МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики

и вычислительной техники

Кафедра информационной безопасности

Отчёт к курсовому проекту

по дисциплине “Безопасность систем баз данных”

**Разработка базы данных для молочных ферм**

Выполнили: студенты группы БИ-31

Скульдицкий Никита,  
Чекурова Яна  
Круговых Виктор

Матвеев Евгений

Проверил: доцент кафедры

ИБ Сучков Д.С.

Йошкар-Ола

2020 г.

**СОДЕРЖАНИЕ**

[**Введение** 2](#_Toc62060446)

[**1.** **Техническое задание** 3](#_Toc62060447)

[***1.1 Требования к курсовой работе:*** 3](#_Toc62060448)

[***1.2 Требования к базе данных*** 3](#_Toc62060449)

[**1.3 Требование к API (минимальное количество реализованных методов)** 3](#_Toc62060450)

[**2.** **Требования по пунктам** 4](#_Toc62060451)

[**3.** **Порядок выполнения работы** 18](#_Toc62060452)

[**4.** **Структура проекта.** 21](#_Toc62060453)

[**5.** **Структура приложения.** 23](#_Toc62060454)

[**6.** **Вид сайта.** 26](#_Toc62060455)

[**Приложения.** 30](#_Toc62060456)

[**Вывод.** 30](#_Toc62060457)

# **Введение**

В курсовой работе мы рассматриваем создание базы данных, предназначенной для поиска доступных билетов и их покупка. База данных позволяет клиентам просматривать доступные рейсы, так и покупку билетов

## **Техническое задание**

## ***1.1 Требования к курсовой работе:***

* Получить структуру данных из файла, согласно варианта. Привести к 3й нормальной форме. Добавить недостающие таблицы.
* Составить ER-диаграмму
* Разработать API для базы данных на любом языке, выполняющемся на стороне сервера (php, ASP.NET, Java, python, node.js, etc)
* Взаимодействие должно осуществляться по клиент-серверной архитектуре, подключение с клиентской программы недопустимо
* Провести настройку пользователей базы данных для разграничения прав доступа, привести пример конфигурации
* Все документы и исходные коды для курсовой работы должны храниться под контролем системы контроля версий — git или mercurial (https://github.com/, https://bitbucket.org/)
* Во время сдачи курсового проекта необходимо предоставить отчет о проделанной работе в печатном виде (отчет)

## ***1.2 Требования к базе данных***

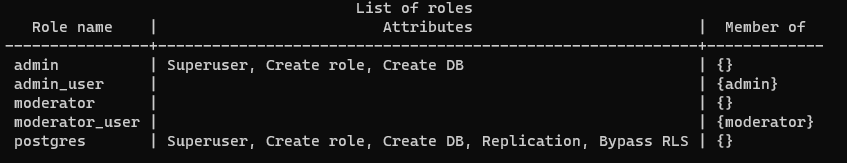
* Наличие не менее 7 таблиц, в том числе таблицы сессий и пользователей
* Структура таблицы должна содержать не менее 3-х полей, одно из которых ключевое
* Правомерное использование типов данных
* Обязательно использование триггеров и/или хранимых процедур
* Форма нормализации не менее 3NF
* Индексирование по полям поиска

## **1.3 Требование к API (минимальное количество реализованных методов)**

* аутентификация пользователя (создание сессии);
* добавление/удаление/изменение данных в таблицах;
* выборка данных их ключевых таблиц по запросам;
* выборка данных из таблиц с объединением результатов.

1. **Требования по пунктам**

1)Есть 2 группы пользователей: admin, moderator.



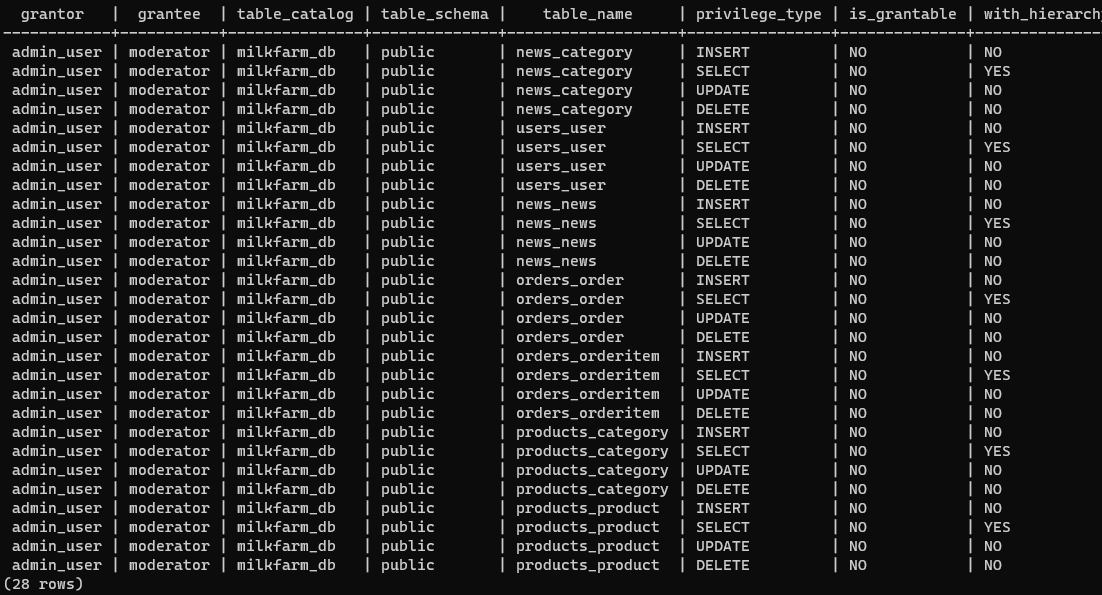
Admin – имеет права изменять все.



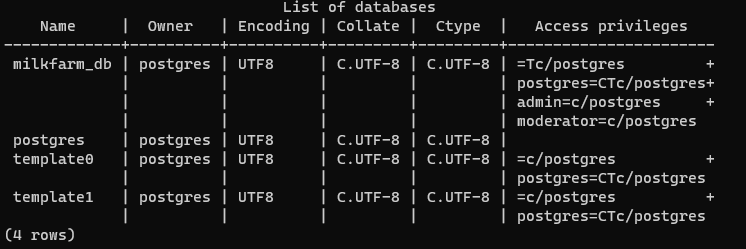
Команда для установки привилегий:



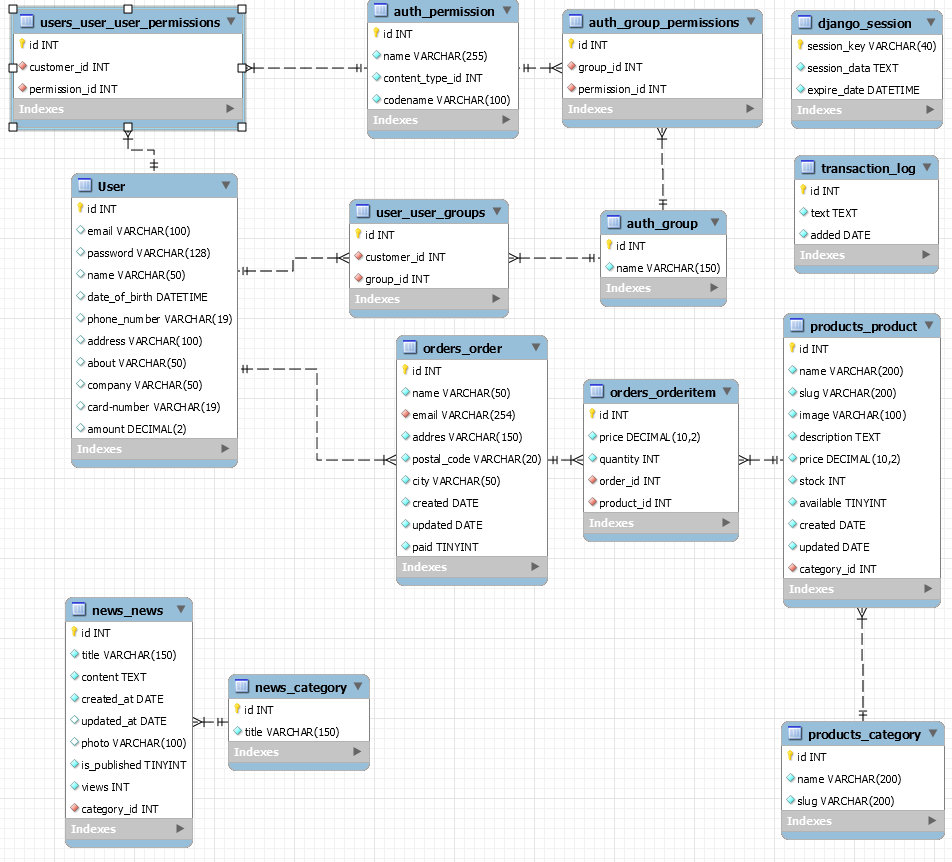
Moderator – может изменять только контент на сайте.



Все роли имеют изначальный доступ к базе данных:

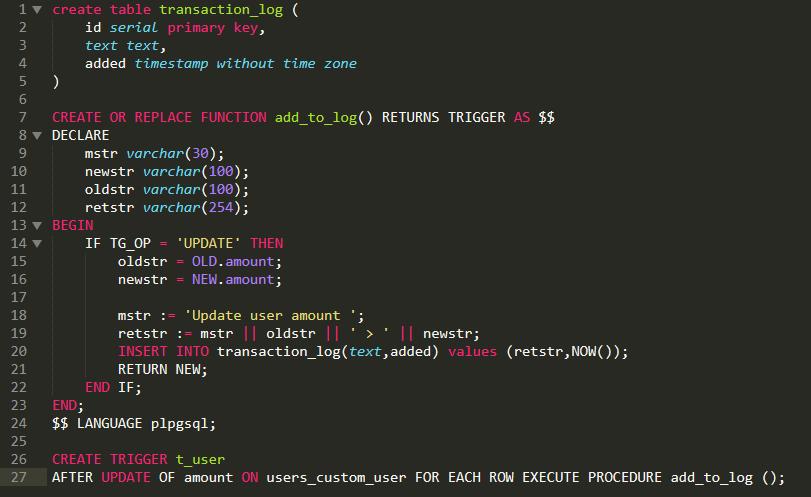


2)



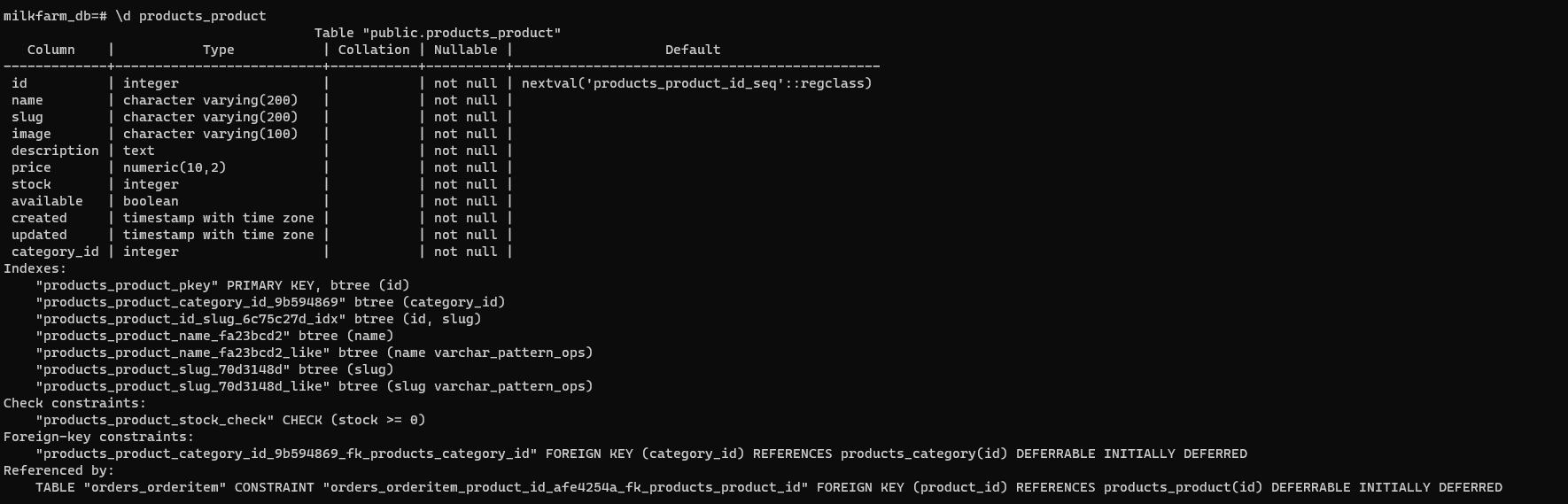
3)Триггер для аудита движения средств на счете клиента.





4)В таблицах присутствуют индексы на ключах и ключевых полях.





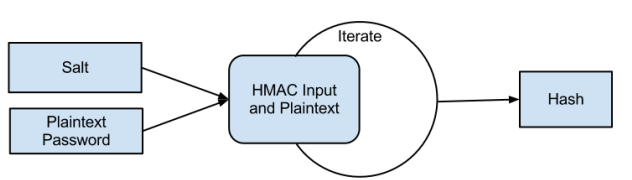




5)Для новых пользователей, которые регистрируются через сайт используется **pbkdf2\_sha256.**

PBKDF2 использует псевдослучайную функцию для получения ключей. Длина генерируемого ключа не ограничивается (хотя эффективная мощность пространства ключей может быть ограничена особенностями применяемой псевдослучайной функции). Использование PBKDF2 рекомендовано для новых программ и продуктов. В качестве псевдослучайной хеш-функции выбрана **sha256**.

При использовании этого алгоритма хэшируется соль и открытый текст для получения первого хэша, затем в цикле тот же алгоритм используется для вычисления хэша от открытого текста и результата предыдущей итерации, после чего возвращается результат применения операции XOR ко всем вычисленным хэшам. Выполнив на серверной стороне данную операцию 100000 или больше раз, вы с генерируете сильный, стойкий ко взлому хэш, который можно безопасно использовать для хранения паролей.

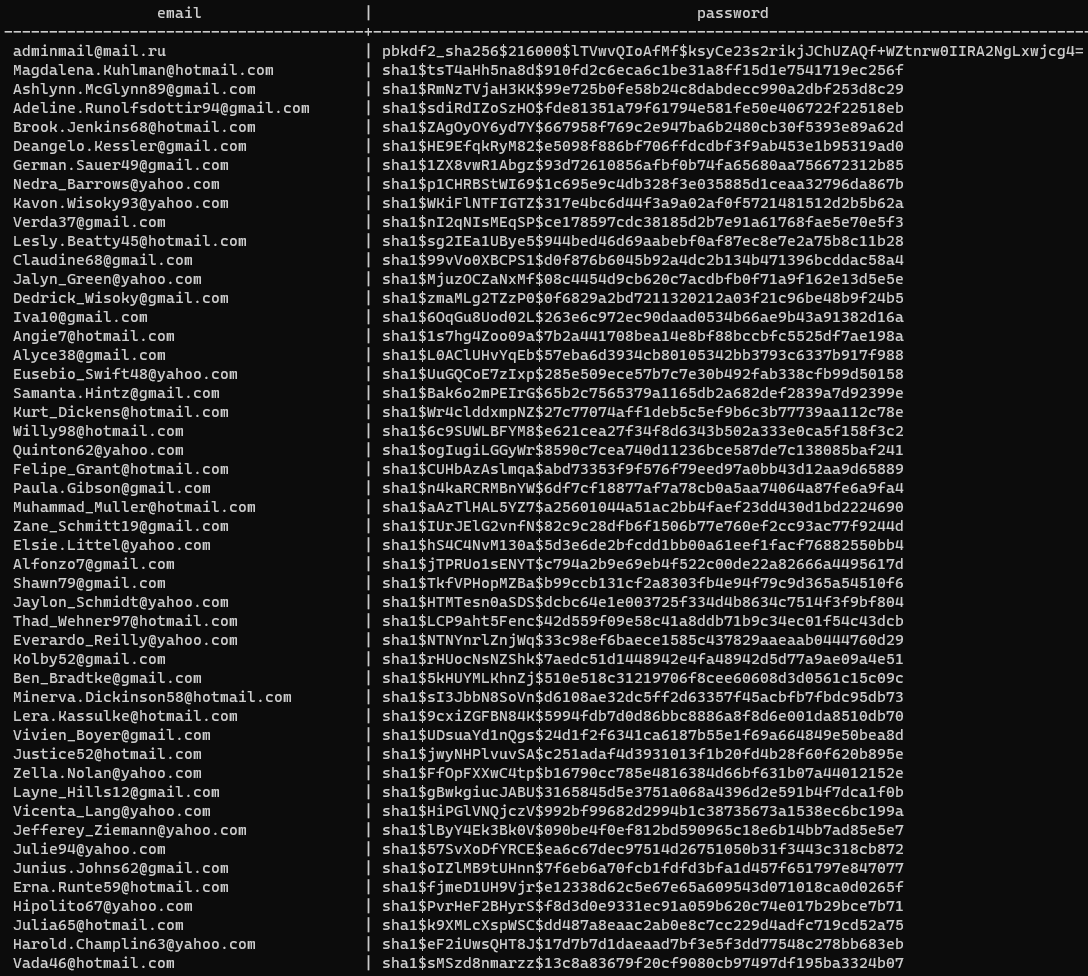


Для уже существующих пользователей был выбран алгоритм **sha1**.

SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента{\displaystyle h\_{i}=f(M\_{i},h\_{i-1})}. Хеш-значением всего сообщения является выход последнего блока.

Вид пароля в базе данных:

<algorithm>$<iterations>$<salt>$<hash>



6)Данные карт шифруются перед вставкой в базу данных.

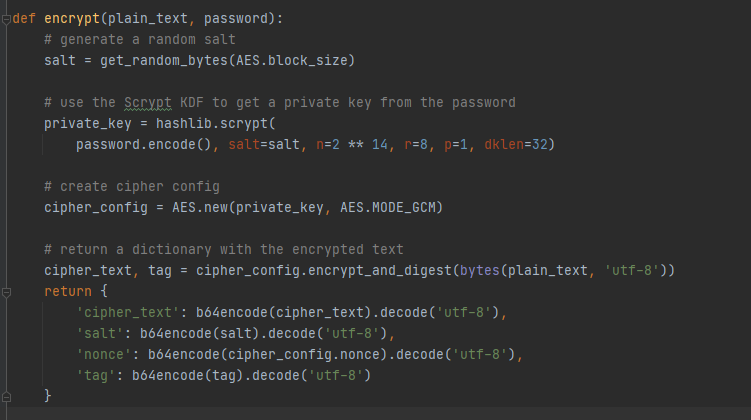
Был выбран AES. является стандартом, основанным на алгоритме Rijndael. Для AES длина input (блока входных данных) и State (состояния) постоянна и равна 128 бит, а длина шифроключа **K** составляет 128, 192, или 256 бит. При этом исходный алгоритм Rijndael допускает длину ключа и размер блока от 128 до 256 бит с шагом в 32 бита. Для обозначения выбранных длин input, State и Cipher Key в 32-битных словах используется нотация Nb = 4 для input State, Nk = 4, 6, 8 для Cipher Key соответственно для разных длин ключей.

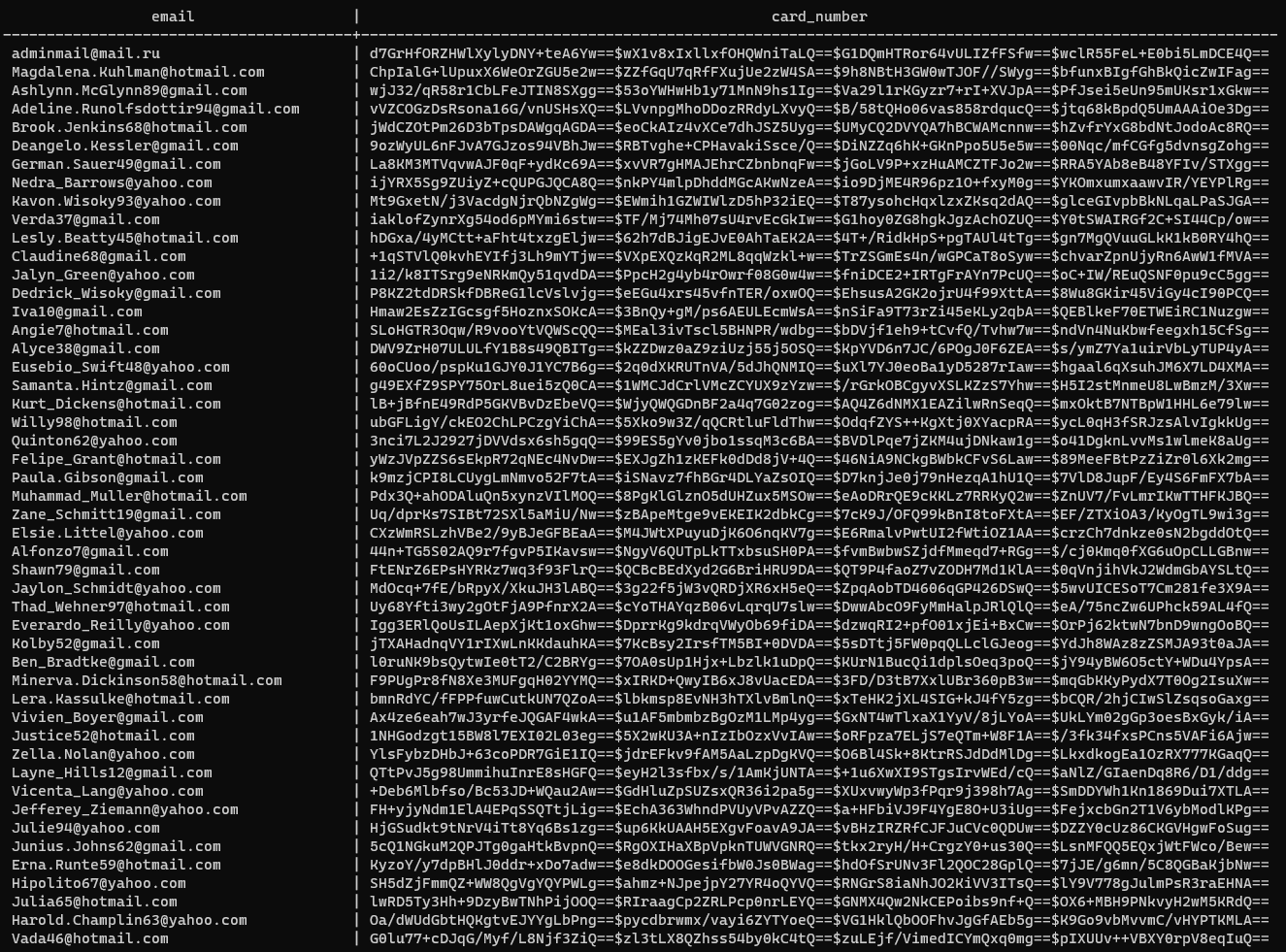
В базе данных он представлен в следующем виде:

<cipher\_text>$<salt>$<nonce>$<tag>

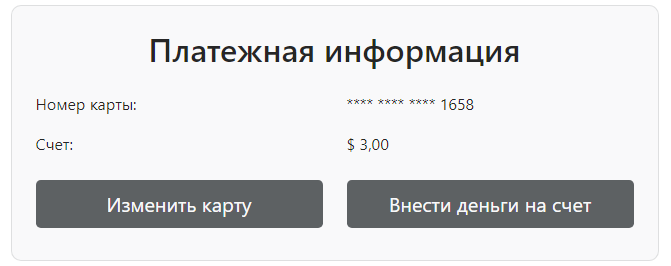
Пример строки:

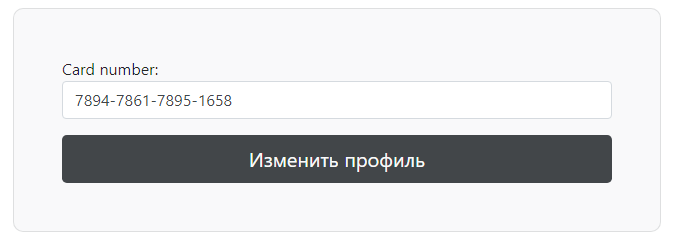
dS6FJ8nBEoiUfHvrE5nj7scSX00Jwu9mQFM=$Lgp17qt8jnmy13GeNhg64Q==$22w+0mvfcsLFrYv1sxTpUw==$X43yj852+brrhQEgV2KyLw==

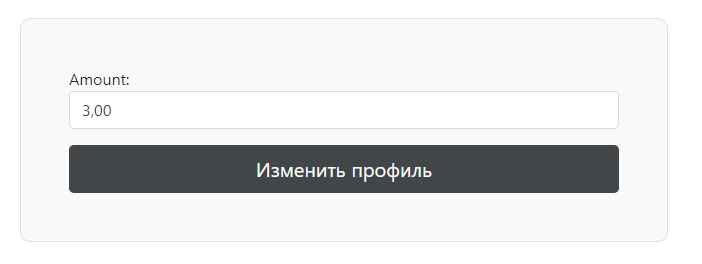


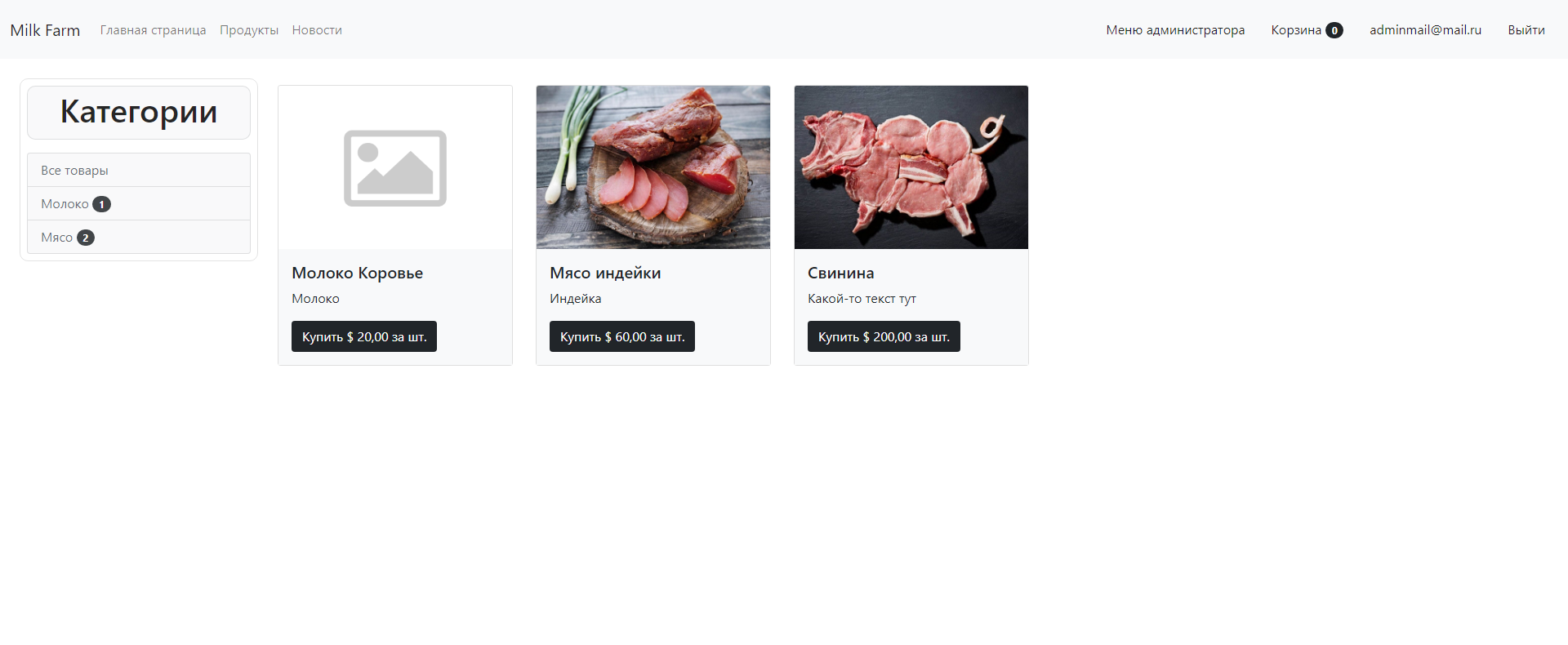


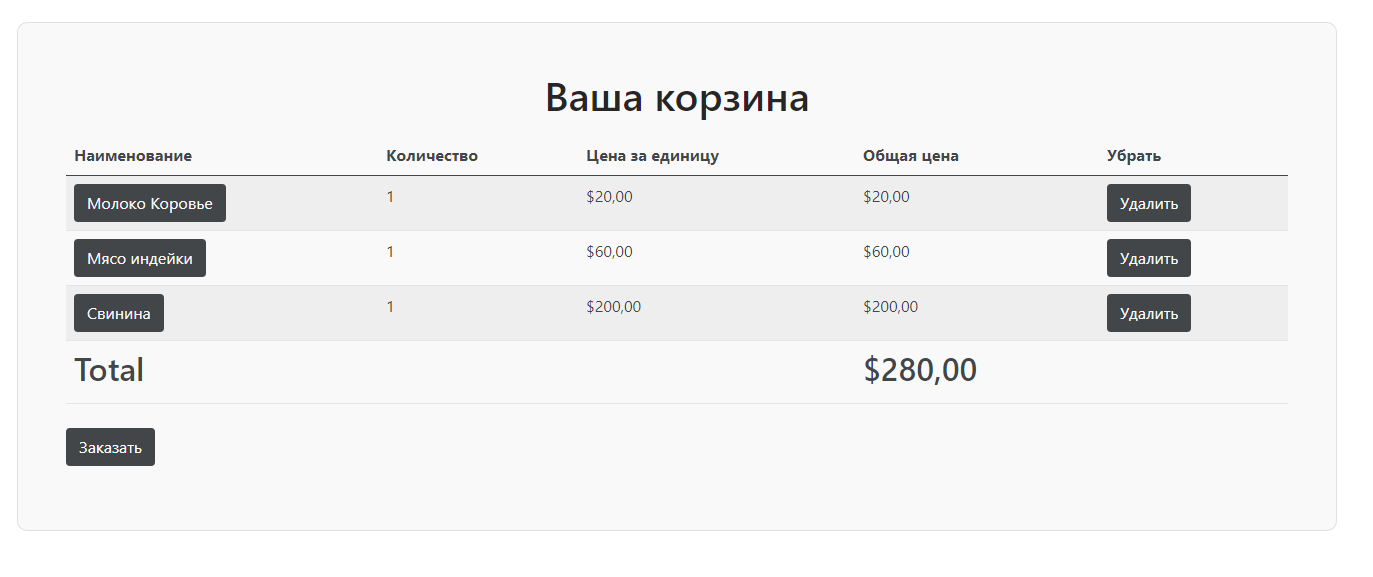
7)

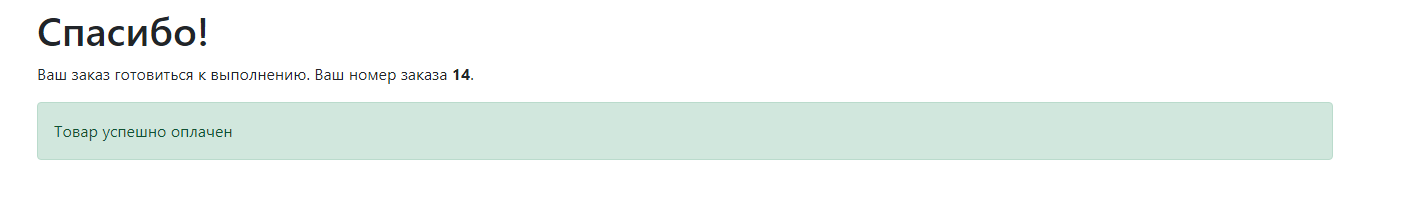








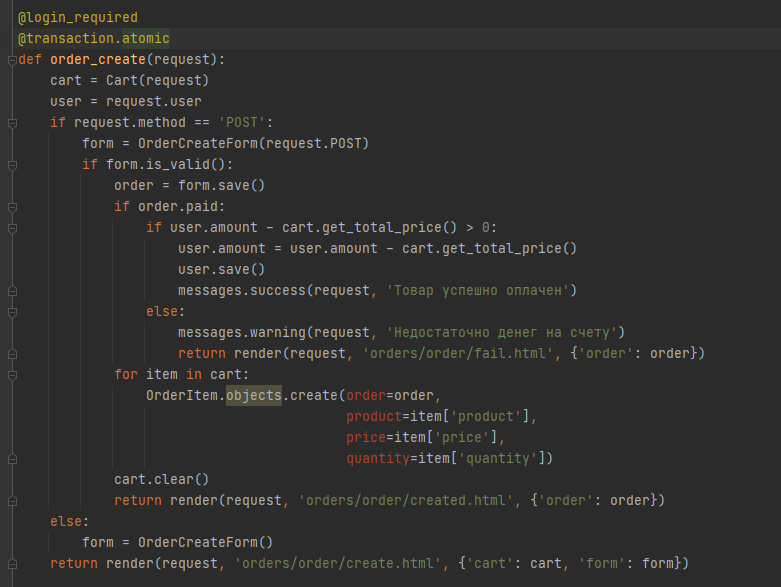




В Django присутствуют атомарные операции, то есть транзакции.

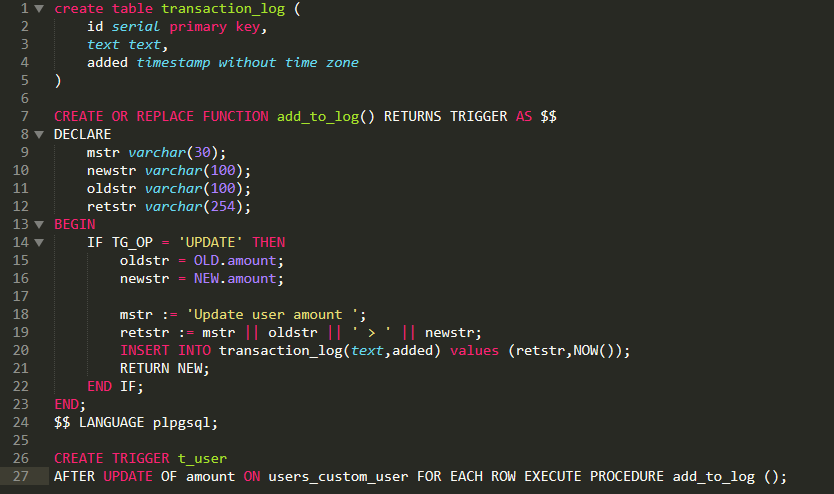
Сделать функцию атомарной можно с помощью декоратора @transaction.atomic.

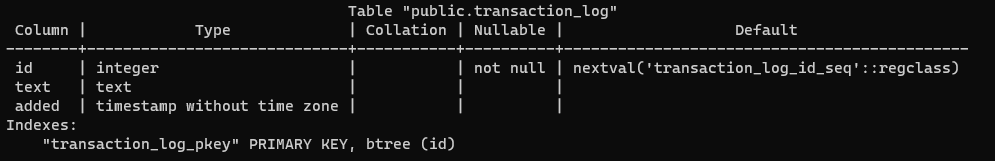
Если возникнет ошибка в процессе выполнения функции, то ничего не запишется в базу данных.

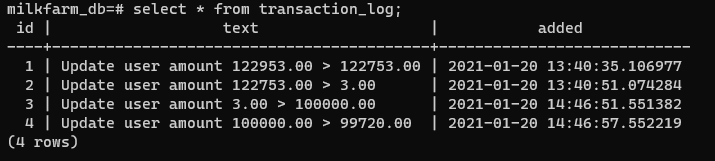


8)Использование триггеров для аудита движения средств на счете клиента.

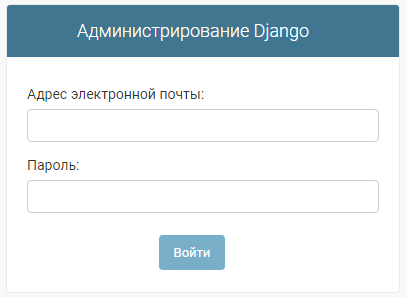
Использование триггера на обновления строки amount. При изменении записывается сумма до и после изменения в отдельную таблицу.

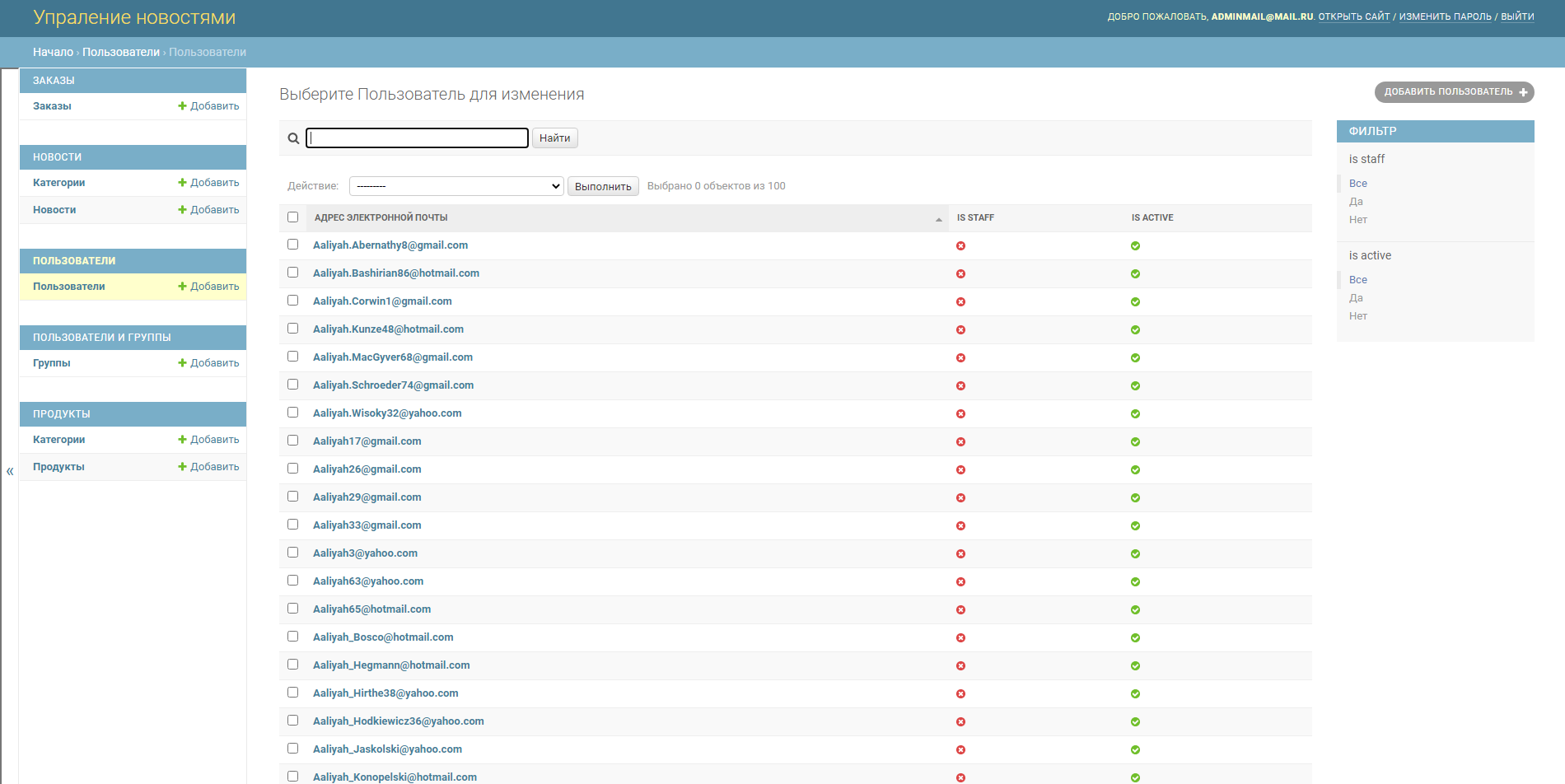


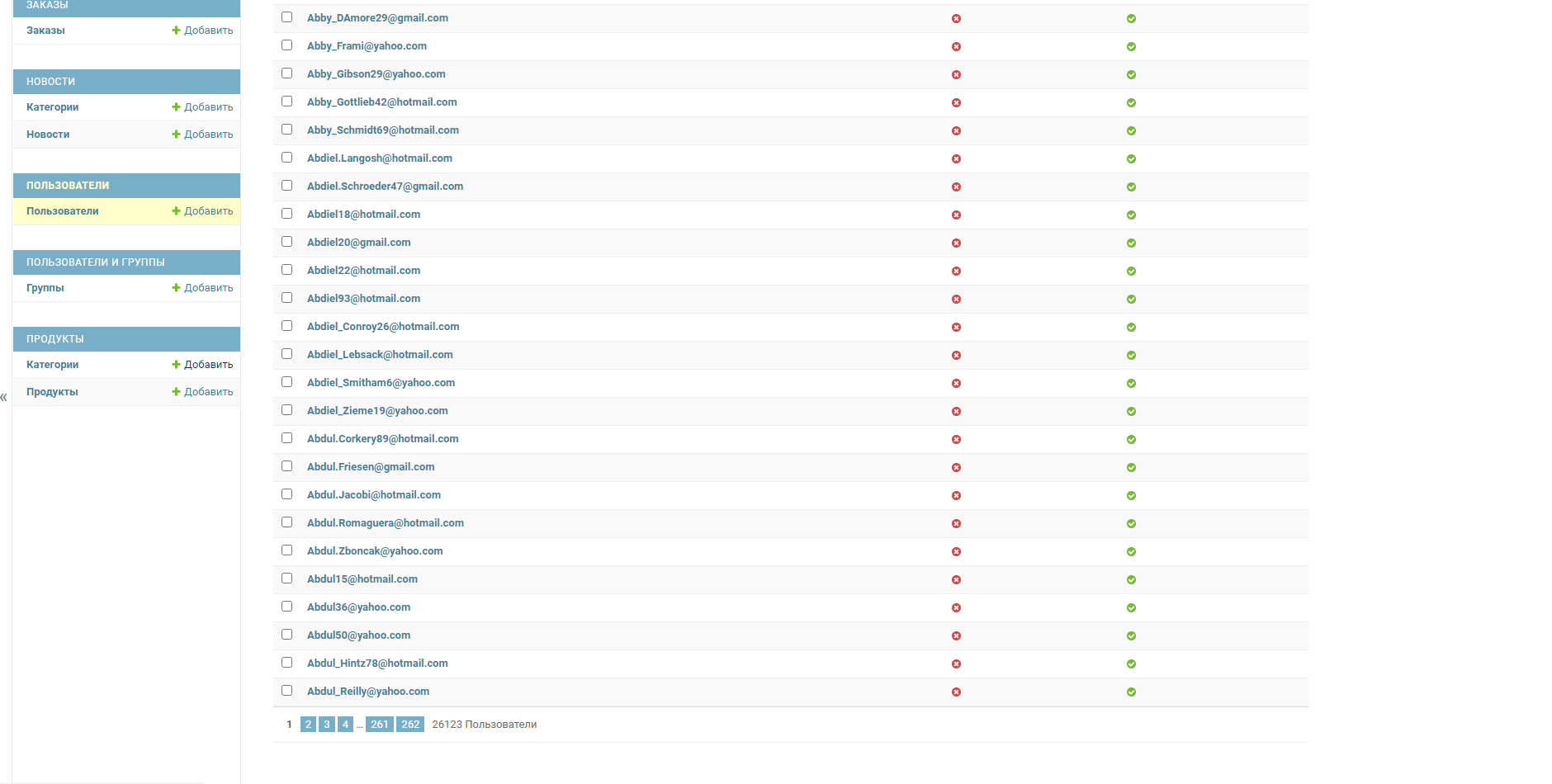


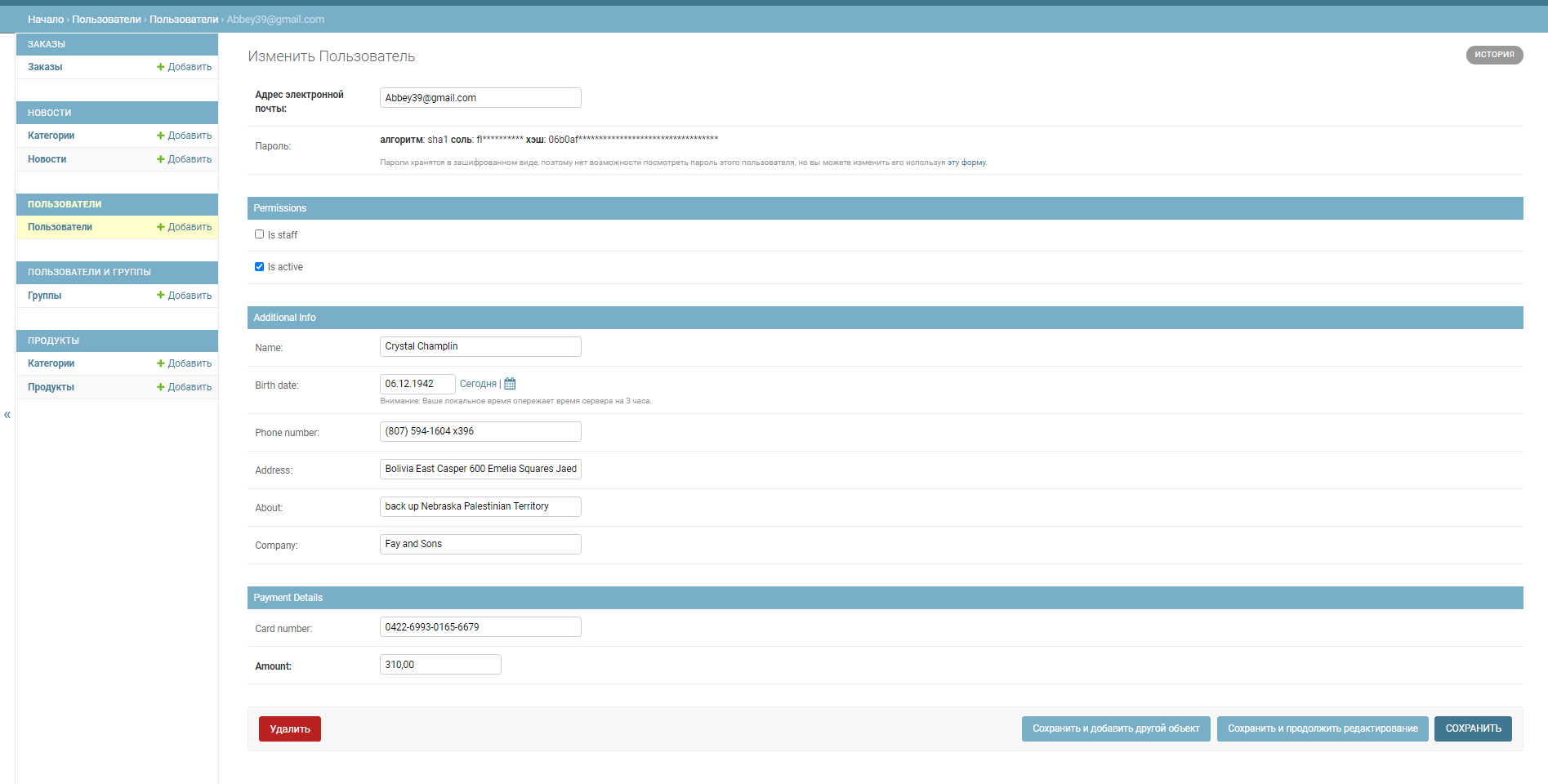


9)







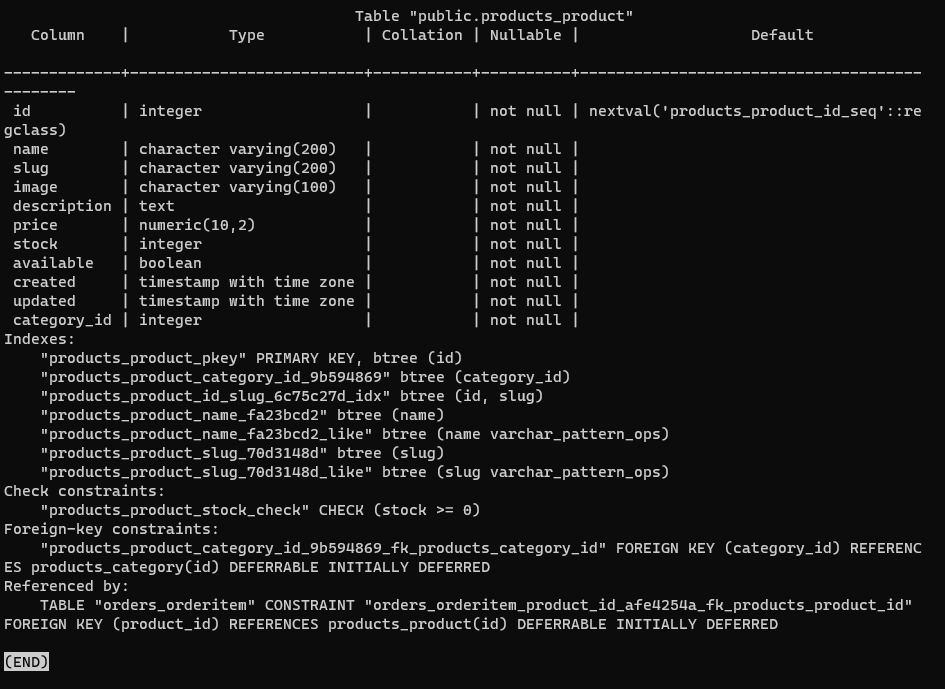


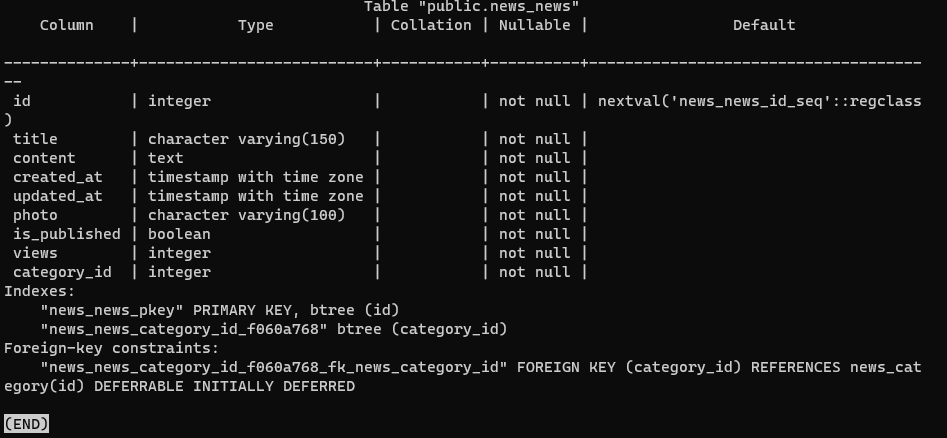
1. **Порядок выполнения работы**

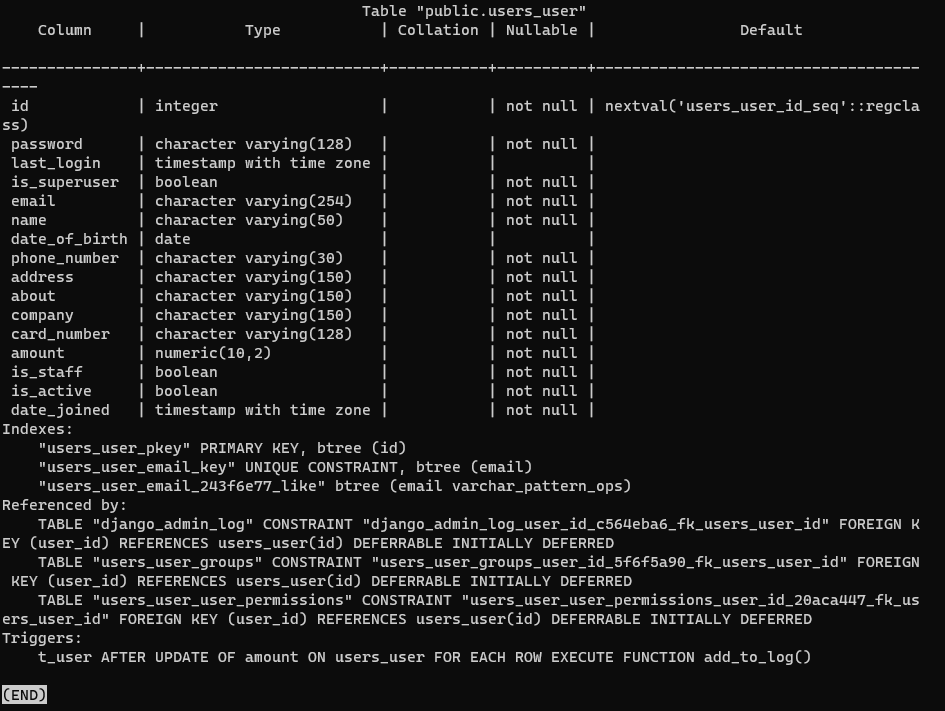
Разработана база данных, содержащая 33 таблиц из них половина служебных. Одни из основных являются:

* Products\_product
* news\_news
* Users \_user

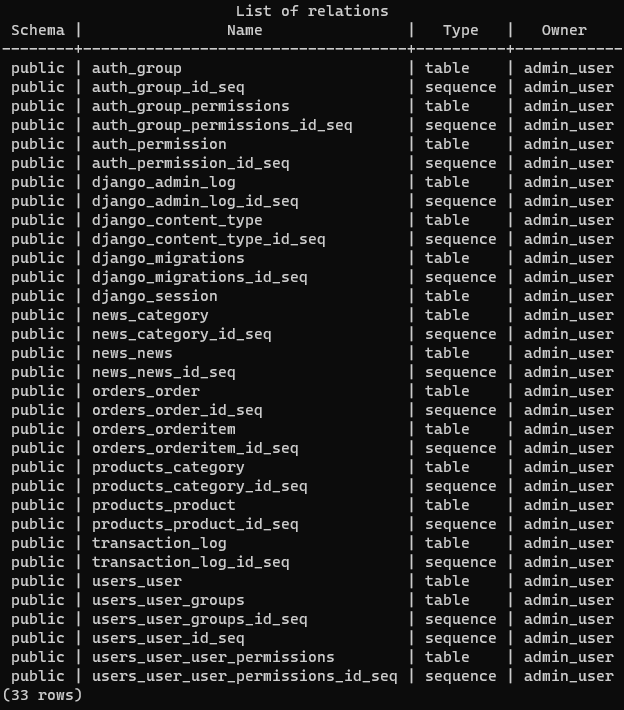
Структура главных таблиц:



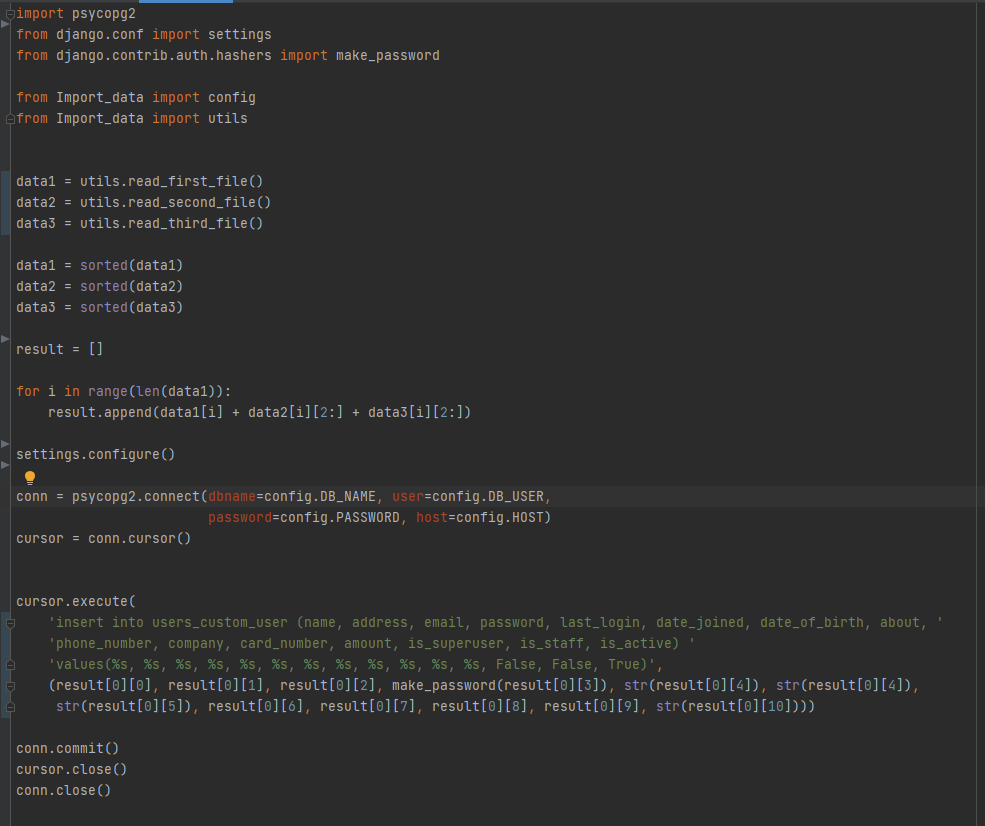
**

**

Полный список таблиц:

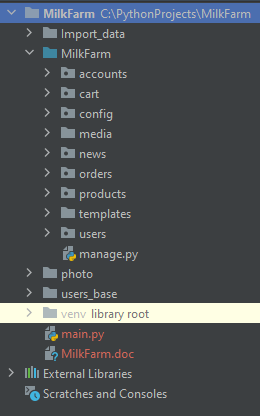


Скрипт для вставки сырых данных из таблиц в бд:



1. **Структура проекта.**

Сама структура:



- Import\_data – вставка сырых данных в таблицу

- MilkFarm – корневой дистрибутив Django

- accounts, cart, orders, products, users, news – Django приложения

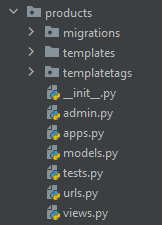
- config – пакет настроек Django

- static – статичные файлы проекта, такие как фото, стили, скрипты

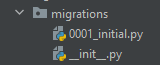
- templates – html шаблоны проекта

1. **Структура приложения.**

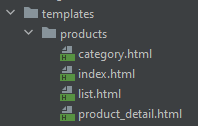
Сама структура:



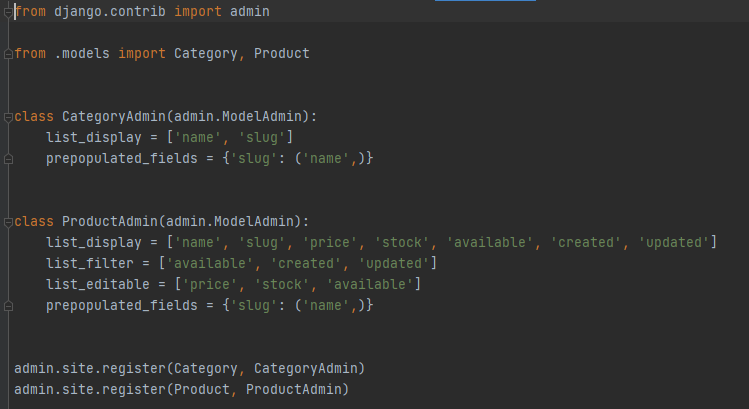
- Migrations – пакет с изменениями в базе данных



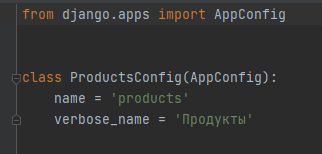
- Templates – html шаблоны приложения, (views)



- Admin.py – файл регистрации моделей в админке и вывод их



- Apps.py - файл регистрации и настройки приложения

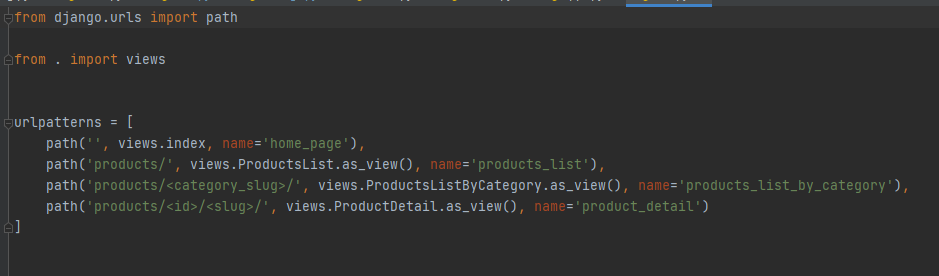


*Рис 14. Forms.py*

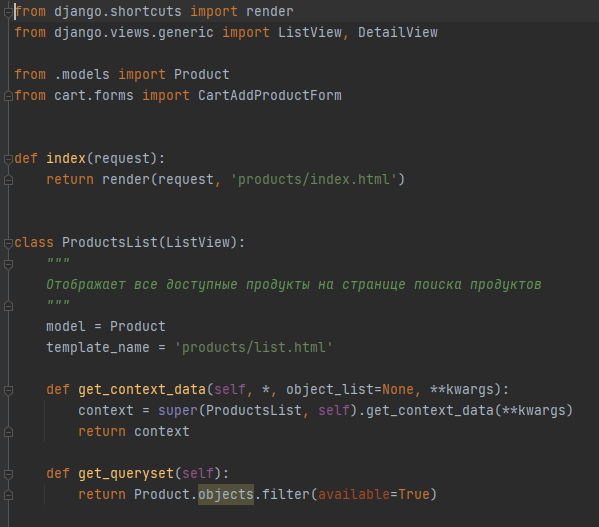
- Models.py – модели, используемые в приложении. Позволяет управлять базой данных (Models)



- Urls.py – пути данного приложения



- Views.py – логика отображения (controller)

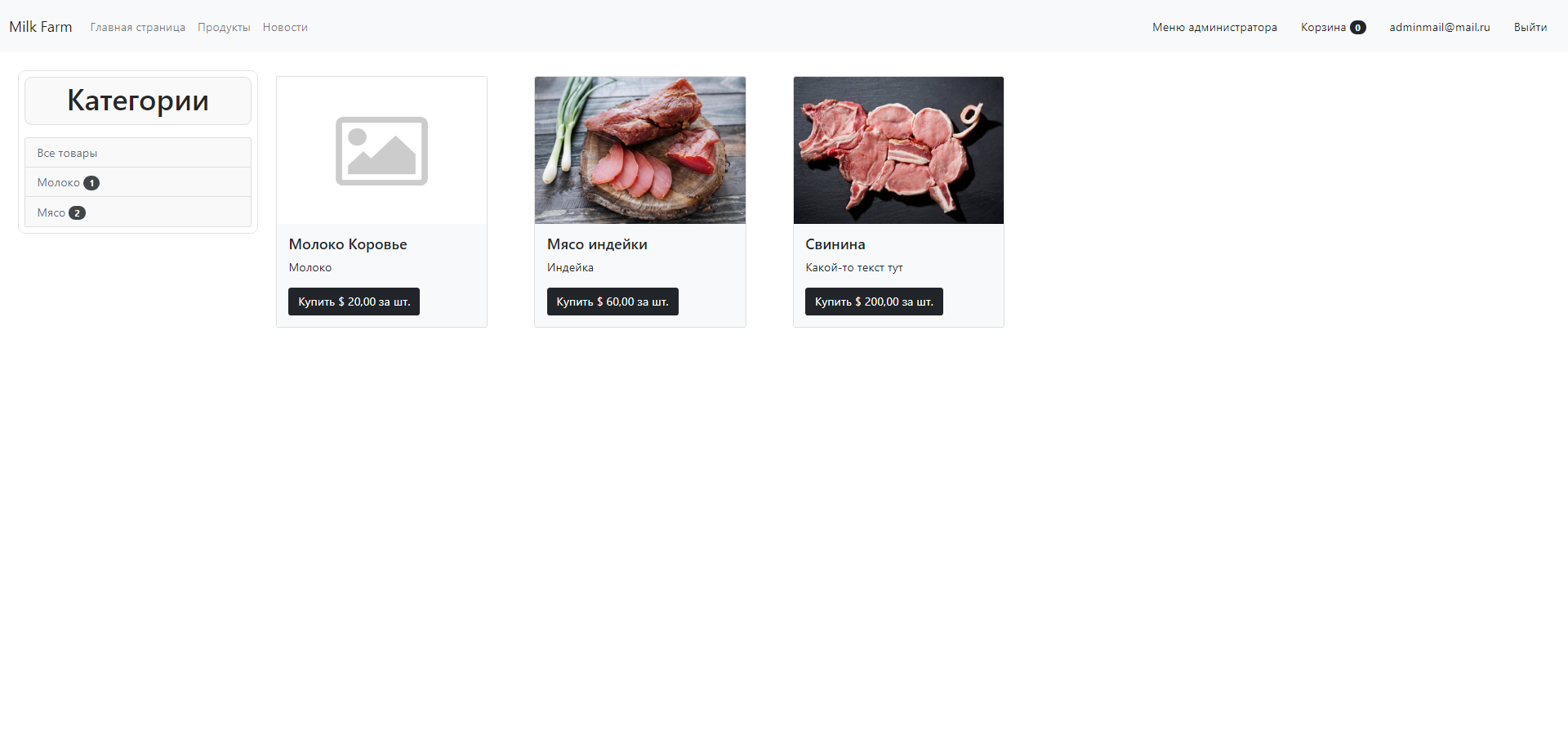


1. **Вид сайта.**

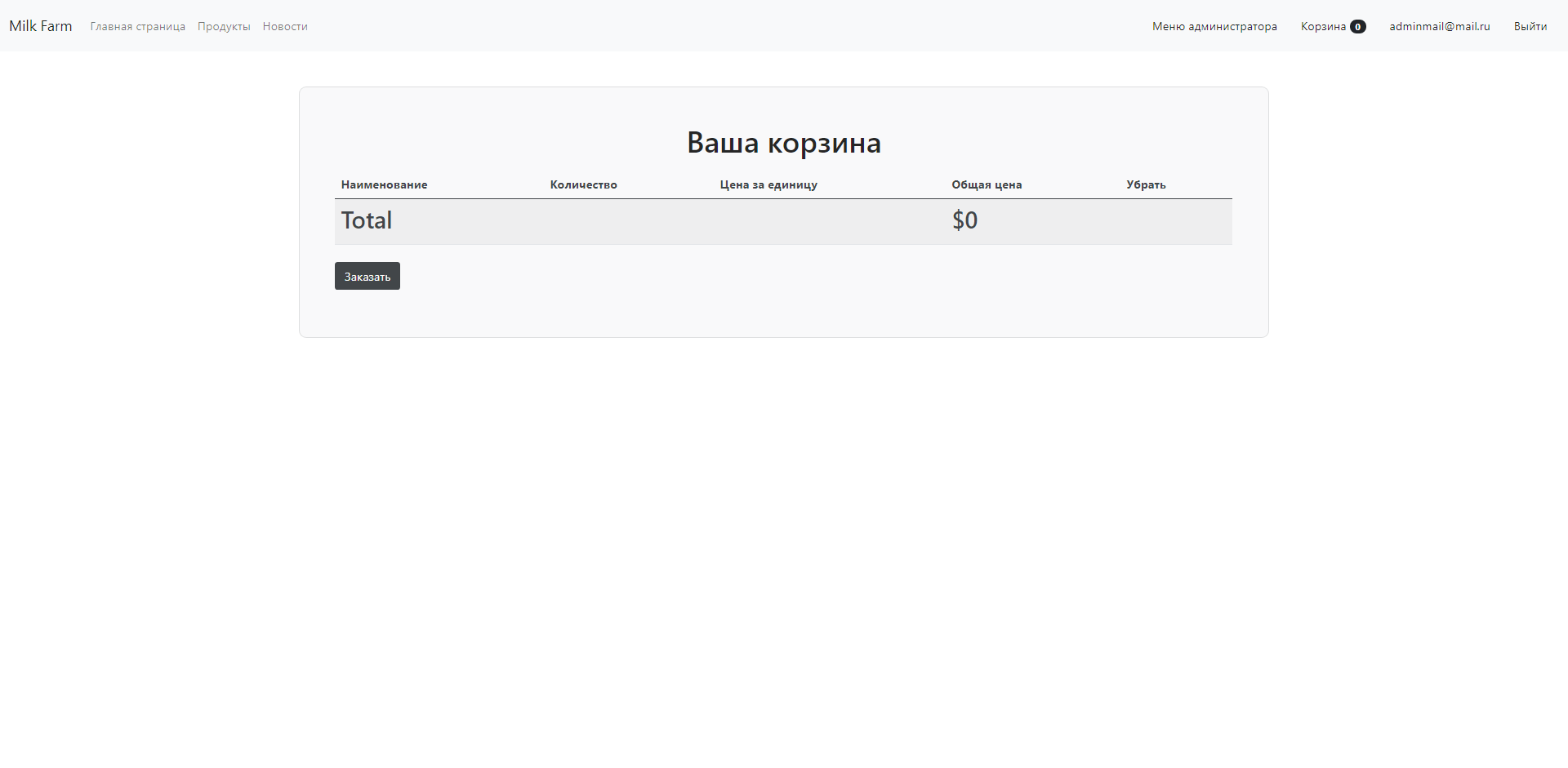
- Главная страница



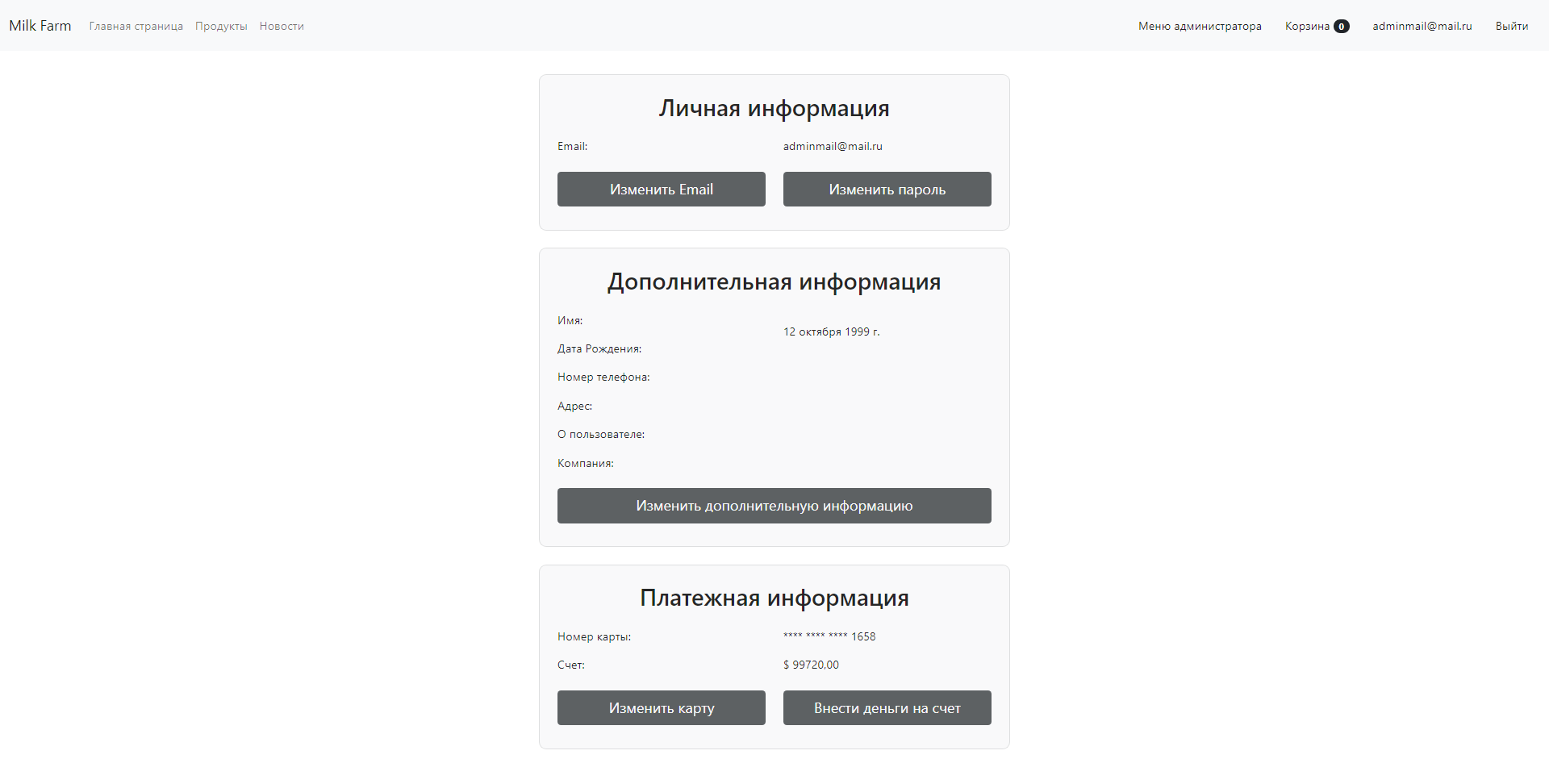
- Страница поиска



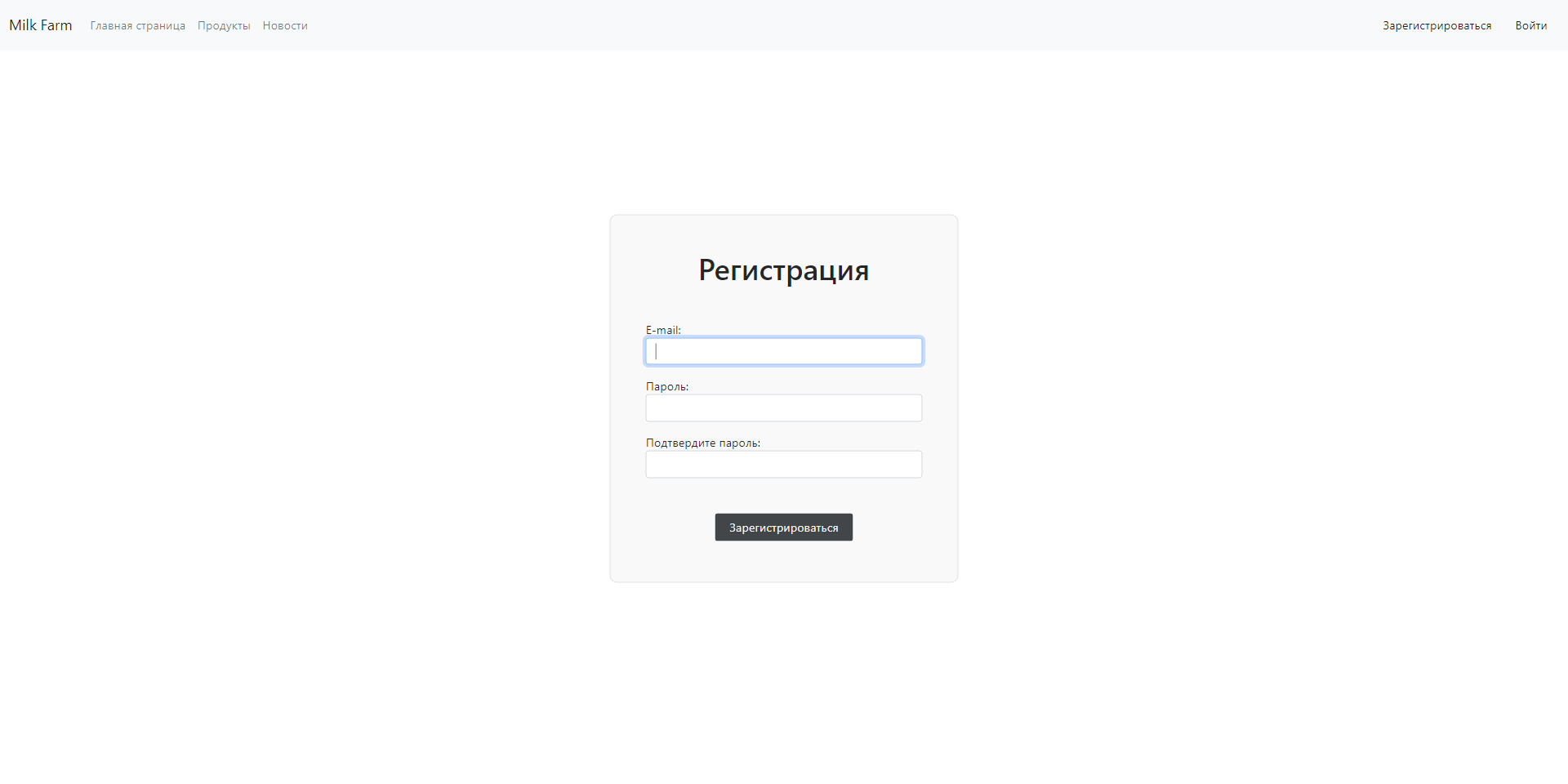
- Страница корзины



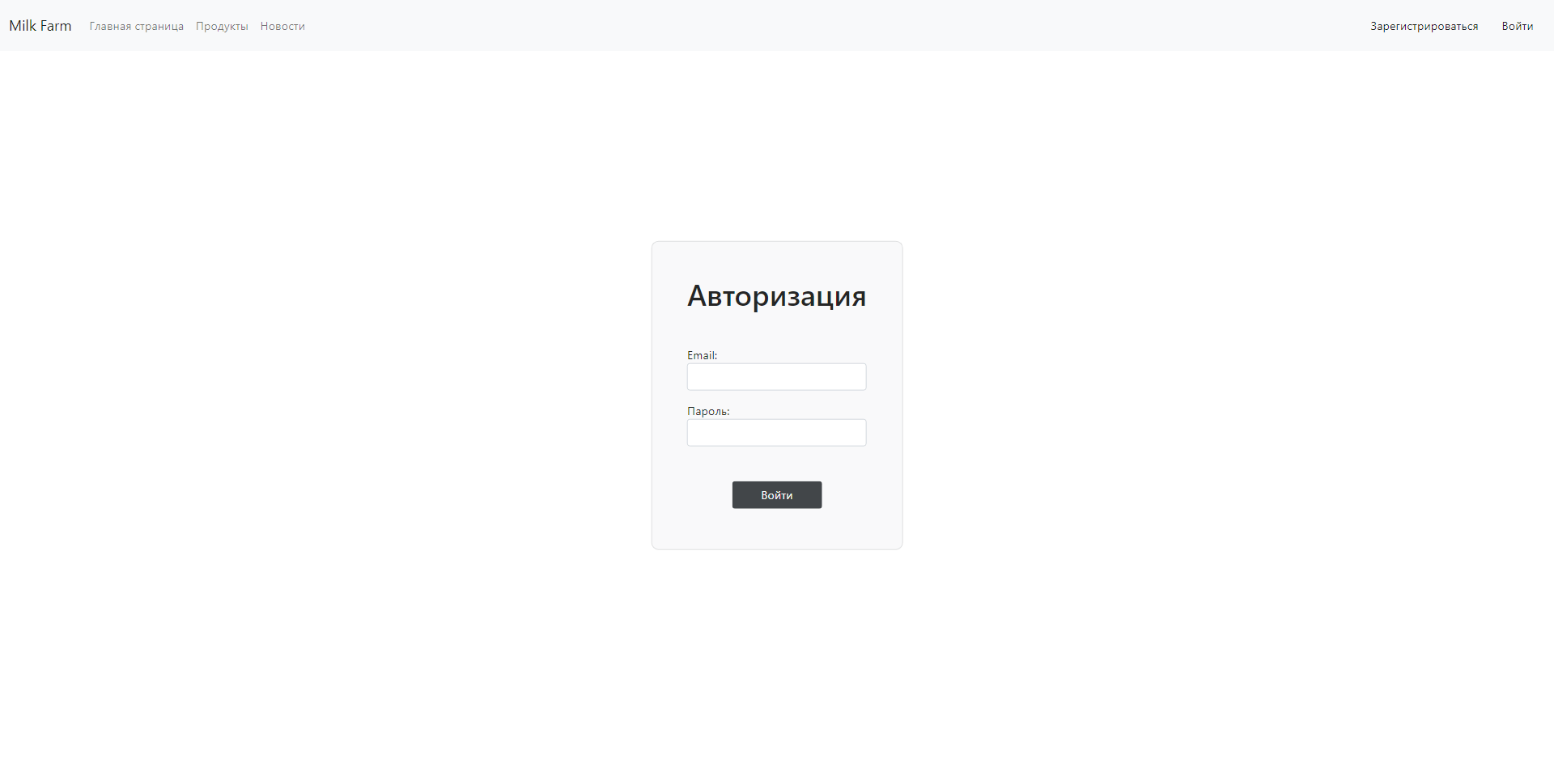
- Аккаунт пользователя



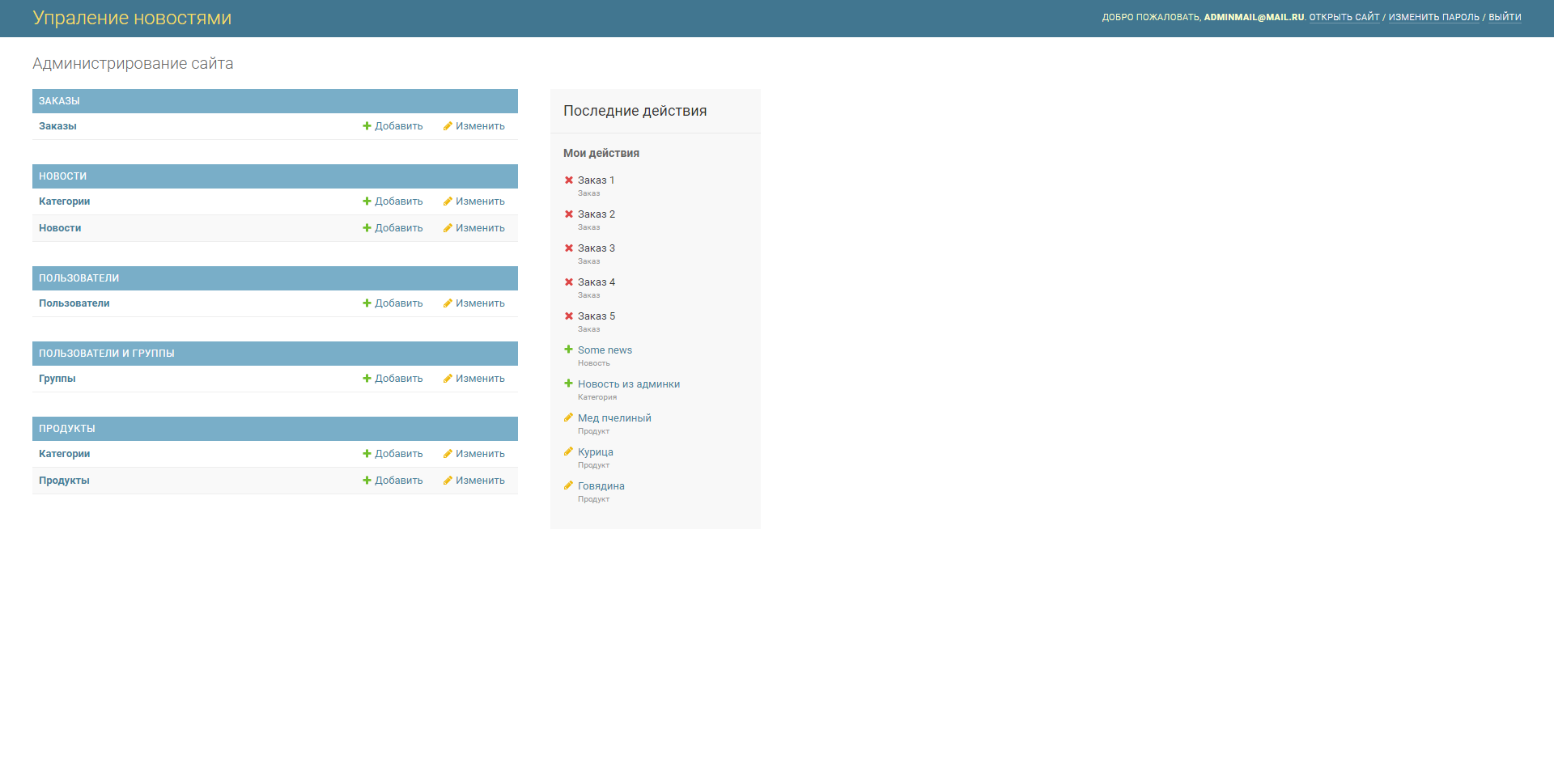
- Страница регистрации



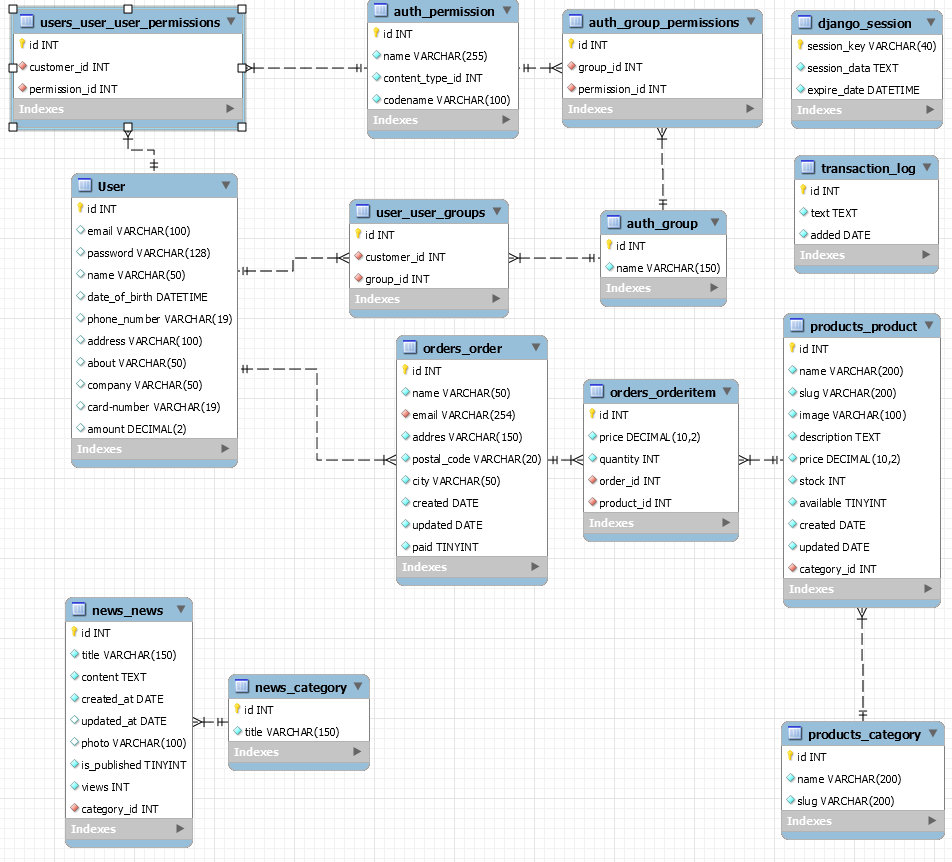
- Страница авторизации



- Страница администратора



# **Приложения.**

ER- диаграмма

1. Исходные коды и документы

https://github.com/nikitaskul/MilkFarm\_

**Вывод.**

Во время выполнения данной работы были ы методы работы с базами данных, способы управления. Также улучшены навыки создания сайтов и связь их с базой данных. В ходе были выполнены все требования, которые нам были даны.