**Министерство транспорта Российской Федерации**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«РУТ (МИИТ)»**

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

КУРСОВОЙ ПРОЕКТ

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему  
«Кондитерская»**

Выполнил: ст. гр. ТКИ-541  
Анисимов Д.М.  
Проверил: Васильева М.А.

Москва 2022

**Цель работы:**

Научится использовать структуру ОРМ на любом языке программирования.

**Постановка задачи:**

Написать на языке Python с использованием фреймворка Django базу данных производства кондитерских изделий (кулинария) . Должна быть возможность хранить рецептуры, запасы ингредиентов, запасы и поставщиков.

**Выполнение работы:**

**models.py**

from django.db import models  
  
class Suplier(models.Model):  
 company\_name = models.CharField(max\_length=255, verbose\_name=**"Поставщик"**, unique=True)  
  
 def \_\_str\_\_(self):  
 return self.company\_name  
  
  
class Ingredients(models.Model):  
 name = models.CharField(max\_length=255, unique=True)  
 count = models.IntegerField()  
 delivery = models.ForeignKey(Suplier, on\_delete=models.CASCADE)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Store(models.Model):  
 store\_name = models.CharField(max\_length=255, verbose\_name=**"Название магазина"**)  
 address = models.CharField(max\_length=255, verbose\_name=**"Адрес магазина"**)  
  
 def \_\_str\_\_(self):  
 return self.store\_name  
  
  
class Menu(models.Model):  
 name = models.CharField(max\_length=255, unique=True)  
 ingredients = models.ManyToManyField(Ingredients, verbose\_name=**"Необходимые ингредиенты для выпечки"**)  
 transfer\_to = models.ManyToManyField(Store)  
 cost = models.IntegerField(verbose\_name=**"Цена"**)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Produced(models.Model):  
 menu = models.ForeignKey(Menu, on\_delete=models.CASCADE)  
 count = models.IntegerField(verbose\_name=**"Количество"**)  
 date = models.DateField(verbose\_name=**"Дата выпечки"**)  
  
 def \_\_str\_\_(self):  
 return self.menu.name  
  
  
class Purchase(models.Model):  
 Sell = models.ForeignKey(Menu, to\_field=**"name"**, verbose\_name=**"Prodano"**, on\_delete=models.CASCADE)  
 count = models.IntegerField()  
 date = models.DateField()  
  
 def \_\_str\_\_(self):  
 return **f"**{self.Sell} {self.count} {self.date}**"**

**Описание классов:**

***Suplier***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *company\_name* | *CharField(255)* | *Да* |

***Ingredients***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *name* | *CharField(255)* | *Да* |
| *count* | *IntegerField* | *Да* |
| *delivery* | *CharField(255)* | *Да* |

***Store***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *store\_name* | *CharField(255)* | *Да* |
| *address* | *CharField(255)* | *Да* |

***Menu***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *name* | *CharField(255)* | *Да* |
| *ingredients* | *ManyToManyField* | *Да* |
| *transfer\_to* | *ManyToManyField* | *Да* |
| *cost* | *IntegerField* | *Да* |

***Produced***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *name* | *ForeignKey* | *Да* |
| *count* | *IntegerField* | *Да* |
| *delivery* | *DateField* | *Да* |

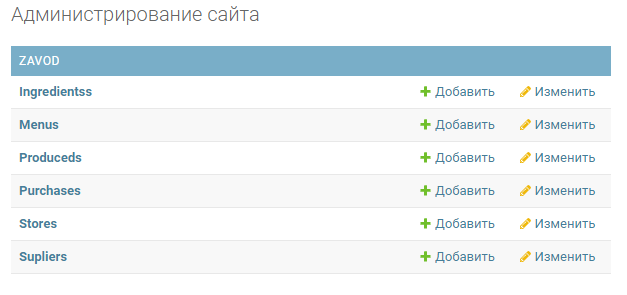
***Purchase***

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Обязательное* |
| *sell* | *ForeignKey* | *Да* |
| *count* | *IntegerField* | *Да* |
| *date* | *DateField* | *Да* |

**Скриншоты работы**

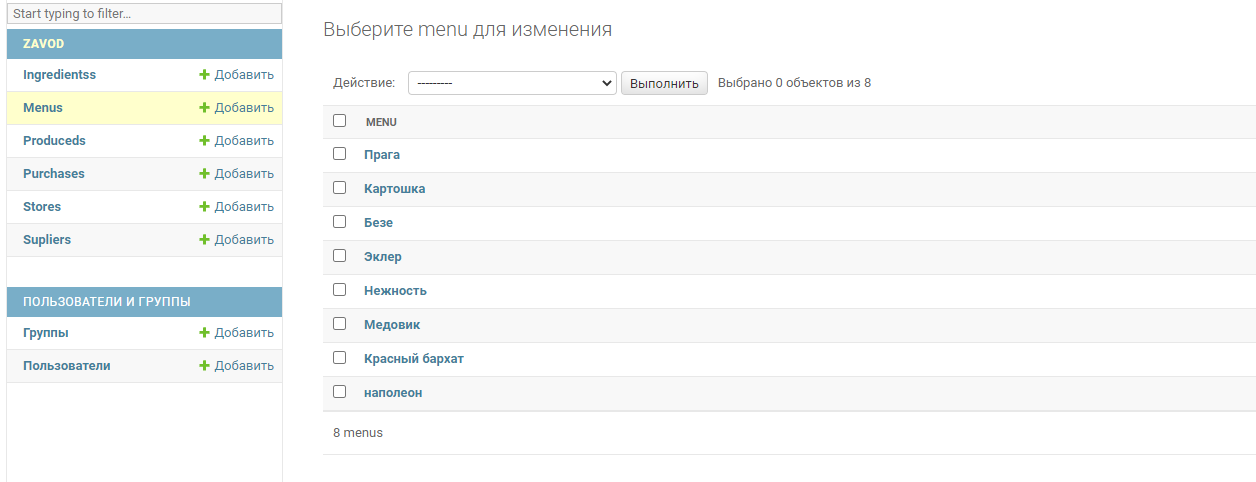
После создания классов в администрировании Django стало доступно изменение и добавления элементов в базу данных.

**Изменение данных**

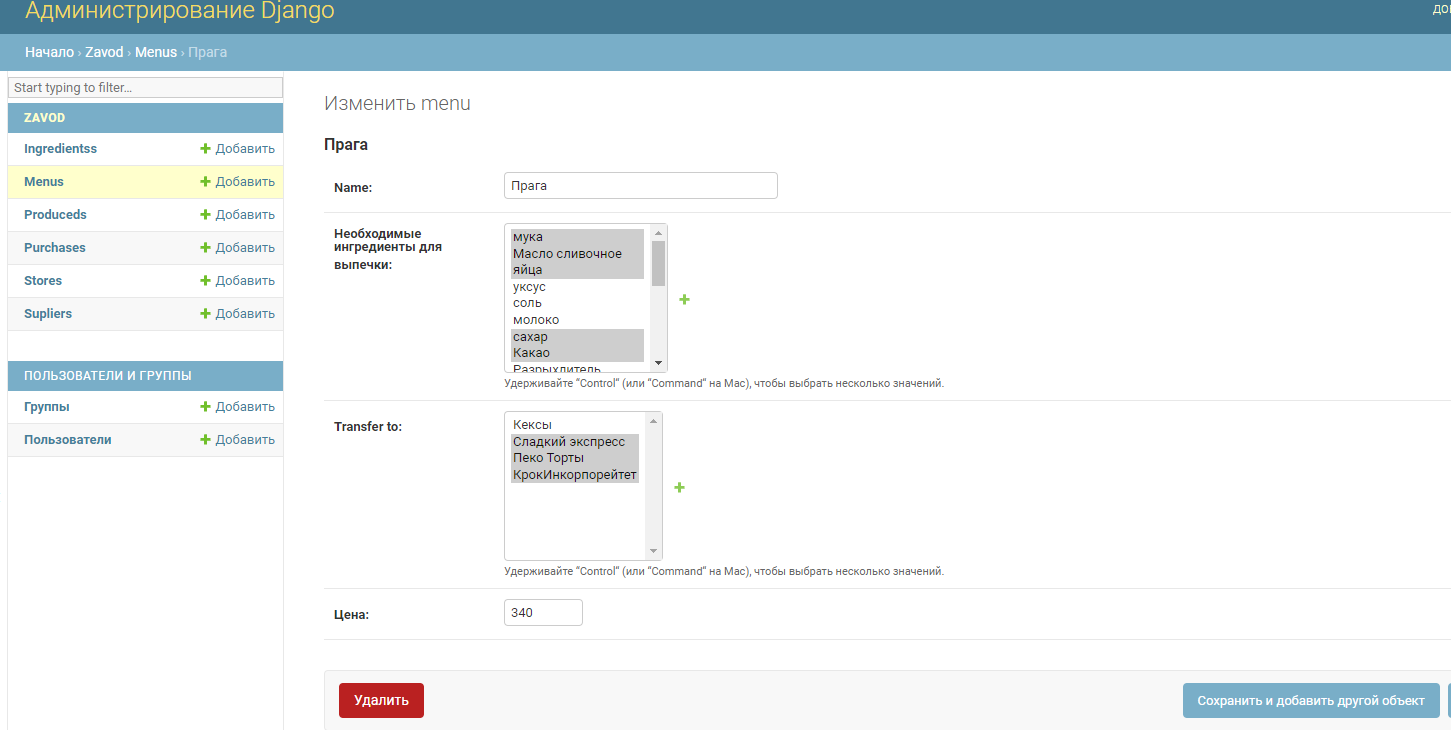
****

*рисунок 1 страница администратора*

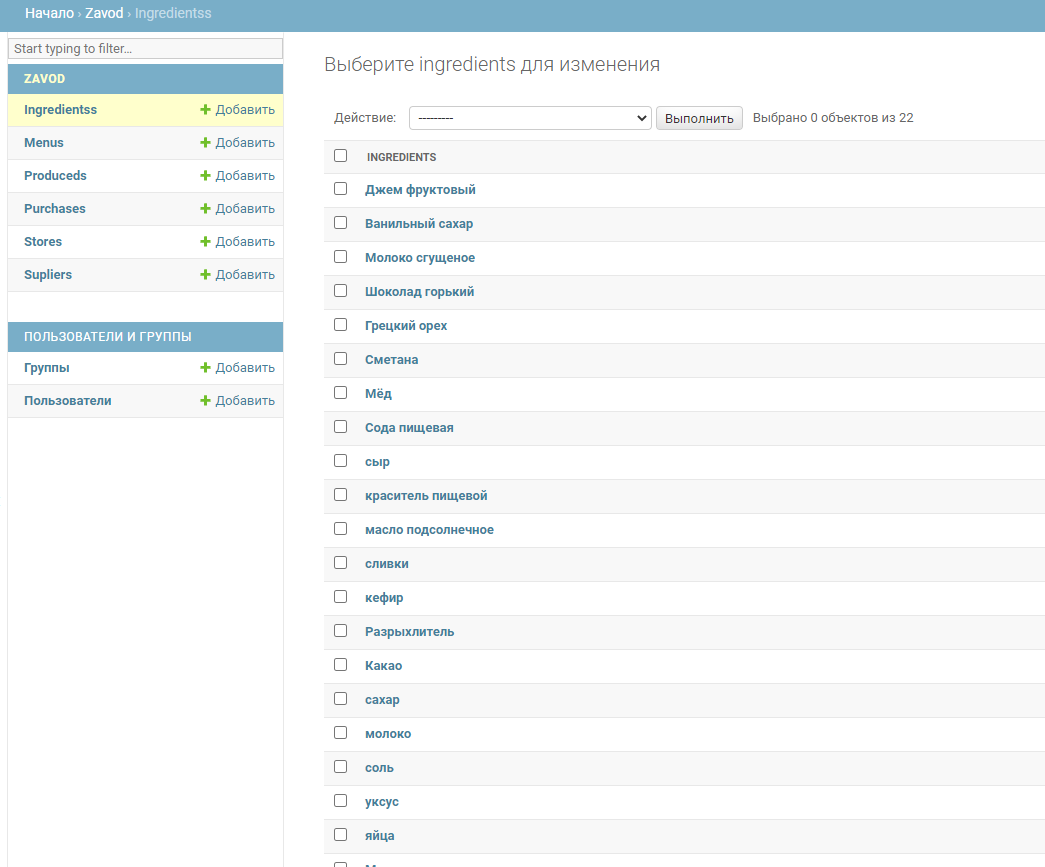
**Изменение меню**

**

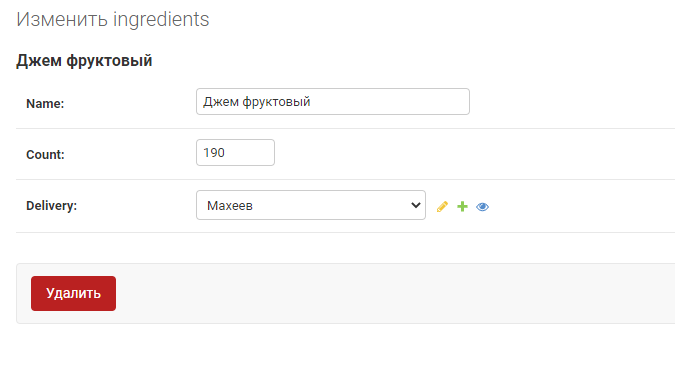
*рисунок 2 Страница изменения меню*

**

*рисунок 3 страница изменения меню*

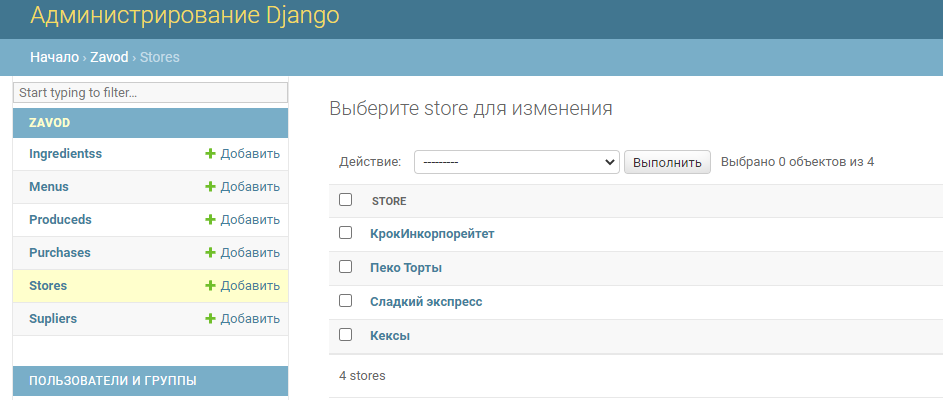
**Изменение серий****

*рисунок 4 страница изменения ингредиентов*

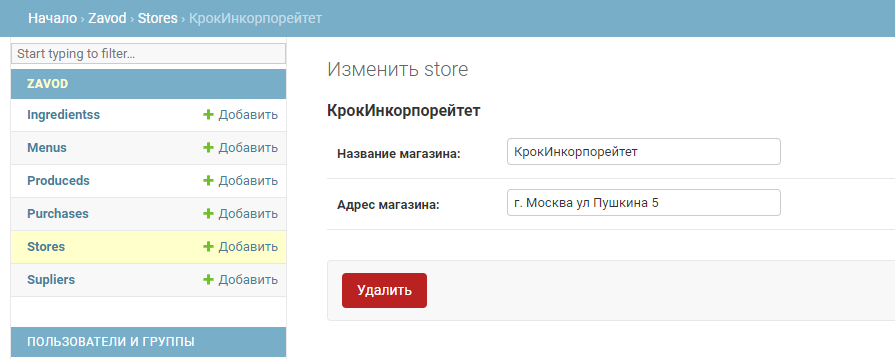
**

*рисунок 5 страница изменения ингредиентов*

**Изменение магазинов**

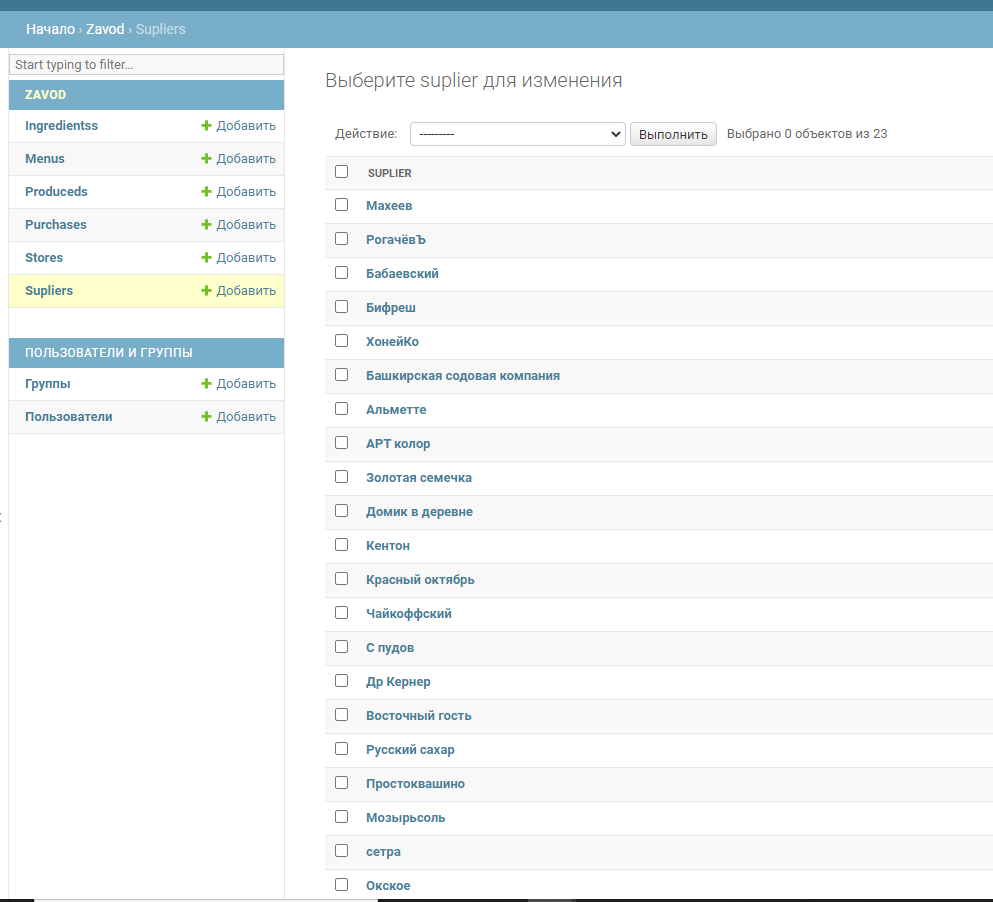
****

*рисунок 6 страница изменения магазинов*

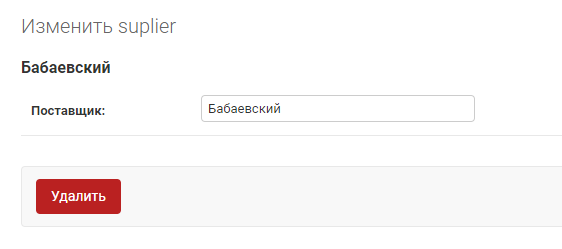
**

*рисунок7 страница изменения магазинов*

**Изменение поставщиков**

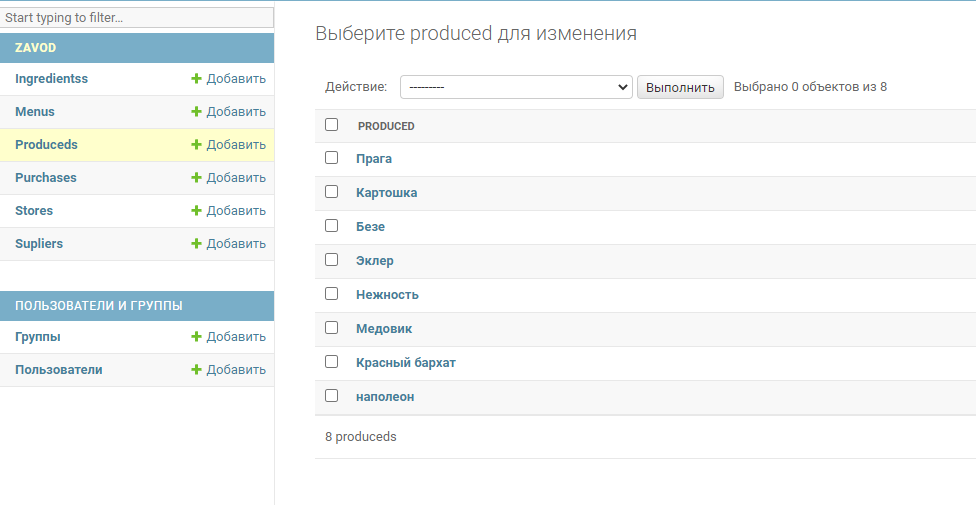
****

*рисунок 8 страница изменения поставщиков*

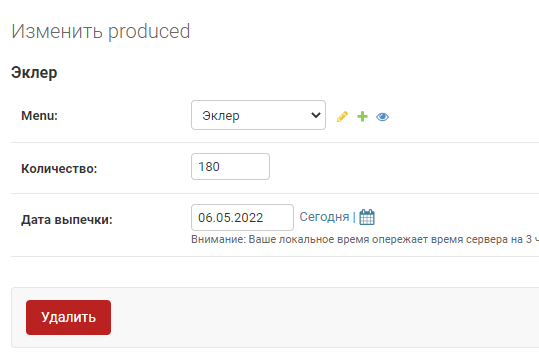
**

*рисунок 8 страница изменения поставщиков*

**Изменение производства**

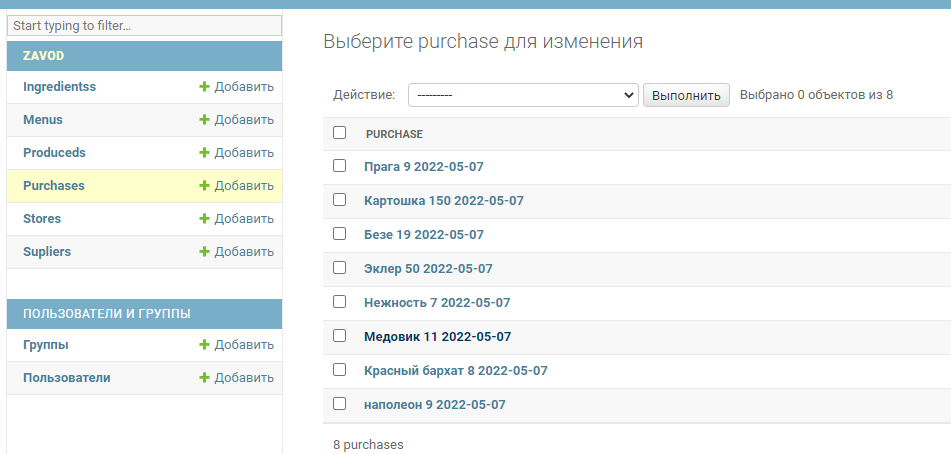
****

*рисунок 9 страница изменения производств*

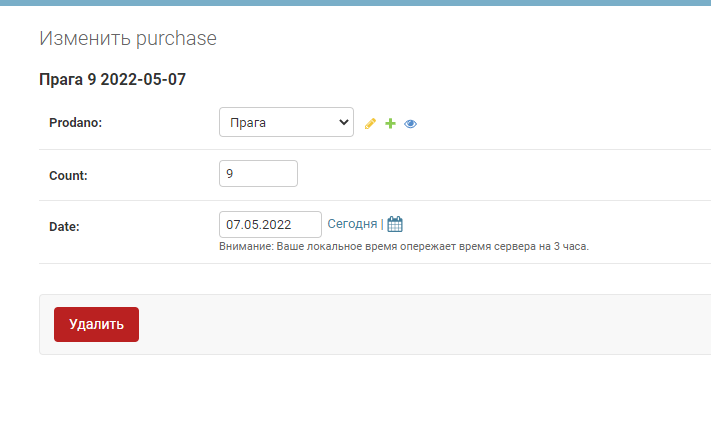
**

*рисунок 10 страница изменения производств*

**Изменение продаж**

****

*рисунок 11 страница изменения продаж*

**

*рисунок 12 страница изменения продаж*

**Вывод:**

Была произведена работа по созданию базы данных с помощью технологии ОРМ Python Django