

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: ИС для свопа

Студент гр. 9382

Савельев И.С.

Студент гр. 9382

Кузьмин Д.И.

Преподаватель

Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты

Савельев И.С.

Кузьмин Д.И.

Группа 9382

Тема работы: Разработка веб-приложения для свопа.

Исходные данные:

Необходимо реализовать приложение для СУБД(MongoDB).

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 9382

Савельев И.С.

Студент гр. 9382

Кузьмин Д.И.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В данной работе было разработано ИС для свопа. Для разработки использовались MongoDB^[1] в качестве базы данных, Django^[2] для серверной и клиентской части приложения. Был создан первый прототип в котором реализована возможность просматривать содержимое БД с помощью таблиц, добавлять новые записи. В приложении был реализован массовый экспорт и импорт данных в формате json.

SUMMARY

In this work, an IS for swap was developed. To develop an up-to-date MongoDB database as Django for back-end and front-end applications. The first prototype was created, which implemented the possibility of using the contents of the database using tables, including new records. The application implemented bulk export and import of data in json format.

СОДЕРЖАНИЕ

Введение	6
1. Качественные требования к решению	6
2. Сценарии использования	6
3. Модель данных	8
4. Разработанное приложение	19
Вывод	23
Список использованных источников	24
Приложение	25

1. Введение

Цель работы - создать веб-приложение для предоставления возможности обмена вещами онлайн.

2. Качественные требования к решению

Требуется разработать веб-приложение с использованием СУБД MongoDB.

3. Сценарии использования

Макет UI:

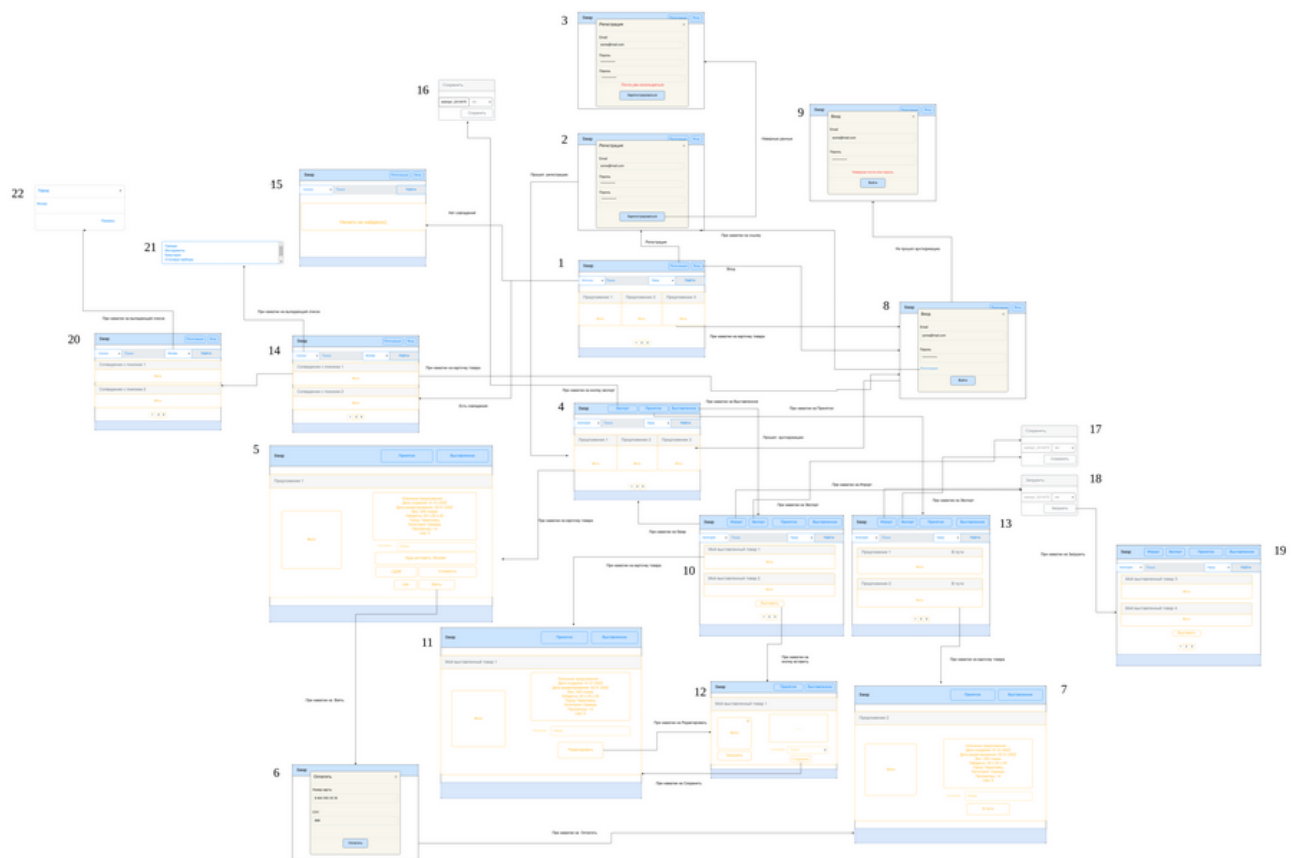


Рисунок 1 –Use Case

Описание сценариев использования:

Сценарий использования – "Регистрация"

Действующее лицо: Пользователь.

Начальные условия: Пользователь зашёл на сайт приложения (экран 1).

Основной сценарий:

Пользователь нажимает на кнопку "Регистрация" (экран 1)

Пользователь вводит данные для регистрации, нажимает кнопку "Зарегистрироваться" (экран 2)

Если данные введены корректно то пользователь авторизуется на сайте (экран 4)

Альтернативный сценарий:

В случае некорректно введённых данных система сообщает об ошибке (экран 3)

Сценарий использования – "Вход на сайт"

Действующее лицо: Пользователь.

Начальные условия: Пользователь зашёл на сайт приложения (экран 1).

Основной сценарий:

Пользователь нажимает на кнопку "Войти" (экран 1)

Пользователь вводит данные для входа, нажимает кнопку "Войти" (экран 8)

Если данные введены корректно то пользователь авторизуется на сайте (экран 4)

Альтернативный сценарий:

В случае некорректно введённых данных система сообщает об ошибке (экран 9)

Сценарий использования – "Взять товар"

Действующее лицо: Пользователь.

Начальные условия: Пользователь авторизован на сайт приложения (экран 4).

Основной сценарий:

Пользователь нажимает на карточку товара (экран 4)

Переходит к описанию товара (экран 5)

Пользователь нажимает кнопку "Взять" (экран 5)

Переходит к оплате доставки товара (экран 6)

Вводит данные банковской карты и нажимает "Оплатить" (экран 6)

Переходит к карточке взятого товара, может отслеживать статус доставки (экран 7)

Сценарий использования – "Поиск товар"

Действующее лицо: Пользователь.

Начальные условия: Пользователь зашёл на сайт приложения (экран 1).

Основной сценарий:

Пользователь вводит наименование товара, устанавливает фильтры и нажимает кнопку найти (экран 1)

Переходит к результатам поиска (экран 14)

Альтернативный сценарий:

В случае отсутствия совпадений, получает соответствующее сообщение (экран 15)

Сценарий использования – "Посмотреть принятый товары"

Действующее лицо: Пользователь.

Начальные условия: Пользователь авторизован на сайт приложения (экран 4).

Основной сценарий:

Пользователь нажимает кнопку "Принятое" (экран 4)

Переходит к каталогу принятых товаров (экран 13)

Нажимает на карточку товара (экран 13)

Переходит к карточке взятого товара, может отслеживать статус доставки (экран 7)

Сценарий использования – "Добавить товар"

Действующее лицо: Пользователь.

Начальные условия: Пользователь авторизован на сайт приложения (экран 4).

Основной сценарий:

Пользователь нажимает кнопку "Выставленное" (экран 4)

Переходит к каталогу выставленных товаров (экран 10)

Пользователь нажимает кнопку "Выставить" (экран 10)

Переходит к карточке товара где может добавить фотографии, описание (экран 12)

Сценарий использования – "Изменить описание товара"

Действующее лицо: Пользователь.

Начальные условия: Пользователь авторизован на сайт приложения (экран 4).

Основной сценарий:

Пользователь нажимает кнопку "Выставленное" (экран 4)

Переходит к каталогу выставленных товаров (экран 10)

Пользователь нажимает карточку товара, который хочет изменить (экран 10)

Переходит к карточке товара и нажимает на кнопку "Редактировать" (экран 11)

Переходит к карточке товара где может добавить или удалить существующие фотографии, изменить описание (экран 12)

Сценарий использования – "Экспорт данных"

Действующее лицо: Пользователь.

Начальные условия: Пользователь авторизован на сайт приложения (экран 4).

Основной сценарий:

Пользователь нажимает кнопку "Экспорт" (экран 4)

Переходит к диалоговому окну, где может выбрать формат файла и название, нажимает кнопку "Сохранить", файл зашুরুзается на компьютер (экран 16)

Сценарий использования – "Импорт данных"

Действующее лицо: Пользователь.

Начальные условия: Пользователь авторизован на сайт приложения (экран 4).

Основной сценарий:

Пользователь нажимает кнопку "Принятое" (экран 4)

Переходит к каталогу принятых товаров (экран 13)

Пользователь нажимает кнопку "Импорт" (экран 13)

Переходит к диалоговому окну, где может выбрать формат файла и название, нажимает кнопку загрузить (экран 18)

Переходит к списку загруженных предметов (экран 19)

4. Модель данных

Схема БД.

Модель NoSQL

```
offers:
{
  _id: ObjectId,
  product: {
    weight: string,
    size: string,
    category: string,
    state: string,
    photo: string,
    description: string
  },
  create_date: date,
  edit_date: date,
  status: string,
  city: string,
  delivery_city: string,
  price: int,
  views: int,
  likes: int,
  owner_id: ObjectId
  purchaser_id: ObjectId
}

users:
{
  email: string,
  password: string,
  full_name: string
}
```

Объем:

$$V = V_o + V_u$$

Учитывая, что размер ObjectId = 12 байт, int = 4 байт, date - 8 байт, string = средняя длина поля*4 байт

Примерный расчет средней длины строковых полей коллекций:

```
offers.product.size - 5
offers.product.category - 8
offers.product.state - 8
offers.product.photo - 30 (url)
offers.product.description - 150
```

offers.status - 8
offers.city - 12
offers.delivery_city - 12

users.email - 14
users.password - 10
users.full_name - 30

$V_o = (4 + 5 + 8 + 8 + 30 + 150 + 8 + 12 + 12) * 4 + 3 * 4 + 3 * 12 + 2 * 8 = 1012$ байт

$V_u = (14 + 10 + 30) * 4 = 216$

Предположим, что каждый пользователь делает 0.2 предложения. Тогда, если количество пользователей n будет 5000, то объем данных будет

$V = n * V_u + 0.2 * n * V_o = 2031, 25$ кбайт

Коллекция избыточна за счет дублирования product. Если заменить это поле на id, то размер каждого документа коллекции offers будет 811

Коллекция users не избыточна

Тогда

$V_{clean} = n * V_u + 0.2 * n * 833 = 1858,3984375$

$V/V_{clean} = 1,099316868$

Запросы:

Аутентификация пользователя:

Предположим, что был введен email test@gmail.com и пароль pass123

```
db.users.find({email: "test@gmail.com", password: "pass123"})
```

Создание предложения об обмене:

Пусть его создает пользователь с id 1

```
db.offers.insertOne({
  {
    product: {
      weight: "3 кг",
      size: "2 м",
      category: "Одежда",
      state: "Новое",
      photo: "https://swapdomain.com/photos/product1.png",
      description: "Отдается новая кофта, почти не ношенная. Не подошла по
размеру"
    },
    create_date: 06.06.2022,
    edit_date: 06.06.2022,
```

```

    status: "Опубликован",
    city: "Москва",
    delivery_city: "Санкт-Петербург",
    price: 300,
    views: 10,
    likes: 2,
    owner_id: 1
    purchaser_id: null
  }
})

```

Посмотреть предложения с товарами, категория, которых - одежда

```
db.offers.find({"product.category": "Одежда"})
```

Осуществить покупку какого-то товара через предложение:

Пусть id предложения - 50, id - пользователя, совершающего покупку - 2

```
db.offers.updateOne({_id: 50}, {$set: {purchaser_id: 2}})
```

Так как при просмотре каждого товара, который еще не купили (`purchaser_id = null`) необходимо делать запрос также к коллекции `users`, то количество запросов будет $n*2$, где n - количество совершенных предложений. Условно $n = 1000$, тогда количество запросов = 2000

Реляционная модель:

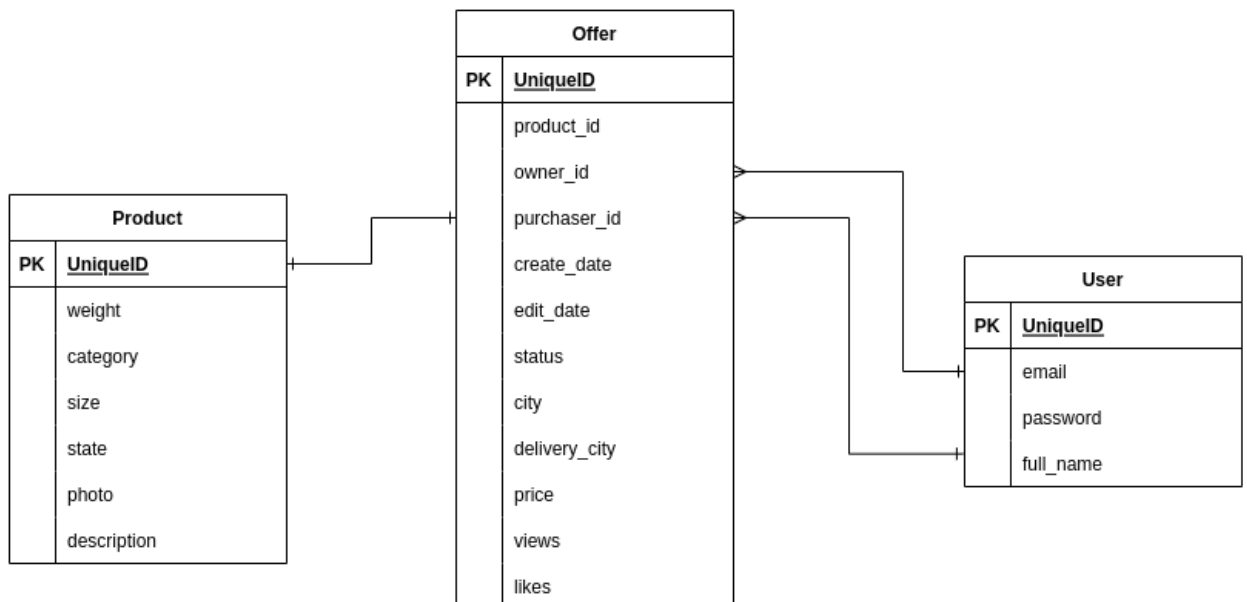


Рисунок 2 – реляционная модель

Объем

$$V = V_p + V_o + V_u$$

Пусть каждое строковое поле будет `varchar(m)`, где `m` - средняя длина поля * 2, а каждое поле `id - serial`

$$V_p = 4 + (4 + 5 + 8 + 8 + 30 + 150) * 2 = 414$$

$$V_o = 3 * 4 + 2 * 4 + (8 + 12 + 12) * 2 + 3 * 4 = 96$$

$$V_u = (14 + 10 + 30) * 2 = 108$$

Пусть также количество пользователей 5000 и каждый пользователь сделал 0.2 предложения

$$V = (V_p + V_o) * 1000 + 5000 * V_u = 1050000 \text{ байт}$$

Чистый объем можно посчитать, убрав связывающие ключи из таблиц

$$V_p = (4 + 5 + 8 + 8 + 30 + 150) * 2 = 410$$

$$V_o = 2 * 4 + (8 + 12 + 12) * 2 + 3 * 4 = 84$$

$$V_u = (14 + 10 + 30) * 2 = 108$$

$$V = (V_p + V_o) * 1000 + 5000 * V_u = 1034000 \text{ байт} = 1009,76562 \text{ кбайт}$$

$$V/V_{\text{clean}} = 1,015473888$$

Запросы:

Аутентификация

```
SELECT * FROM users WHERE  
email = "test@gmail.com" AND  
password = "pass123"
```

Создание предложения об обмене

```
INSERT INTO product  
VALUES ("3 кг", "2 м", "Одежда", "Новое", "https://swapdomain.com/photos/product1.png",  
"Отдается новая кофта, почти не ношенная. Не подошла по размеру");  
INSERT INTO offer  
VALUES (1, 1, NULL, 06.06.2022, 06.06.2022,  
"Опубликован", "Москва", "Санкт-Петербург", 300, 10, 2);
```

Как видно для создания одного предложения об обмене необходимо 2 запроса к БД, в то время как в реляционной модели можно было ограничиться 1м запросом.

Посмотреть предложения с товарами, категория, которых - одежда

```
SELECT * FROM offer JOIN product ON offer.product_id = product.id  
WHERE category = "Одежда"
```

Осуществить покупку какого-то товара через предложение:

Пусть id предложения - 50, id - пользователя, совершающего покупку - 2``

```
UPDATE offer  
SET purchaser_id = 2  
WHERE id = 50
```

Сравнение моделей

Модель NoSQL занимает больше памяти чем модель в SQL. Некоторые запросы в NoSQL более быстрые, чем в SQL, однако некоторые нет

5. Разработанное приложение

Краткое описание

Приложение реализовано на django, для работы с MongoDB используется PyMongo^[3] с помощью которого создается подключение, создание коллекций для хранения информации о вещах и пользователях, добавление и фильтрация. Клиентская часть реализована с помощью шаблонов django и стилей css.

Схема экранов приложения

Экраны приложения и переходы между ними представлены на рисунке .

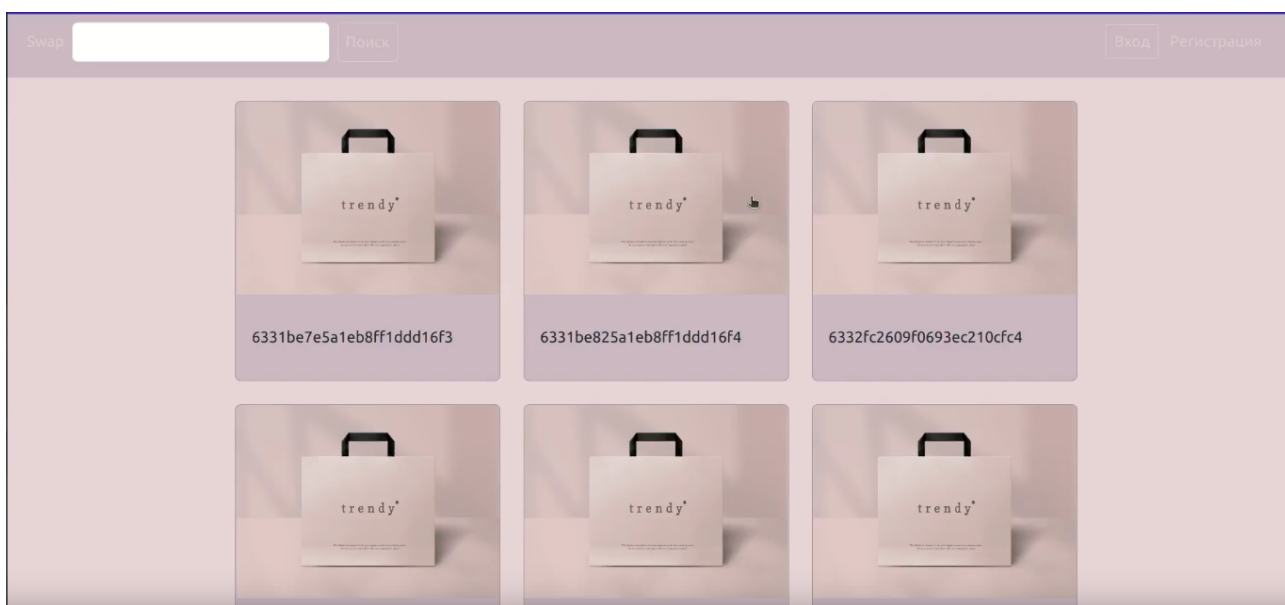


Рисунок 3 - Главная



Рисунок 4 - Кнопка экспорт всех предложений на главной странице



The registration form is located on a light purple background. At the top, there is a header bar with a 'Swap' button, a search input field with a 'Поиск' button, and 'Вход' and 'Регистрация' buttons. The main form area contains three input fields labeled 'Email:', 'Пароль:', and 'ФИО:'. Below these fields is a 'Зарегистрироваться' button.

Swap Поиск

Вход Регистрация


Email:

Пароль:

ФИО:

Зарегистрироваться

Рисунок 5 - Регистрация



The authorization form is located on a light purple background. At the top, there is a header bar with a 'Swap' button, a search input field with a 'Поиск' button, and 'Вход' and 'Регистрация' buttons. The main form area contains two input fields labeled 'Email:' and 'Пароль:'. The 'Email:' field contains the text 'newuser6@gmail.com'. Below the 'Пароль:' field is a 'Войти' button.

Swap Поиск

Вход Регистрация

Email:

Пароль:

Войти

Рисунок 6 - Авторизация


Swap Поиск

Название:

Описание:

Photo: No file chosen

Рисунок 7 - Создание нового предложения



Вес:

Габариты:

Город:

Категория:

Состояние:

Просмотр:

```
{'_id': ObjectId('6387bda6efec7c2d407db406'), 'title': 'New offer', 'photo': '/media/tmp/somename_eQcD7mv.png'}
```

6387bda6efec7c2d407db406

Рисунок 8 - Страница отдельного предложения

Использованные технологии

СУБД: MongoDB

Backend: Python, Django, PyMongo

Frontend: HTML, CSS.

Ссылки на приложение

Ссылка на github: <https://github.com/moevm/nosql2h22-swap>

ВЫВОД.

Результаты

В ходе работы было разработано web-приложение “Swar”, позволяющее пользователям взаимодействовать с базой данных: просмотр содержимого СУБД, добавление новых элементов, экспорт и импорт данных, реализована регистрация и авторизация.

Недостатки и пути для улучшения полученного решения

На данном этапе веб элементы не соответствуют веб макету, отсутствует возможность редактировать предложение, не реализована возможность забрать предложение.

Будущее развитие решения

Планируется реализовать недостающие возможности, доделать верстку приложения.

ПРИЛОЖЕНИЕ

Документация по сборке и развертыванию приложения

1. Скачать проект из репозитория
2. Запустить docker командой `docker-compose up --build -d`
3. Открыть приложение в браузере по адресу `localhost:49088`

ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

1. Документация MongoDB: <https://www.mongodb.com/docs/>
2. Документация Django: <https://docs.djangoproject.com/en/4.1/>
3. Документация PyMongo: <https://pymongo.readthedocs.io/en/stable/>