['first_name'])
    user.last_name =
strip_tags(form.cleaned_data['last_name'])
    user.save()
    auth.login(request, user)
    messages.success(request, "Вы успешно зарегистрирова-
лись")

    return HttpResponseRedirect('/')
else:
    messages.error(request, "Пароли не совпадают")
else:
    messages.error(request, "Ошибка: не все поля введены")
    return HttpResponseRedirect(request.path_info)
else:
    form = RegistrationForm()
    return render(request, 'registration.html', {'form': form})

```

```

@login_required
def edit_password(request):
    if request.method == 'POST':
        form = PasswordForm(request.POST)
        if form.is_valid():
            username = request.user.username
            password = form.cleaned_data['old_password']
            user_check = authenticate(username=username, pass-
word=password)
            if user_check is not None:
                if form.cleaned_data['new_password'] ==
form.cleaned_data['new_password2']:
                    user = Customer.objects.get(id=request.user.id)
                    user.set_password(form.cleaned_data['new_pass-
word'])

                    user.save()
                    messages.success(request, "Пароль успешно изме-
нён")

                    return HttpResponseRedirect('/')
            else:
                messages.error(request, "Пароли не совпадают")
        else:
            messages.error(request, "Введён неверный пароль")
            return HttpResponseRedirect(request.path_info)
    else:
        form = PasswordForm()
        return render(request, 'edit_password.html', {'form': form})

```

```

@login_required
def edit_profile(request):
    if request.method == 'POST':
        form = ProfileForm(request.POST)
        if form.is_valid():
            user = Customer.objects.get(id=request.user.id)
            user.first_name =

```

```

strip_tags(form.cleaned_data['first_name'])
        user.last_name =
strip_tags(form.cleaned_data['last_name'])
        user.email = strip_tags(form.cleaned_data['email'])
        user.phone_number = form.cleaned_data['phone']
        user.address = strip_tags(form.cleaned_data['address'])
        user.save()
        messages.success(request, "Данные профиля сохранены")
        return HttpResponseRedirect(request.path_info)
    else:
        messages.error(request, "Ошибка: не все поля введены")
else:
    form = ProfileForm()
    user = Customer.objects.get(id=request.user.id)
    form.fields['first_name'].initial = user.first_name
    form.fields['last_name'].initial = user.last_name
    form.fields['email'].initial = user.email
    form.fields['phone'].initial = user.phone_number
    form.fields['address'].initial = user.address
    return render(request, 'edit_profile.html', {'form': form})

def main_page(request):
    state = login(request)
    if not state:
        return render(request, 'main_page.html')
    elif state == 'login':
        return HttpResponseRedirect('/')
    else:
        return HttpResponseRedirect('/')

def login(request):
    btn_auth = request.POST.get('btn_auth', None)
    btn_deauth = request.POST.get('btn_deauth', None)
    if btn_auth is not None and request.user.is_authenticated ==
False:
        username = request.POST['username']
        password = request.POST['password']
        user = auth.authenticate(username=username, password=password)
        if user is not None and user.is_active:
            auth.login(request, user)
        else:
            messages.error(request, "Логин или пароль неверный!")
            return 'login'
    if btn_deauth is not None and request.user.is_authenticated ==
True:
        auth.logout(request)
        return 'logout'
    return False

```

Листинг Б.3 – Файл urls.py

```

from django.conf.urls import url
from django.urls import path
from django.views.generic import RedirectView

from .views import *

```

```
urlpatterns = [
    path('', main_page, name='main_page'),
    path('profile/edit', login_required(edit_profile),
name='edit_profile'),
    path('profile/password', login_required(edit_password),
name='edit_password'),
    path('registration', registration, name='registration'),
    path('catalog', catalog, name='catalog'),
    path('cart', login_required(cart), name='cart'),
    path('feedback', feedback, name='feedback'),
    path('orders', login_required(orders), name='orders'),
]
```

Листинг Б.4 – Файл forms.py

```
from django import forms

class FeedbackForm(forms.Form):
    def __init__(self, *args, **kwargs):
        super(FeedbackForm, self).__init__(*args, **kwargs)

    text = forms.CharField(
        label='Текст',
        widget=forms.TextInput,
        min_length=5,
        max_length=300,
    )
    score = forms.DecimalField(
        label='Оценка (из 10)',
        widget=forms.NumberInput,
        max_digits=2,
        decimal_places=0,
        min_value=1,
        max_value=10,
    )

class OrderForm(forms.Form):
    def __init__(self, *args, **kwargs):
        super(OrderForm, self).__init__(*args, **kwargs)

    phone = forms.DecimalField(
        label='Телефон',
        widget=forms.NumberInput,
        max_digits=11,
        decimal_places=0,
        min_value=0,
        max_value=99999999999,
    )
    address = forms.CharField(
        label='Адрес',
        widget=forms.TextInput,
        max_length=80,
    )

class RegistrationForm(forms.Form):
```

```

def __init__(self, *args, **kwargs):
    super(RegistrationForm, self).__init__(*args, **kwargs)

    username = forms.CharField(
        label='Логин',
        widget=forms.TextInput,
        max_length=30
    )
    first_name = forms.CharField(
        label='Имя',
        widget=forms.TextInput,
        max_length=30
    )
    last_name = forms.CharField(
        label='Фамилия',
        widget=forms.TextInput,
        max_length=30
    )
    email = forms.CharField(
        label='Эл. почта',
        widget=forms.TextInput
    )
    password = forms.CharField(
        label='Пароль',
        widget=forms.PasswordInput,
        max_length=128
    )
    password2 = forms.CharField(
        label='Повторите пароль',
        widget=forms.PasswordInput,
        max_length=128
    )

class PasswordForm(forms.Form):
    def __init__(self, *args, **kwargs):
        super(PasswordForm, self).__init__(*args, **kwargs)

        old_password = forms.CharField(
            label='Старый пароль',
            widget=forms.PasswordInput,
            max_length=128
        )
        new_password = forms.CharField(
            label='Новый пароль',
            widget=forms.PasswordInput,
            max_length=128
        )
        new_password2 = forms.CharField(
            label='Повторите пароль',
            widget=forms.PasswordInput,
            max_length=128
        )

class ProfileForm(forms.Form):
    def __init__(self, *args, **kwargs):
        super(ProfileForm, self).__init__(*args, **kwargs)

```



```

first_name = forms.CharField(
    label='Имя',
    widget=forms.TextInput,
    max_length=30
)
last_name = forms.CharField(
    label='Фамилия',
    widget=forms.TextInput,
    max_length=30
)
email = forms.CharField(
    label='Эл. почта',
    widget=forms.TextInput
)
phone = forms.DecimalField(
    label='Телефон',
    widget=forms.NumberInput,
    max_digits=11,
    decimal_places=0,
    min_value=0,
    max_value=999999999999,
    required=False
)
address = forms.CharField(
    label='Адрес',
    widget=forms.TextInput,
    max_length=80,
    required=False
)

```

Листинг Б.5 – Файл bootstrap_alerts.py

```

from django import template

register = template.Library()

def get_bootstrap_alert(tags):
    return 'danger' if tags == 'error' else tags

register.simple_tag(get_bootstrap_alert, name='get_bootstrap_alert')

```

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

ДНЕВНИК ПРАКТИКИ

за период с 02.02.2020 по 02.07.2020

Студент	<u>Махоткин А.П.</u> (Фамилия, И.О.)
Факультет	<u>СПО</u>
Группа	<u>Y2335</u>
Направление (специальность)	<u>09.02.07 Информационные системы и программирование</u>
Место прохождения практики	<u>Факультет СПО</u>
Руководитель практики	<u>Ефимова Т.Н., факультет СПО, преподаватель</u> <u>Говоров А.И., факультет СПО, преподаватель</u>
Ответственный за проведение практики от университета	<u>Королев В.В., зам. директора факультета</u>

Индивидуальное задание
выполнено полностью

(подпись ответственного
за проведение практики от уни-
верситета)

(дата)

Санкт-Петербург

2020

Период	Краткое содержание работы	Отметка о выполнении
02.02.2020 – 09.02.2020	Вводный инструктаж. Ознакомление с инструкцией по технике безопасности. Ознакомление с целями и задачами практики.	
09.02.2020 – 09.03.2020	Анализ индивидуального задания. Обследование предметной области согласно индивидуальной теме учебной практики.	
10.03.2020 – 31.03.2020	Описание предметной области. Создание диаграммы классов. Создание таблиц.	
01.04.2020 – 07.04.2020	Создание модели Django в соответствии с моделью данных и настройка связи между таблицами	
08.04.2020 – 21.04.2020	Реализация элементов инфраструктуры Django, в соответствии с архитектурным паттерном Model-View-Template или сокращенно MVT. Реализация интерфейсов к системе средствами Django Templates или сторонними средствами.	
17.06.2020 – 23.06.2020	Подготовка отчетных материалов по результатам практики.	
24.06.2020 – 02.07.2020	Защита результатов практики.	

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

О Т З Ы В

руководителя учебной практики
по профессиональному модулю
ПМ.11 «Разработка, администрирование и защита баз данных»

Обучающийся Махоткин А.П.

(Фамилия, И.О.)

Факультет СПО

Группа У2335

Направление (специальность) 09.02.07 Информационные системы и программирование

Место прохождения практики Факультет СПО

Тема индивидуального задания

Оценка достигнутых результатов

№ п/п	Планируемые результаты обучения (показатели)	Оценка			
		5	4	3	2
1.	Корректность определения структуры базы данных				
2.	Качество реализации компонентов описания модели данных средствами Django ORM				
3.	Качество реализации контроллеров Django				
4.	Качество реализации интерфейсов к системе средствами Django Templates или сторонними средствами.				
Итоговая оценка					

Отмеченные достоинства:

.....

Отмеченные недостатки:

.....

Заключение: Считаю, все задачи, поставленные на практику, выполнены и по результатам практики студент(ка) заслуживает оценки «.....».

Руководитель практики _____

(подпись)

(ФИО)

« ____ » _____ 2020г.

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АТТЕСТАЦИОННЫЙ ЛИСТ
Характеристика профессиональной деятельности студента во время прохождения учебной практики

Студент ____ Махоткин А.П. _____

Группа __Y2335__

Специальность 09.02.07 Информационные системы и программирование

Место проведения практики _____

Сроки прохождения практики 20.01.2020 – 02.07.2020

Наименование профессиональных модулей (видов деятельности)

ПМ.11 «Разработка, администрирование и защита баз данных»

Виды выполняемых работ:

Результаты (освоенные профессиональные компетенции)	Основные показатели оценки результата	Отметка о выполнении ¹
ПМ.11 Разработка, администрирование и защита баз данных		
ПК 11.1. Осуществлять сбор, обработку и анализ информации для проектирования баз данных	— осуществление корректного сбора, обработки и анализа информации для проектирования баз данных	
	— обоснование выявления объектов проектируемой БД и установки отношений между ними на основе анализа предметной области;	
ПК 11.2. Проектировать базу данных на основе анализа предметной области.	— использование CASE-средств автоматизированного проектирования при моделировании базы данных при построении концептуальной, даталогической и физической моделей БД;	
	— соответствие проекта структурной и манипуляционной частей БД заданным критериям функциональности.	
ПК 11.3. Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.	использование CASE-средств автоматизированного проектирования при моделировании базы данных при построении концептуальной, даталогической и физической моделей БД;	

¹ Указывается «+» или «-». Считается, что программа практики выполнена, если студентом выполнено не менее 70% перечисленных видов работ.

Результаты (освоенные профессиональные компетенции)	Основные показатели оценки результата	Отметка о выполнении ¹
	обоснование выбранных методов защиты объектов базы данных в соответствии с требованиями задачи.	
ПК 11.4. Реализовывать базу данных в конкретной системе управления базами данных.	соответствие реализации структурной части БД средствами СУБД даталогической и физической моделям данных;	
	— соответствие реализации манипуляционной части БД средствами СУБД заданным критериям;	
	— соответствие реализации приложения БД заданным критериям функциональности;	
ПК 11.5. Администрировать базу данных.	— обоснованность выбора архитектуры клиент-серверного взаимодействия в соответствии с технологией разработки базы данных;	
	соответствие заданным требованиям обеспечения целостности данных и контроля доступа к данным;	
	— соответствие заданным требованиям управления привилегиями пользователей базы данных программными средствами;	
	— соответствие конфигурирования сетевых устройств требованиям обеспечения доступа к данным.	
ПК 11.6. Защищать информацию в базе данных с использованием технологии защиты информации.	соответствие конфигурирования сетевых устройств требованиям защиты данных при передаче данных по сети	
	соответствие заданным требованиям программных средств защиты информации в базе данных средствами СУБД	
	— соответствие заданным требованиям управления привилегиями пользователей базы данных программными средствами;	

Руководитель практики от факультета СПО: _____

Дата: _____