

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

Факультет Инфокоммуникационных технологий

Образовательная программа Интеллектуальные системы в гуманитарной сфере

Направление подготовки (специальность) Интеллектуальные системы в гуманитарной сфере

О Т Ч Е Т

по курсовой работе по дисциплине «Основы Web-программирования»

Тема задания: Web-сервис для организаторов выставок собак

Обучающийся Лукина А. С., группа К3343

Преподаватель дисциплины: Говоров А.И.

Оценка за курсовую работу ____

Подписи членов комиссии:

(подпись)

(подпись)

(подпись)

Дата ____

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
ПРЕДМЕТНАЯ ОБЛАСТЬ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ	4
1.1. Предметная область	4
1.2. Функциональные требования	6
2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ	8
3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ	13
4. ОПИСАНИЕ СРЕДСТВ РАЗРАБОТКИ	24
ЗАКЛЮЧЕНИЕ	26
СПИСОК ЛИТЕРАТУРЫ	27

ВВЕДЕНИЕ

Курсовая работа посвящена web-сервису для организаторов выставок собак. Организация любых выставок — сложный трудоёмкий процесс, который требует большого внимания и учёта множества факторов, влияющих на качество выставки. Один и тот же человек может выступать организатором огромного числа выставок, из-за чего встает вопрос не только о хранении и свободном доступе к собственным проектам, но и об эффективном планировании и улучшении качества организации мероприятия. В частности, для выставок собак для организатора важно иметь возможность быстрого доступа к списку животных и их хозяев с указанием контактных данных; планировать выставку, разделяя разные соревнования в разные дни и указывать разные ринги для выступления участников; просматривать отчётную информацию по пройденной выставке и производить поиск нужных экспертов.

В решении этих и ряда других проблем может помочь разрабатываемый сервис, который предоставляет быстрый доступ к функциональным возможностям в сети Интернет с использованием персонального компьютера или мобильного устройства.

Цель: разработать веб-сервис для организаторов выставок собак с использованием средств Django REST framework и Vue.js.

Задачи:

1. Изучить предметную область;
2. Выделить функциональные характеристики;
3. Спроектировать и реализовать базу данных;
4. Реализовать серверную часть;
5. Реализовать клиентскую часть.

ПРЕДМЕТНАЯ ОБЛАСТЬ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1.1. Предметная область

Предметной областью для курсовой работы является организация ежегодных выставок собак. Выставки могут быть двух типов: монопородные (выставки, в которых принимают участие собаки только одной заранее определенной организатором породы) и полипородные (выставки, для которых критерий отбора для участия выбирается иным способом, т.е. не ограничены породой). Помимо этого, у выставки могут быть спонсоры — люди или компании, финансирующие выставки, которые также могут влиять на различного рода ограничения.

Выставка, по усмотрению организатора, может быть назначена на один день или на несколько. Каждая выставка состоит из одного или более соревнований, каждое из которых проходит в свой день и свой собственный ринг, который обслуживают эксперты. Каждый ринг могут обслуживать несколько экспертов. Помимо этого, каждая порода собак выступает на своем ринге, но на одном и том же ринге в разное время могут выступать разные породы.

Процедура организации проходит следующим образом: хозяин указывает различную информацию о себе и своей собаке: кличка, порода и возраст, классность, сведения о родословной (номер документа, клички родителей), дата последней прививки для собаки, и фамилия, имя, отчество и паспортные данные для себя. Кроме того, так как участие может быть индивидуальным или от клуба, то указывается и название клуба, если собака принадлежит к одному из них. Перед соревнованиями каждая собака должна пройти обязательный медосмотр, без которого ни одно животное не будет допущено ни на одно из соревнований. Помимо медосмотра, обязательным является и наличие оплаты, так как участие на выставке является платным. Без отметки об оплате собака также не будет допущена. Пройти медосмотр и оплатить участие, хозяева животных должны после этапа регистрации, после чего они могут приступать к выполнению упражнений в свой время и на своем ринге. Каждая собака должна выполнить три упражнения, за каждое из которых она получает баллы от экспертов. Сведения об экспертах включают фамилию и имя, номер ринга, который он обслуживает, клуб, название клуба, в котором он состоит. Итогом выставки является определение медалистов по каждой породе по итоговому рейтингу.

Важными действиями организатора также являются добавление нового участника или нового эксперта, снятие эксперта с судейства (или замена его другим), а также отстранение собаки от участия в выставке. Помимо этого, организатору важно знать следующую информацию: на каком ринге выступает заданный хозяин со своей собакой; какими породами

представлен заданный клуб; сколько собак были отстранены от участия в выставке; какие эксперты обслуживают породу; и количество участников по каждой породе. Часто организаторы составляют отчетную информацию, например, о результатах заданной выставки: сколько всего участников было представлено, какие породы участвовали на соревновании, сколько медалей было получено по каждой породе.

1.2. Функциональные требования

Функциональные требования должны отвечать всем установленным запросам и максимально удовлетворять потребностям предметной области, которая в подробности была описана в главе 1.1. Все функциональные требования можно поделить на несколько групп, разделенных по виду манипулирования информацией, а именно: блок требований, касающийся функций просмотра информации, блок требований для функций добавления информации, блок требований для функций редактирования информации, а также для удаления информации. Помимо этого, отдельной группой стоят такие функциональные возможности как регистрация и авторизация.

Структура классификации функциональных требований представляет собой следующее разделение:

6. Функции просмотра:
 - a. Информации о выставках;
 - b. Информации о соревнованиях;
 - c. Информации о хозяевах собак;
 - d. Информации об экспертах;
 - e. Информации о регистрации собак;
 - f. Информации о результатах выставки;
 - g. Отчетных материалов по каждой выставке;
 - h. Информации из личного профиля;
7. Функции добавления:
 - a. Добавление новой выставки;
 - b. Добавление нового соревнования;
 - c. Назначение эксперта на выставку;
 - d. Регистрация собаки;
 - e. Назначение собаки на выставку;
8. Функции редактирования:
 - a. Редактирование личной информации из профиля
9. Функции удаления:
 - a. Снятие эксперта с судейства
 - b. Отстранение собаки от участия
10. Функции регистрации и авторизации.

Для наилучшей реализации сервиса система должна хранить следующую информацию: для выставки: название, город, адрес, тип выставки, дата начала, дата завершения, изображение и организатора, который занимается данной выставкой. Для соревнования: номер ринга, названия трех упражнений, дата проведения и выставка, на которой проходить соревнование. Для владельца собаки: имя, фамилия, номер паспорта, город, телефон и электронная почта. Для экспертов: имя, фамилия, телефон, электронная почта, образование и небольшая информация о себе. О собаках: кличка, порода, дата рождения, информация о владельце. Для регистрации: выставка, на которую проходит регистрация, владелец, совершающий регистрацию и его питомец (так как у одного хозяина может быть несколько питомцев), а также информация об оплате и медосмотре. Для результатов: эксперт, который поставил оценку, его оценки по каждому из трёх упражнений и конкретное соревнование конкретной собаки. Для клубов: название клуба и его участники.

К функциональным характеристикам также можно отнести следующие требования:

1. При назначении собаки на соревнование необходимо реализовать вывод только тех собак, которые принадлежат владельцу, которого организатор хотел бы добавить на выставку.
2. При выводе результатов соревнований по выставке необходимо выводить записи только в порядке убывания по сумме трех баллов за упражнения.
3. Разграничить доступ, скрывая от гостей основной функционал и предоставляя им возможность только просматривать список выставок.
4. При снятии эксперта или отстранения собаки необходимо реализовать выпадающий список только из экспертов или собак, которые ранее уже были добавлены, при этом ограничить записи для выставок, которые были организованы пользователем, который непосредственно работает с системой.

2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

В первую очередь была спроектирована схема базы данных, представленная на рисунке 1. На схеме отображены все функциональные характеристики, необходимые для реализации программной системы и которые описаны в главе 1.2.

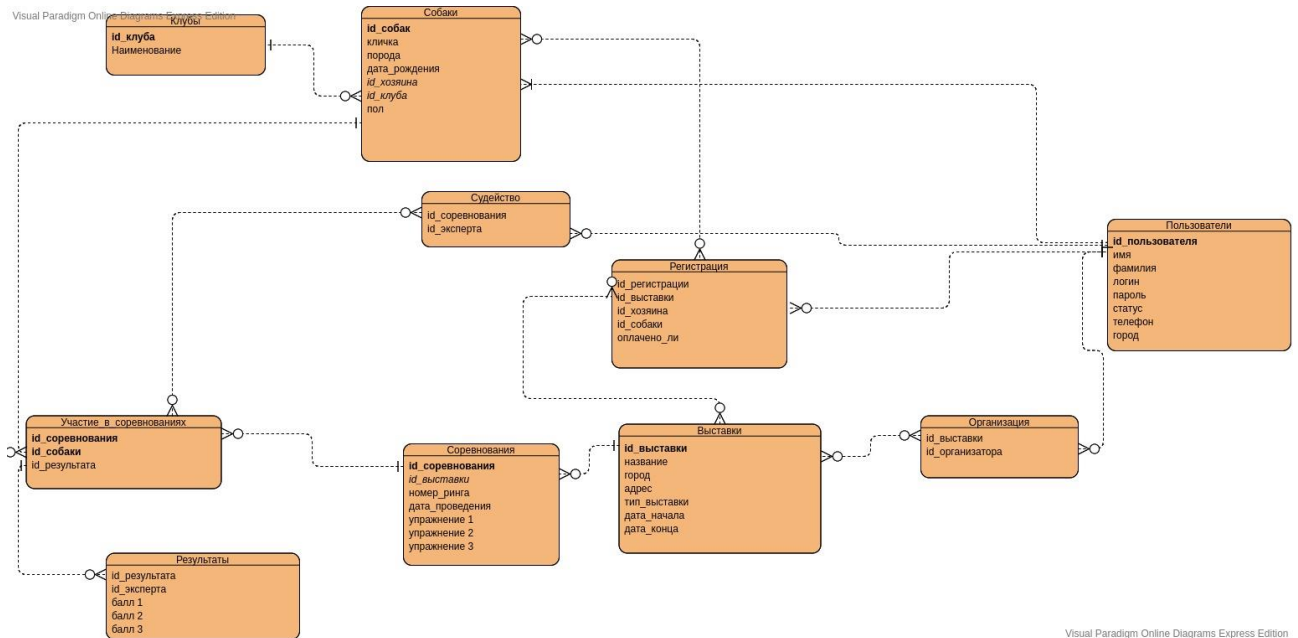


Рисунок 1 - ER-диаграмма для программной системы "Выставка собак"

Программная система состоит из двух главных частей: бэкенд и фронтэнд, которые реализованы с помощью Django REST Framework и Vue JS соответственно. Главное преимущество использования данных технологий состоит в том, что серверную и клиентскую часть можно модернизировать параллельно, что позволяет сервису, написанному с указанными технологиями быть легко масштабируемым.

Использование Django REST Framework позволило работать с Django API, что упростило работу с отслеживанием состояния системы и формированию запросов. Приведем пример такого использования для функционала пользователей (Users).

Для начала было необходимо описать сериализатор в файле *serializers.py*, который выглядит следующим образом:

```
class ProfileSer(serializers.ModelSerializer):
    """Сериализация профиля"""
    user = UserSerializer()
    follow = UserSerializer(many=True)

    class Meta:
        model = Profile
        fields = ('__all__')
```

Рисунок 2 - Сериализация профиля пользователя

Данный сериализатор был использован для написания представления в файле *views.py*. Ниже представлены два представления на основе сериализатора профиля: вывод и редактирование профиля.

```
class ProfileUser(APIView):
    """Вывод профиля пользователя"""
    permission_classes = [permissions.IsAuthenticated]

    def get(self, request):
        ser = ProfileSer(Profile.objects.get(user=request.user))
        return Response(ser.data)

class UpdateProfile(APIView):
    """Редактирование профиля"""
    permission_classes = [permissions.IsAuthenticated]

    def post(self, request):
        prof = Profile.objects.get(user=request.user)
        ser = EditAvatar(prof, data=request.data)
        if ser.is_valid():
            if "avatar" in request.FILES:
                ser.save(avatar=request.FILES["avatar"])
                return Response(status=201)
            else:
                return Response(status=400)
```

Рисунок 3 - Представления для вывода и редактирования профиля пользователя

Для того, чтобы выйти на страницу API и посмотреть полученную информацию, необходимо прописать путь в файле *urls.py*:

```
urlpatterns = [
    path('', ProfileUser.as_view()),
    path('update-ava/', UpdateProfile.as_view()),
]
```

Рисунок 4 - Пути для доступа

Используя любой из указанных путей, можно просмотреть «серверную» информацию о конкретной модели, описанной в сериализаторе. Например, на рисунке ниже представлено отображение информации для модели пользователя:

Profile User

Вывод профиля пользователя

GET /api/v1/profile/

HTTP 200 OK
 Allow: GET, HEAD, OPTIONS
 Content-Type: application/json
 Vary: Accept

```
{
  "id": 3,
  "user": {
    "id": 3,
    "username": "eviekevie"
  },
  "avatar": null,
  "name": "Анастасия",
  "surname": "Лукина"
}
```

Рисунок 5 - Вывод информации о пользователях через API

На рисунке 5 показано отображение одного из пользователей, уже зарегистрированного в системе, с указанием его информации. С каждой новой регистрацией информация о пользователях будет пополняться.

Описанным выше способом был реализован и дальнейший функционал системы, описывающий все модели, отображенные в файле *models.py*. (Реализованная база данных). Файлы *models.py* всех приложений содержат следующие модели, представленные в таблице 1:

Таблица 1 – Модели, описанные в файле models.py

Название модели	Назначение
Exhibition	Информация о выставках
Competition	Информация о соревнованиях
DogOwner	Информация о хозяевах собак
Expert	Информация об экспертах
ExpertCompetition	Информация о судействе (какие эксперты назначены на соревнования)
Dog	Информация о собаках
DogRegistration	Информация о регистрации собаки на выставку
CompParticipation	Информация об участии на соревновании
Result	Информация о результатах конкретного соревнования
Club	Информация о клубах
ClubParticipation	Информация о членстве в клубах
Dismissed	Информация об отстранённых собаках
User	Информация о пользователе

Файлы *views.py* представлен следующими представлениями, которые отображены в таблице 2:

Таблица 2 – Представления, описанные в файле views.py

Название представления	Описание
exhibition_info	Информация о всех выставках
exhibition_add	Добавление выставки
one_exhibition_info	Информация о конкретной выставке и отчёта по ней
experts_output	Информация обо всех экспертах
set_experts	Назначение и снятие экспертов на выставку
competition_add	Добавление соревнования к выставке
dog_to_comp	Назначение и снятие собак с соревнования
dog_reg	Регистрация собак на выставку
query	Представление для формирования пяти запросов
ProfileView	Вывод информации о пользователе
ProfileEditView	Редактирование пользователя

Помимо прочего были определены формы для внесения изменений в существующую базу данных, которые хранятся в файлах *forms.py*. Информация о них представлена в таблице 3.

Таблица 3 – Формы, описанные в файле forms.py

Форма	Назначение
ExhibitionForm	Форма добавления выставки
SetExpertForm	Форма назначения эксперта на выставку
DelExpertForm	Форма снятия эксперта с выставки
CompetitionForm	Форма добавления соревнования
DogToCompForm	Форма назначения собаки на соревнования
DelDogFromCompForm	Форма снятия собаки с соревнования
DogRegForm	Форма регистрации собаки на участие в выставке
Query2Form	Форма запроса №2
Query3Form	Форма запроса №3
ProfileForm	Форма редактирования профиля

Таким образом в данной работе представлена серверная часть.

3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

Клиентская сторона представлена шаблонами и файлами с расширением `.vue`. Все функциональные `vue` файлы хранятся в директории ***components*** и состоят из трёх частей: шаблона, скриптовой части и части стилей. Приведем пример описания одного из таких файлов — `Login.vue` (авторизация). Авторизация представляет собой модальное окно, которое было реализовано посредством технологии AJAX, что отображено в части скриптов.

Первая часть — часть шаблонов — ограничена тэгом `template` и описывает внешний структурный вид модального окна, как показано на рисунке 6:

```
<template>
  <div class="" id="loginModal">
    <div class="modal-dialog modal-dialog-centered auth-modal">
      <div class="modal-content">
        <!-- Modal Header -->
        <div class="modal-header">
          <h4 class="modal-title">Вход</h4>
          <button @click="close" type="button" class="close" data-dismiss="modal">
            &times;
          </button>
        </div>

        <!-- Modal body -->
        <div class="modal-body">
          <p>{{mess}}</p>
          <input type="text" placeholder="Логин" value="" v-model="user.username">
          <input type="password" placeholder="Пароль" value="" v-model="user.password">
          <button type="button" @click="setLogin">Войти</button>
        </div>
      </div>
    </div>
  </div>
</template>
```

Рисунок 6 - Часть шаблонов в файле `Login.vue`

Вторая часть — часть скриптов — ограничена тэгом `script` и описывает функции, срабатывающие при нажатии на кнопки. Часть скрипта представлена на рисунке 7.

```
<script>
export default {
  name: "Login",
  data() {
    return {
      user: {
        username: "",
        password: ""
      },
      mess: ''
    }
  },
  methods: {
    setLogin() {
      $.ajax({
        url: this.$store.getters.get_url_server + 'auth/token/login/',
        type: "POST",
        data: {
          username: this.user.username,
          password: this.user.password
        },
        success: (response) => {
          sessionStorage.setItem("token", response.auth_token)
          this.$store.commit("set_auth", true)

          $.ajaxSetup({
            headers: {'Authorization': "Token " + sessionStorage.getItem('token')},
          });
          this.close()
        },
        error: (response) => {
          if (response.status === 400) {
            this.mess = response.responseJSON.non_field_errors[0]
          }
        }
      })
    },
    close() {
      this.$emit("hideLogin")
    }
  }
}
</script>
```

Рисунок 7 - Часть скриптов в файле Login.vue

И последняя часть — часть стилей — ограничена тэгом `style` и описывает внешний вид шаблона (css стили), представлена на рисунке 8:

```
<style scoped>
  #loginModal {
    position: fixed;
    z-index: 1000;
    top: -150px;
    left: 40%;
  }
</style>
```

Рисунок 8 - Часть стилей в файле Login.vue

Таким образом в данной работе представлен фронтэнд. К этому блоку также можно отнести все назначенные html-шаблоны. Ниже будут представлены основные интерфейсы с логикой работы программы.

В меню представлено пять опций: «Эксперты», «Выставки», «Запросы», аккаунт и «Выход» или «Вход», если вход в системы не был совершен. На рисунке 9 представлен пример выпадающего списка из опции «Выставки» для доступа к основному функционалу.

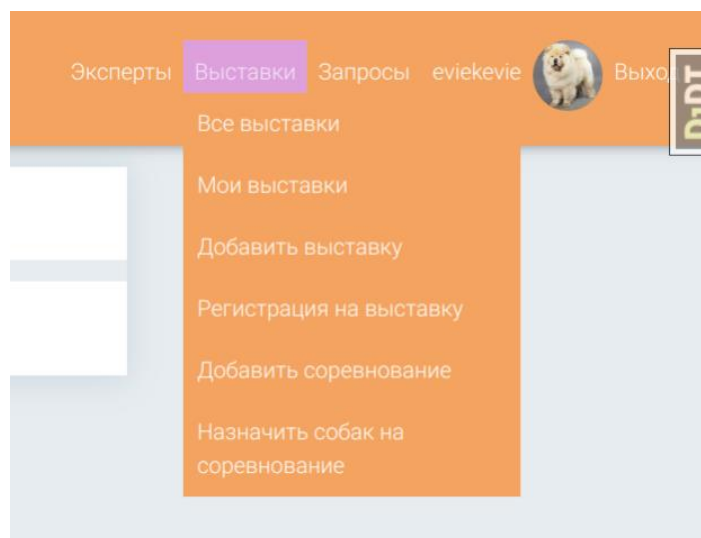


Рисунок 9 - Главное меню сайта

Вывод всех выставок представляет собой список «карточек», при нажатии на которые раскрывается полная информация о выставке. Пример представлен на рисунке 10.

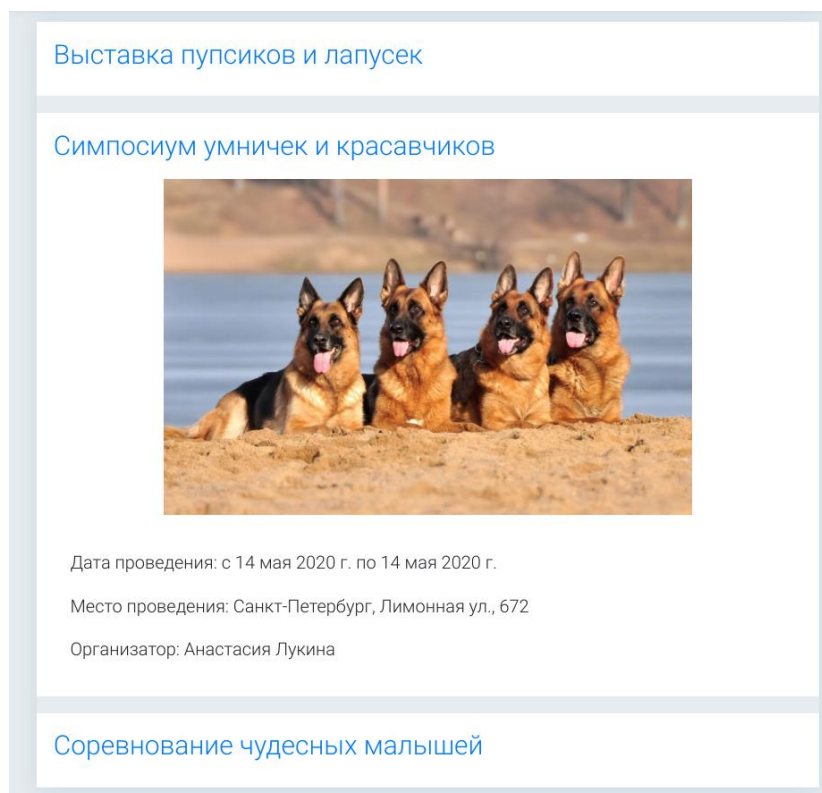
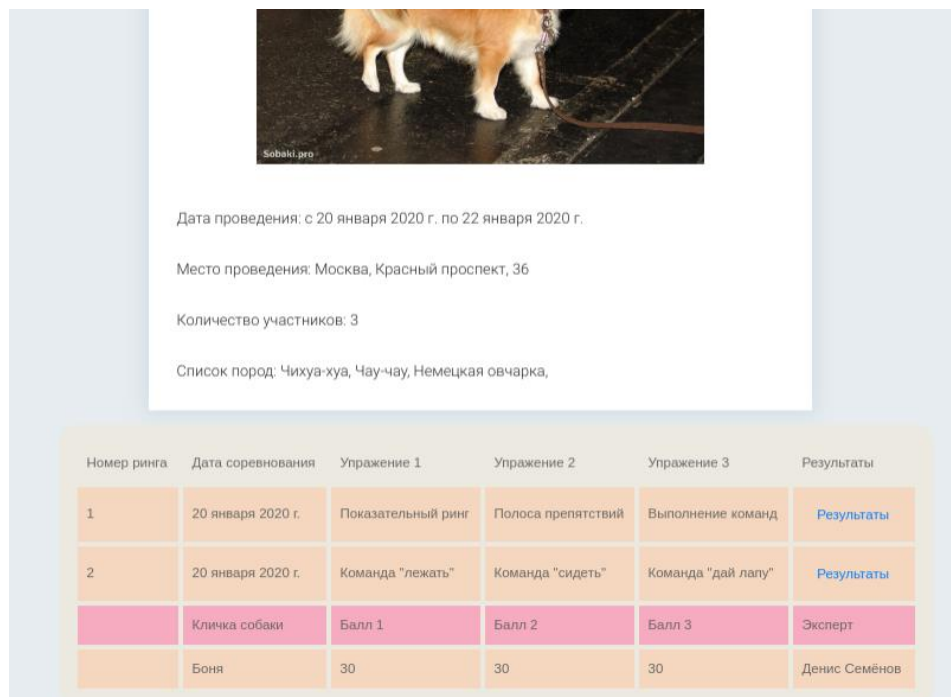


Рисунок 10 - Вывод всех выставок

Информация о конкретной выставке включает себя её название, сроки проведения, организатора и отчётной информации (результаты). Пример представлен на рисунке 11.



Номер ринга	Дата соревнования	Упражнение 1	Упражнение 2	Упражнение 3	Результаты
1	20 января 2020 г.	Показательный ринг	Полоса препятствий	Выполнение команд	Результаты
2	20 января 2020 г.	Команда "лежать"	Команда "сидеть"	Команда "дай лапу"	Результаты
	Кличка собаки	Балл 1	Балл 2	Балл 3	Эксперт
	Боня	30	30	30	Денис Семёнов

Рисунок 11 - Пример вывода информации о выбранной выставке

Интерфейс регистрации собаки на выставку содержит поля с выбором выставки, хозяина и его собаки. Поле ввода собаки предусмотрено таким образом, чтобы были доступны только собаки выбранного хозяина. Пример регистрации представлен на рисунке 12.

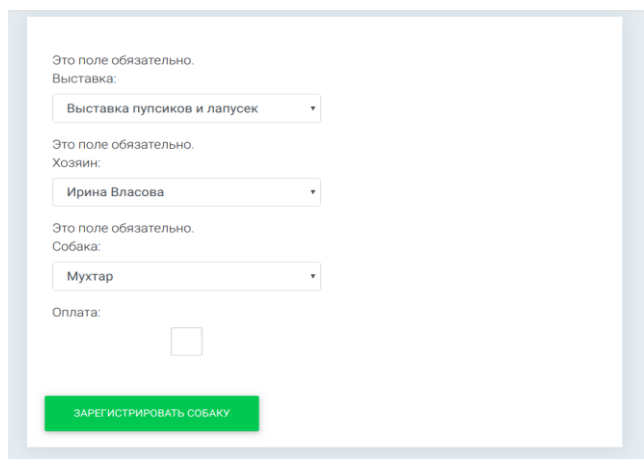


Рисунок 12 - Пример регистрации собаки на выставку

Добавление соревнования на выставку подразумевает ввод номера ринга, названия трёх упражнений, дату и саму выставку. Список выставок включает только выставки организатора,

который произвел вход в систему. При этом, если организатор по случайности укажет неверную дату, то систему ему об этом сообщит. Пример приведен на рисунке 13.

Выберите подходящие данной выставке даты с 13 мая 2020 г. по 15 мая 2020 г.

Номер ринга:

Упражнение №1:

Упражнение №2:

Упражнение №3:

Дата:

Выставка:

Рисунок 13 - Пример добавления соревнования на выставку

Назначение и снятие собаки с соревнования представлены на одной странице с возможностью переключения предусмотренными кнопками. При назначении собаки организатору предоставлена возможность выбора только тех собак, которые были зарегистрированы на участие в выставке, на которой рассматриваемое соревнование проходит. В свою очередь для снятия собак предоставлен выбор собак, которые ранее были назначены на соревнование. Пример представлен на рисунке 14.

Это поле обязательно.
Соревнование:

Это поле обязательно.
Собака:

Рисунок 14 - Пример страницы с назначением и снятием собак с соревнований

Назначение и снятие экспертов с выставки реализовано похожим способом. Список соревнований включает себя только те соревнования, которые являются частью выставок того пользователя, который вошел в систему. Пример представлен на рисунке 15.

Назначить Снять

Это поле обязательно.
Соревнование:

Симпозиум умничек и красавчиков 2020-05-14 1 ринг

Это поле обязательно.
Эксперт:

Валерия Павлова

НАЗНАЧИТЬ ЭКСПЕРТА

Рисунок 15 - Пример страницы с назначением и снятием экспертов с соревнований

Помимо прочего в системе реализованы формы для пяти запросов. Первый запрос представляет собой форму для ответа на вопрос «На каком ринге выступает заданный хозяин со своей собакой?» Для того, чтобы узнать интересующую информацию необходимо указать только релевантную информацию, в противном случае систему сообщит об ошибке (например, о несовпадении принадлежности собаки хозяину). Пример успешного запроса представлен на рисунке 16, а на рисунке 17 — не успешного.

На каком ринге выступает заданный хозяин со своей собакой?

Введите имя хозяина:

Введите кличку собаки:

УЗНАТЬ

Мария Мышкина с собакой Боня выступает на ринге № 2

Рисунок 16 - Пример успешно выполненного запроса №1

На каком ринге выступает заданный хозяин со своей собакой?

Введите имя хозяина:

Введите кличку собаки:

УЗНАТЬ

Такого хозяина нет в базе данных или у указанного хозяина нет указанной собаки

Рисунок 17 - Пример ошибки при некорректном запросе №1

Форма запроса №2 представляет собой одно поле с выбором всем доступных клубов для того, чтобы ответить на вопрос «Какими породами представлен заданный клуб?». Пример выполнения этого запроса представлен на рисунке 18.

Какими породами представлен заданный клуб?

Клуб:

Все псы попадут в рай ▼

УЗНАТЬ

Немецкая овчарка

Чихуа-хуа

Рисунок 18 - Пример выполнения запроса №2

Запрос №3 служит для ответа на вопрос «Сколько собак были отстранены от участия в выставке?». Система ищет по каждому соревнованию для указанной выставки и выводит информацию о количестве и список отстранённых собак. Пример представлен на рисунке 19.

Сколько собак были отстранены от участия в выставке?

Выставка:

Соревнование чудесных малышей ▾

УЗНАТЬ

Всего отстранённых собак: 1

Подробнее: Рекс

Рисунок 19 - Пример выполнения запроса №3

Запрос №4 отвечает на вопрос «Какие эксперты обслуживают породу?». Для этого организатору необходимо из списка выбрать породу, и в случае успеха, система выводит список экспертов. Пример представлен на рисунке 20.

Какие эксперты обслуживают породу?

Выберите породу из списка:

Немецкая овчарка ▾

УЗНАТЬ

Павлова Валерия

Рисунок 20 - Пример выполнения запроса №4

Последний запрос ищет количество участников по каждой породе. Для этого нужно указать выставку и интересующую породу. В случае успеха система выводит количество собак, а также список собак. Пример представлен на рисунке 21.

Количество участников по каждой породе?

Выберите выставку:

Выставка пупсиков и лапусек ▾

Выберите породу из списка:

Немецкая овчарка ▾

УЗНАТЬ

С указанными параметрами количество участников: 1


Мухтар

Рисунок 21 - Пример выполнения запроса №5

Помимо рассмотренных функций пользователь (организатор) имеет возможность просмотреть и отредактировать свои личные данные. При регистрации пользователю устанавливается стандартный аватар, который при желании можно поменять на любой другой. При успешном редактировании система выводит оповещение о сохранённых изменениях. Пример просмотра информации представлен на рисунке 22, редактирования — на рисунке 23.

Здесь отображены Ваши данные:

Ваша фотография:



Имя: Анастасия
Фамилия: Лукина

РЕДАКТИРОВАТЬ
ДАННЫЕ

Рисунок 22 - Просмотр личной информации в профиле

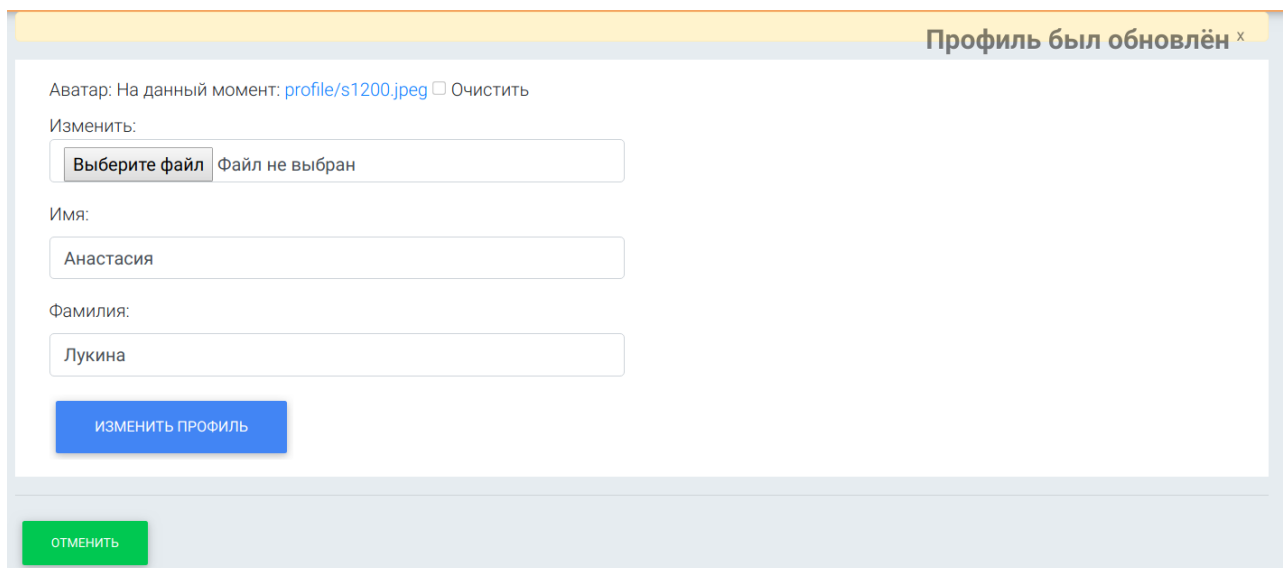


Рисунок 23 - Пример редактирование личной информации в профиле

Новые пользователи также имеют возможность использовать систему, но перед этим им необходимо пройти процесс регистрации (рисунок 24) и авторизации (рисунок 25).

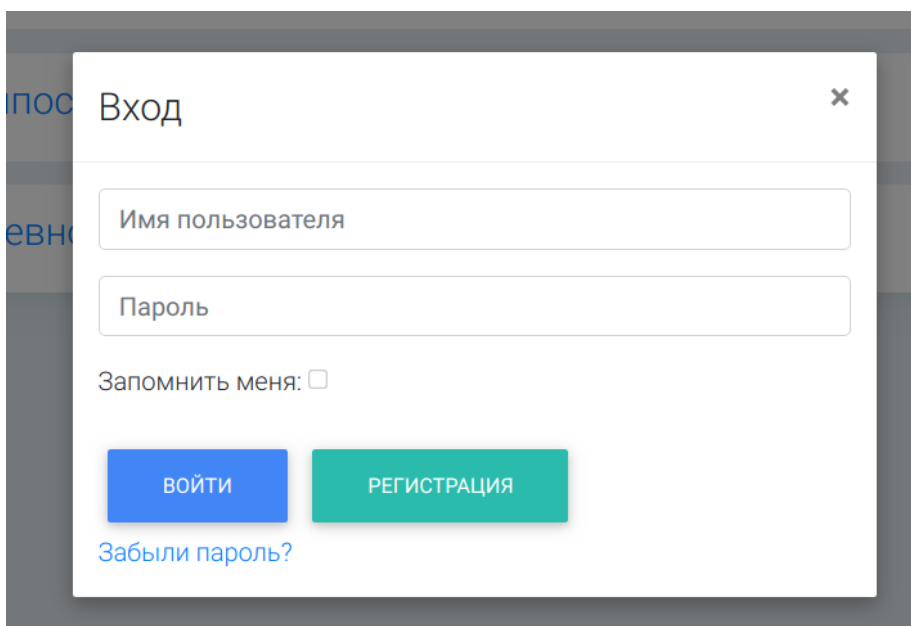
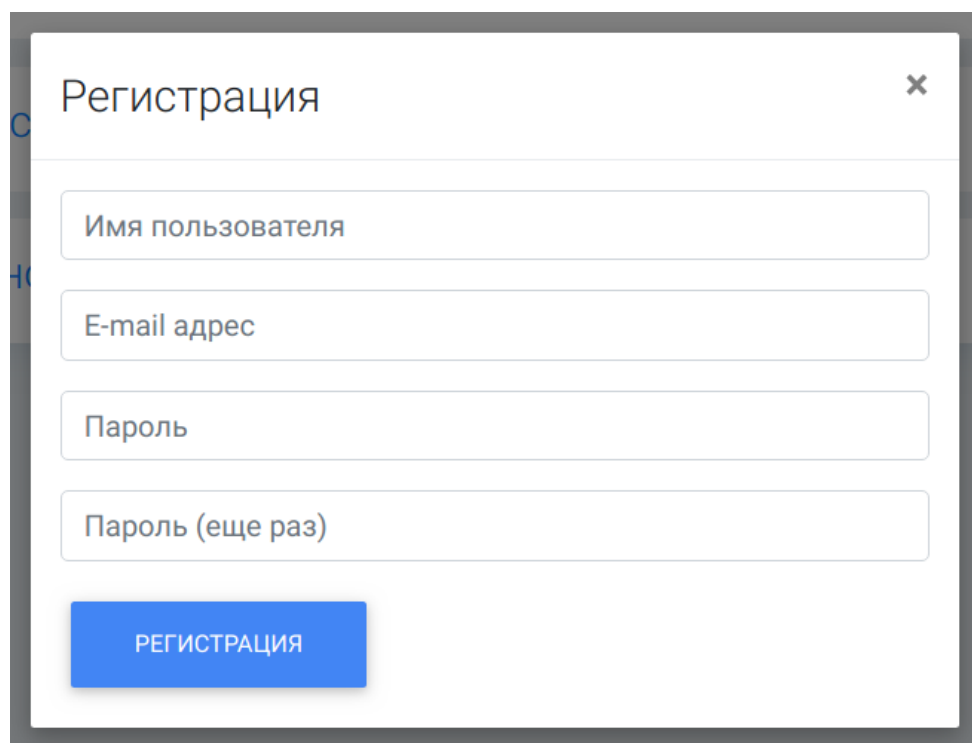


Рисунок 24 - Модальное окно для авторизации

A registration modal window with a title bar containing the word "Регистрация" and a close button (X). The form contains four input fields: "Имя пользователя", "E-mail адрес", "Пароль", and "Пароль (еще раз)". Below the fields is a blue button labeled "РЕГИСТРАЦИЯ".

Регистрация

Имя пользователя

E-mail адрес

Пароль

Пароль (еще раз)

РЕГИСТРАЦИЯ

Рисунок 25 - Модальное окно для регистрации

Авторизованные пользователи имеют возможность выхода из системы. В этом случае они становятся «гостями» и имеют возможность только просматривать существующие выставки. Пример представлен на рисунке 26, на котором показано, что у гостей нет никаких других прав.

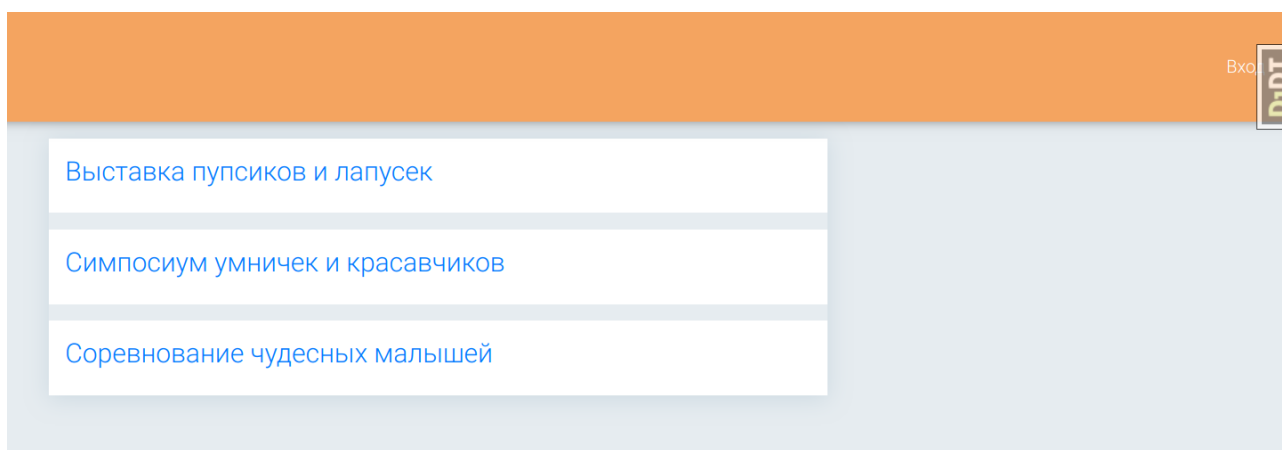


Рисунок 26 - Главная страница для гостей с возможностью входа

4. ОПИСАНИЕ СРЕДСТВ РАЗРАБОТКИ

Для реализации web-сервиса были применены следующие средства разработки: язык программирования Python, фреймворк Django и Django REST Framework , Vue JS, HTML, CSS и docker.

Django — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Django использует шаблон проектирования MVC (model-view-controller), который разделяет логику приложения на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Серверная сторона была реализована с помощью библиотеки Django – Django REST Framework.

Django REST Framework – это библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта. API в Django REST Framework состоит из 3-х слоев: сериализатора, который преобразует информацию, хранящуюся в базе данных и определенную с помощью моделей Django; представление, которое определяет функции (чтение, создание, обновление, удаление), которые будут доступны через API и маршрутизатора, который определяет URL-адреса, которые будут предоставлять доступ к каждому представлению.

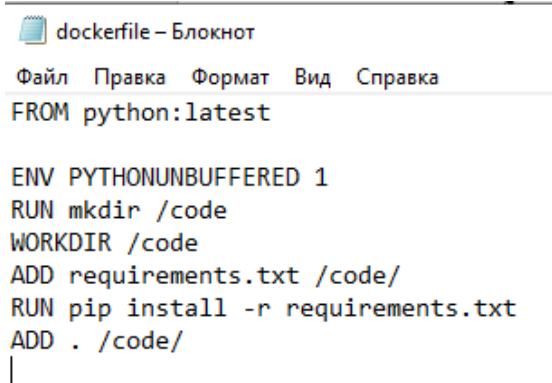
Для реализации клиентской стороны были использован язык разметки HTML, язык описания CSS, и фреймворк Vue JS.

Vue JS – JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Подобно другим JavaScript-фреймворкам, Vue.js позволяет разделить приложение или страницу на логические единицы, которые могут применяться как переиспользуемые компоненты. Каждый из таких компонентов может содержать внутри себя HTML, CSS и JavaScript-код, присущий только этому компоненту и отвечающий за работоспособность данного компонента.

Для обеспечения спорки проекта была реализована докеризация с использованием средства Docker.

Docker является инструментом с открытым кодом, который автоматизирует разворачивание приложения внутри программного контейнера. Все процессы при контейнеризации протекают на уровне операционной системы, что позволяет существенно экономить ресурсы и увеличивать эффективность работы с приложениями.

Для создания и запуска контейнера с помощью Docker были реализованы два файла: `dockerfile` и `docker-compose`. `Dockerfile` – файл, в котором прописываются пути и названия виртуальных папок контейнера, а также происходит установка важных частей проекта – различных библиотек, без которых работа с веб-сервисом невозможна. Эти требования описаны в файле `requirements.txt`. На рисунке 27 изображено содержимое файла `dockerfile`:

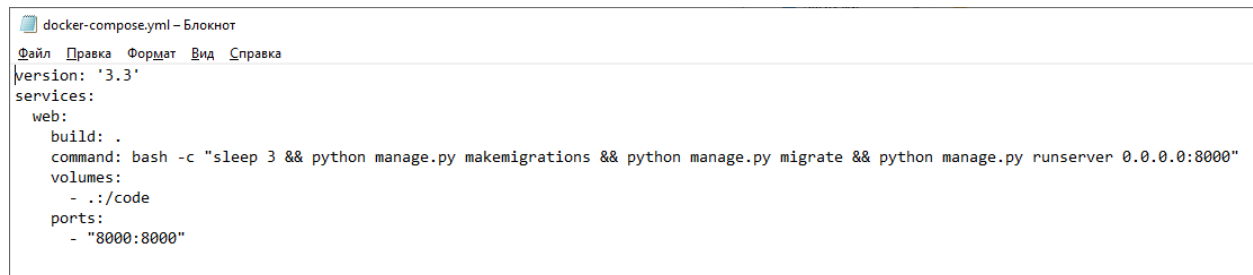
A screenshot of a text editor window titled "dockerfile - Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The content of the file is as follows:

```
FROM python:latest

ENV PYTHONUNBUFFERED 1
RUN mkdir /code
WORKDIR /code
ADD requirements.txt /code/
RUN pip install -r requirements.txt
ADD . /code/
```

Рисунок 27 - содержимое файла `dockerfile`

`Docker Compose` используется для одновременного управления несколькими контейнерами, входящими в состав приложения. Файл `docker-compose` для курсовой работы представлен на рисунке 28.

A screenshot of a text editor window titled "docker-compose.yml - Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The content of the file is as follows:

```
version: '3.3'
services:
  web:
    build: .
    command: bash -c "sleep 3 && python manage.py makemigrations && python manage.py migrate && python manage.py runserver 0.0.0.0:8000"
    volumes:
      - ../code
    ports:
      - "8000:8000"
```

Рисунок 28 - содержимое файла `docker-compose`

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы были получены навыки разработки веб-сервиса с выполнением всех этапов разработки: была выбрана и проанализирована предметная область, выявлены функциональные требования, которыми должен ответить веб-сервис, была спроектирована и реализована модель базы данных, были реализованы серверная и клиентская сторона веб-системы. Помимо этого, был проведен этап тестирования для выявления ошибок и исправления некорректной работы системы.

В ходе выполнения курсовой работы были применены навыки и умения, полученные при выполнении практических и лабораторных работ по курсу «Основы Web-программирования». В частности, были изучены и опробованы на практике такие технологии как Django REST Framework, совместно с Django API и Vue JS. Результатом практической деятельности в ходе курса и выполнения курсовой работы является спроектированный и реализованный веб-сервис для организаторов выставок собак.

СПИСОК ЛИТЕРАТУРЫ

1. Дронов В. Django 2.1. Практика создания веб-сайтов на Python, 2019.
2. Винсент, В. Django for APIs: Build web APIs with Python and Django, 2018.
3. Документация Vue.js [Электронный ресурс]. – Режим доступа: <https://ru.vuejs.org/v2/guide/>. – Дата доступа: 20.06.2020.
4. Хэнчетт, Л. Vue.js в действии, 2019
5. Элман, Д. Lightweight Django / Д. Элман, М. Лэвин, 2015