

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

Факультет: Инфокоммуникационных технологий

Направление подготовки (специальность): 09.03.03 «Мобильные и сетевые технологии»

ОТЧЕТ

Тема задания: Реализация простого сайта средствами DJANGO

Выполнил: Дубина С.Д. К33402
(Фамилия И.О.) (номер группы)

Проверил: Говоров А.И.
(Фамилия И.О.)

Дата _____

Санкт-Петербург
2019 г.

Цель: овладеть практическими навыками и умениями реализации webсервисов средствами Django 2.2.

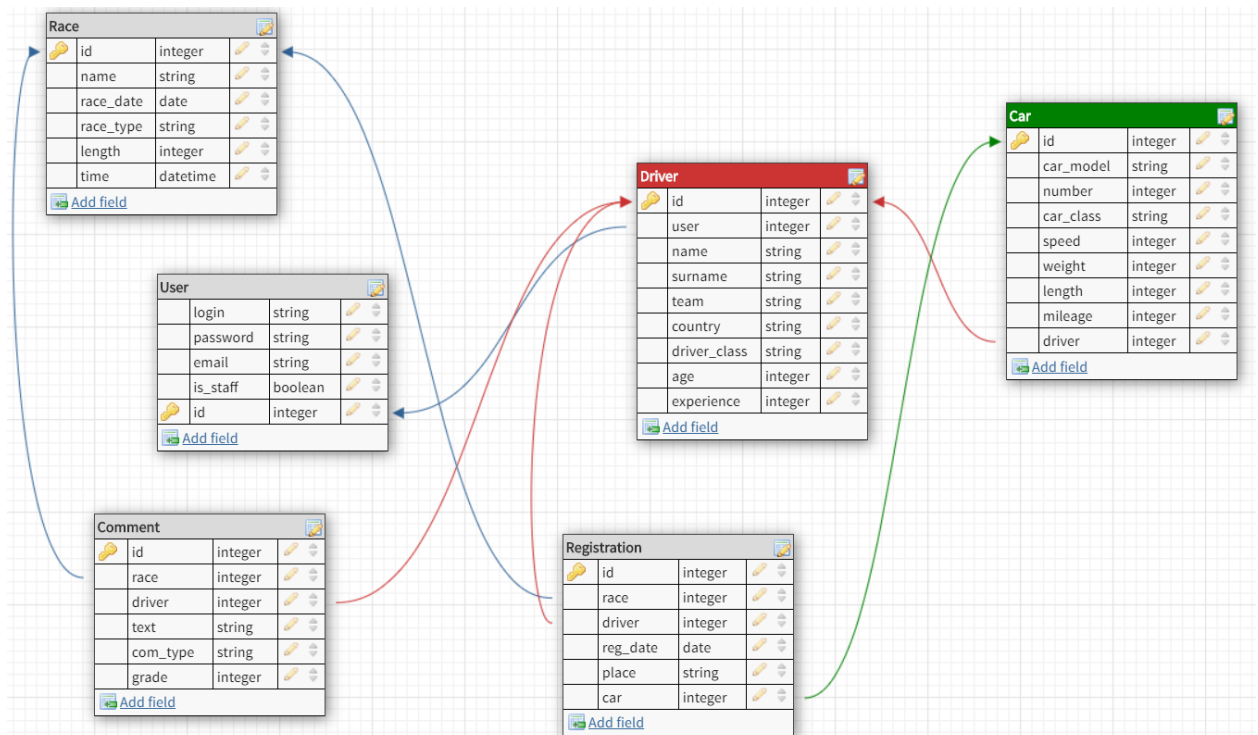
Описание варианта №6 (Табло победителей автогонок):

Табло должно отображать информацию об участниках автогонок: ФИО участника, название команды, описание автомобиля, описание участника, опыт и класс участника.

Необходимо реализовать следующий функционал:

- ☐ Регистрация новых пользователей.
- ☐ Просмотр автогонок и регистрацию гонщиков. Пользователь должен иметь возможность редактирования и удаления своих регистраций.
- ☐ Написание отзывов и комментариев к автогонкам. Предварительно комментатор должен зарегистрироваться. При добавлении комментариев должны сохраняться даты заезда, текст комментария, тип комментария (вопрос о сотрудничестве, вопрос о гонках, иное), рейтинг (1-10), информация о комментаторе.
- ☐ Администратор должен иметь возможность указания времени заезда и результата средствами Django-admin.
- ☐ В клиентской части должна формироваться таблица всех заездов и результатов конкретной гонки.

Модель базы данных:



Описание модели данных (файл models.py):

```
from django.db import models
from datetime import datetime
import django
from django.contrib.auth.models import User

class Driver(models.Model):
    COUNTRIES = (
        ('ES', 'Spain'),
        ('IT', 'Italy'),
        ('FR', 'France'),
        ('DE', 'Deutschland'))

    RACER_TYPES = (
        ('Drag', 'Drag racer'),
        ('Sport', 'Sports car racer'),
        ('Off', 'Off-road racer'))

    user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
    name = models.CharField(max_length=200, null=True)
    surname = models.CharField(max_length=200, null=True)
    team = models.CharField(max_length=200, null=True)
    country = models.CharField(max_length=2, choices=COUNTRIES, null=True)
    driver_class = models.CharField(max_length=5, choices=RACER_TYPES, null=True)
    age = models.IntegerField(null=True)
    experience = models.IntegerField(null=True)

    def __str__(self):
        return '{} {}'.format(self.name, self.surname)
```

```
class Car(models.Model):
    CAR_TYPES = (
        ('Drag', 'Drag'),
        ('Sport', 'Sportscar'),
        ('Off', 'Off-road'))

    car_model = models.CharField(max_length=200, null=True)
    number = models.IntegerField(null=True, unique=True)
    car_class = models.CharField(max_length=200, choices=CAR_TYPES, null=True)
    speed = models.IntegerField(null=True)
    weight = models.FloatField(null=True)
    length = models.FloatField(null=True)
    mileage = models.IntegerField(null=True)
    driver = models.ForeignKey(Driver, null=True, on_delete=models.SET_NULL)

    def __str__(self):
        return '{} {}'.format(self.car_model, self.number)

class Race(models.Model):
    RACE_TYPES = (
        ('Drag', 'Drag racing'),
        ('Sport', 'Sports car racing'),
        ('Off', 'Off-road racing'))

    race_date = models.DateField(null=True)
    race_type = models.CharField(max_length=200, choices=RACE_TYPES, null=True)
    length = models.IntegerField(null=True)
    name = models.CharField(max_length=200, null=True)
    registrations = models.ManyToManyField(Driver, through='Registration', blank=True)
    time = models.TimeField(null=True, blank=True)

    def __str__(self):
        return '{} - {}'.format(self.name, self.id)

class Comment(models.Model):
    TYPE = (
        ('Cooperation', 'Cooperation'),
        ('Race', 'Race'),
        ('Other', 'Other'))

    RATE = (
        ('1', '1'),
        ('2', '2'),
        ('3', '3'),
        ('4', '4'),
        ('5', '5'))

    race = models.ForeignKey(Race, null=True, on_delete=models.SET_NULL)
    driver = models.ForeignKey(Driver, null=True, on_delete=models.SET_NULL)
    text = models.TextField()
    com_type = models.CharField(max_length=11, choices=TYPE, null=True)
    grade = models.CharField(max_length=1, choices=RATE, null=True)

    def __str__(self):
        return '{}: {}'.format(self.driver, self.com_type)
```

```

class Registration(models.Model):
    race = models.ForeignKey(Race, null=True, on_delete=models.SET_NULL)
    driver = models.ForeignKey(Driver, null=True, on_delete=models.SET_NULL)
    reg_date = models.DateField(null=True, blank=True, default = django.utils.timezone.now)
    place = models.IntegerField(null=True, blank=True)
    car = models.ForeignKey(Car, null=True, on_delete=models.SET_NULL)

    def __str__(self):
        return '{} {}'.format(self.race, self.driver)

```

Описание контроллеров (файл views.py)

```

from django.shortcuts import render, redirect
from django.http import HttpResponse
from .models import *
from .forms import *
from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from .decorators import unauthenticated_user, allowed_users
from django.contrib.auth.models import Group
from django.forms import inlineformset_factory

@unauthenticated_user
def regPage(request):
    form = CreateUserForm()
    if request.method == "POST":
        form = CreateUserForm(request.POST)
        if form.is_valid():
            user = form.save()

            group = Group.objects.get(name='driver')
            user.groups.add(group)
            Driver.objects.create(user=user)

            messages.success(request, 'Account was created')
            return redirect('driver_cr')

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

@unauthenticated_user
def loginPage(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('/races')
        else:
            messages.info(request, 'Username or password is not correct')

    context = {}
    return render(request, 'login_page.html', context)

```

```

def logoutUser(request):
    logout(request)
    return redirect('/login')

@login_required(login_url='login')
@allowed_users(allowed_roles=['driver'])
def home(request):
    regs = request.user.driver.registration_set.all()
    driver = Driver.objects.get(id=request.user.driver.id)
    # cars = request.user.driver.car_set.all()
    cars = Car.objects.filter(driver = Driver.objects.get(id=request.user.driver.id))

    return render(request, 'home.html', {'regs': regs, 'cars':cars, 'driver':driver})

@login_required(login_url='login')
def drivers(request, pk_test):
    drivers = []
    cars = []

    registrations = Registration.objects.filter(race = pk_test).order_by('place')
    for reg in registrations:
        drivers.append(reg.driver)
        cars.append(reg.car)
    print(drivers)

    return render(request, 'drivers.html', {'drivers':drivers, 'cars':cars, 'regs':registrations})

@login_required(login_url='login')
def races(request):
    context = Race.objects.all()
    return render(request, 'races.html', {'races': context})

```

```

@login_required(login_url='login')
def createReg(request):
    driver = Driver.objects.get(id=request.user.driver.id)
    car = Car.objects.get(driver=driver)

    # if Registration.objects.get(driver=driver, race=race):
    #     messages.error(request, 'You are already regestered to this race')
    #     return redirect('/home')

    if car.car_model == None:
        messages.error(request, 'You need a car to register')
        return redirect('/home')

    form = RegForm(initial={'driver':driver, 'car':car})
    if request.method == "POST":
        print(request.POST)
        form = RegForm(request.POST)
        race_test = Race.objects.get(id=request.POST.get('race'))
        if Registration.objects.filter(driver=driver, race=race_test):
            messages.error(request, 'You are already regestered to this race')
            return redirect('/home')
        if form.is_valid():
            reg = form.save()
            messages.success(request, 'You were registered')
            return redirect('/home')

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

@login_required(login_url='login')
def raceReg(request, pk):
    driver = Driver.objects.get(id=request.user.driver.id)
    race = Race.objects.get(id=pk)
    if Registration.objects.filter(driver=driver, race=race):
        messages.error(request, 'You are already regestered to this race')
    else:
        car = Car.objects.get(driver=driver)
        if car.car_model != None:
            Registration.objects.create(driver=driver, race=race, car=car)
            messages.success(request, 'You were registered')
        else:
            messages.error(request, 'You need a car to register')

    return redirect('/races')

```

```

@login_required(login_url='login')
def changeReg(request, pk):
    reg = Registration.objects.get(id=pk)
    form = RegForm(instance=reg)
    if request.method == "POST":
        form = RegForm(request.POST, instance=reg)
        if form.is_valid():
            form.save()
            return redirect('/home')
    context = {'form':form}
    return render(request, 'reg_form.html', context)

def regDriver(request):
    driver = Driver.objects.get(name=None)
    form = CreateDriverForm(instance=driver)
    if request.method == "POST":
        form = CreateDriverForm(request.POST, instance=driver)
        if form.is_valid():
            driver = form.save()
            Car.objects.create(driver=driver)
            messages.success(request, 'Driver was created')
            return redirect('login')

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

@login_required(login_url='login')
def deleteReg(request, pk):
    reg = Registration.objects.get(id=pk)
    context = {'reg':reg}
    if request.method == "POST":
        reg.delete()
        return redirect('/home')

    return render(request, 'del_reg.html', context)

```

```

@login_required(login_url='login')
def createCar(request):
    driver = Driver.objects.get(id=request.user.driver.id)
    car = Car.objects.get(driver=driver)
    form = CarForm(instance=car)

    if request.method == "POST":
        form = CarForm(request.POST, instance=car)
        if form.is_valid():
            car = form.save()
            return redirect('/home')
        messages.success(request, 'You successfully registered your car')

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

@login_required(login_url='login')
def changeCar(request, pk):
    car = Car.objects.get(id = pk)
    form = CarForm(instance = car)

    if request.method == "POST":
        form = CarForm(request.POST, instance=car)
        if form.is_valid():
            form.save()
            return redirect('/home')

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

@login_required(login_url='login')
def writeComment(request, pk):
    race = Race.objects.get(id = pk)
    driver = Driver.objects.get(id=request.user.driver.id)
    form = CommentForm(initial={'race':race, 'driver':driver})

    if request.method == 'POST':
        form = CommentForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'You left a comment for admin. Wait for an answer')
            return redirect('/races')

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

```

```

@login_required(login_url='login')
def createRace(request):

    form = RaceForm()
    if request.method == "POST":
        form = RaceForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'Race was created')
            return redirect('/races')

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

@login_required(login_url='login')
def changeRace(request, pk):
    race = Race.objects.get(id = pk)
    form = RaceForm(instance = race)

    if request.method == "POST":
        form = RaceForm(request.POST, instance=race)
        if form.is_valid():
            form.save()
            return redirect('/races')

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

```

```

@login_required(login_url='login')
def results(request, pk):
    raceFormSet = inlineformset_factory(Race, Registration, fields=('driver', 'place'), extra=0)
    race = Race.objects.get(id = pk)
    regs = race.registration_set.count()
    formset = raceFormSet(instance=race)
    if request.method == 'POST':
        formset = raceFormSet(request.POST, instance = race)
        if formset.is_valid():
            formset.save()
            return redirect('/races')

    context = {'formset': formset, 'regs': regs}
    return render(request, 'race_res.html', context)

@login_required(login_url='login')
def comments(request, pk):
    race = Race.objects.get(id = pk)
    comments = race.comment_set.all()
    context = {'comments': comments}

    return render(request, 'comments.html', context)

@login_required(login_url='login')
def delRace(request, pk):
    race = Race.objects.get(id=pk)
    context = {'race': race}
    if request.method == "POST":
        race.delete()
        return redirect('/races')

    return render(request, 'del_race.html', context)

```

Описание роутеров (файл urls.py)

url проекта:

```
from django.contrib import admin
from django.urls import path, include

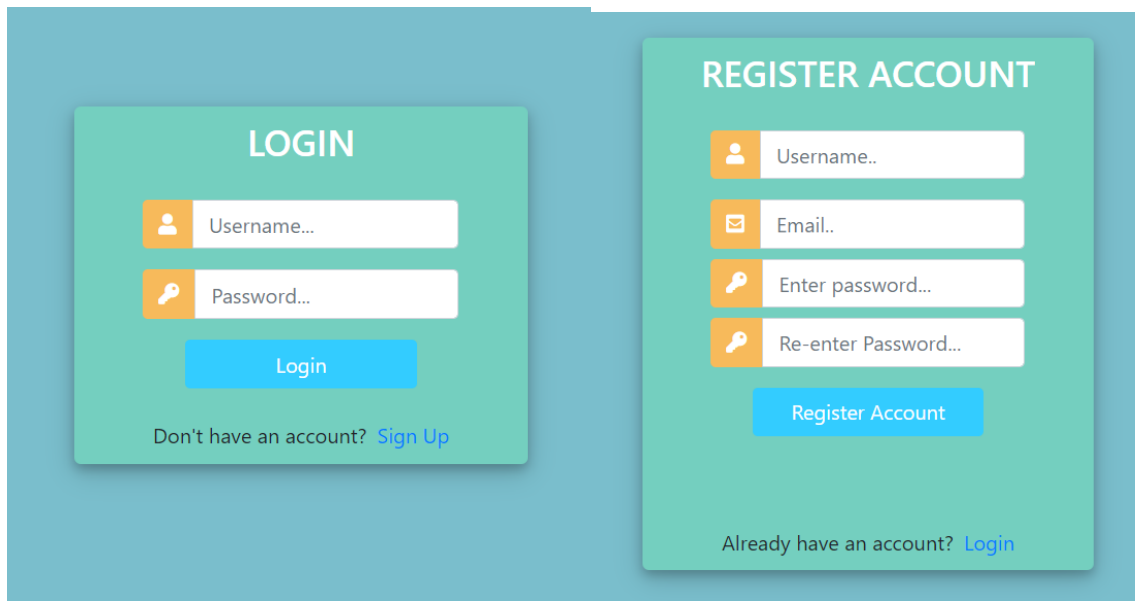
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('races.urls')),
]
```

url приложения:

```
from django.contrib import admin
from django.urls import path
from django.http import HttpResponseRedirect
from . import views

urlpatterns = [
    path('', views.races),
    path('races/<int:pk_test>/drivers', views.drivers, name="race"),
    path('races/', views.races, name="races"),
    path('home/', views.home, name="home"),
    path('registration/create', views.createReg, name="registration_cr"),
    path('registration/<int:pk>/change', views.changeReg, name="registration_ch"),
    path('registration/<int:pk>/delete', views.deleteReg, name="registration_dl"),
    path('car/create', views.createCar, name="car_cr"),
    path('car/<int:pk>/change', views.changeCar, name="car_ch"),
    path('register', views.regPage, name="register"),
    path('login', views.loginPage, name="login"),
    path('logout', views.logoutUser, name="logout"),
    path('driver_cr', views.regDriver, name="driver_cr"),
    path('race/<int:pk>/registrate', views.raceReg, name="race_reg"),
    path('race/<int:pk>/comment', views.writeComment, name="comment"),
    path('race/create', views.createRace, name="race_cr"),
    path('race/<int:pk>/change', views.changeRace, name="race_ch"),
    path('race/<int:pk>/results', views.results, name="results"),
    path('race/<int:pk>/comments', views.comments, name="comments"),
    path('race/<int:pk>/delete', views.delRace, name="race_dl"),
]
```


Страницы регистрации и авторизации:



The image shows two side-by-side login and registration forms on a teal background. The left form is titled 'LOGIN' and contains fields for 'Username...' and 'Password...' with a 'Login' button below. A link 'Don't have an account? Sign Up' is at the bottom. The right form is titled 'REGISTER ACCOUNT' and contains fields for 'Username..', 'Email..', 'Enter password...', and 'Re-enter Password...' with a 'Register Account' button below. A link 'Already have an account? Login' is at the bottom.

LOGIN

Username...

Password...

Login

Don't have an account? [Sign Up](#)

REGISTER ACCOUNT

Username..

Email..

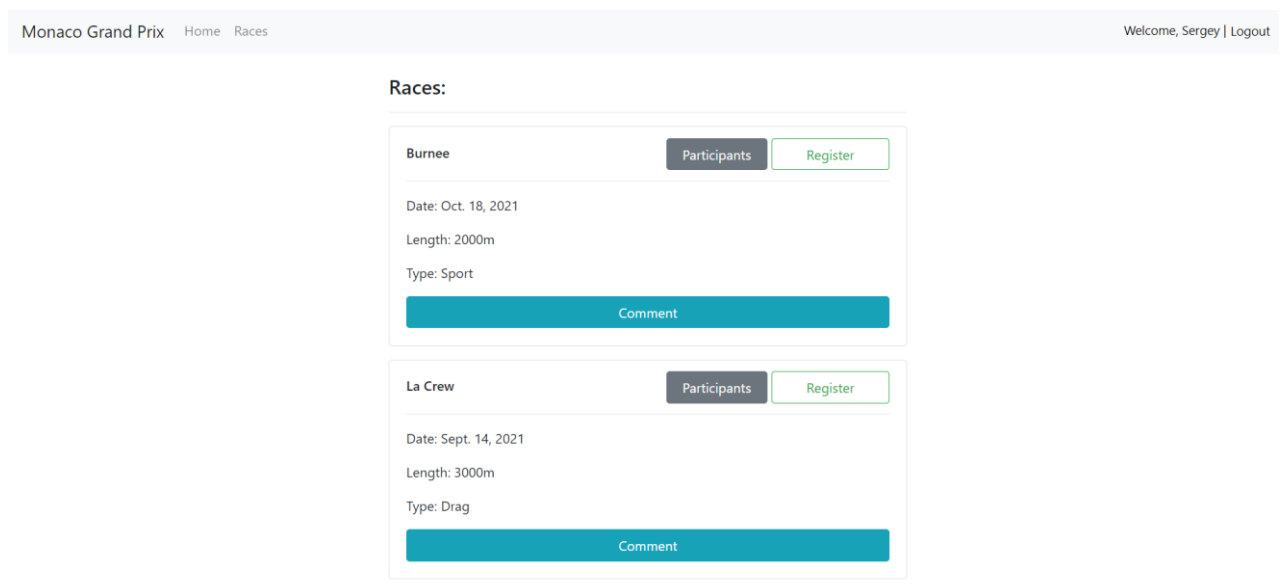
Enter password...

Re-enter Password...

Register Account

Already have an account? [Login](#)

Страница со всеми гонками:



The image shows a web page for 'Monaco Grand Prix' with a header containing 'Home' and 'Races' links, and a user greeting 'Welcome, Sergey | Logout'. The main content is titled 'Races:' and lists two races: 'Burnee' and 'La Crew'. Each race card includes a 'Participants' button, a 'Register' button, and a 'Comment' button. The 'Burnee' race is dated 'Oct. 18, 2021', has a length of '2000m', and is of type 'Sport'. The 'La Crew' race is dated 'Sept. 14, 2021', has a length of '3000m', and is of type 'Drag'.

Monaco Grand Prix Home Races Welcome, Sergey | Logout

Races:

Burnee Participants Register

Date: Oct. 18, 2021

Length: 2000m

Type: Sport

Comment

La Crew Participants Register

Date: Sept. 14, 2021

Length: 3000m

Type: Drag

Comment

Страница участников завершённой гонки

Monaco Grand Prix

Home

Races

Welcome, Sergey | Logout

Drivers:

Kir Lolaev

#1

Team: Best team

Age: 40

Country: ES

Experienece: 20

Class: Sport

Sergey Dubina

#2

Team: Dogs

Age: 40

Country: IT

Experienece: 19

Cars:

Mercedes 3789

Speed: 200km/h

Class: Off

Weight: 234.0kg

Length: 2.4m

Mileage: 2990km

BMV 6 2134

Speed: 125km/h

Class: Drag

Weight: 2134.0kg

Length: 1234.0m

Страница добавления комментария

Write a comment:

Sergey Dubina

Burnee - 1

Text:

Topic

Grade

Submit

Домашняя страница гонщика

Monaco Grand Prix

[Home](#) [Races](#)

Welcome, Sergey | [Logout](#)

Register +

Your registrations:

"Burnee" - Sport

(Oct. 18, 2021)

Edit

Delete

"La Crew" - Drag

(Sept. 14, 2021)

Edit

Delete


Your car:


"BMV 6" - 2134


Edit


Страницы редактирования машины и регистрации на гонку


Write car info:


BMW 6


2134

125

2134,0


1234,0


2000


Drag ▾

Submit

Race registration

Sergey Dubina ▾

Burnee - 1 ▾

BMW 6 2134 ▾

Register

Страница со списком гонок для администратора

Monaco Grand Prix RacesWelcome, serjd | Logout

Race +

Races:

Burnee

ResultsEditDelete

Date: Oct. 18, 2021

Length: 2000m

Type: Sport

Comments

La Crew

ResultsEditDelete

Date: Sept. 14, 2021

Length: 3000m

Type: Drag

Comments

Страницы добавления и редактирования гонки

Write race info

Name..

Date..

Length..

----- ▾

Confirm

Write race info

Burnee

2021-10-18

2000

Sports car racing ▾

Change Race

Страница с комментариями гонки

Monaco Grand Prix Races Welcome, serjd | Logout

Comments:

Max Alilov Race

Text: Гонка класс!

Grade: 4

Max Alilov Other

Text: Повторюсь, очень круто!!!!

Grade: 5

Sergey Dubina Cooperation

Text: Давай работать вместе

Grade: 4

Страница для указания результатов гонки

Set results of race:

Driver:

Sergey Dubina ▾

Place:

Delete:

Driver:

Max Alilov ▾

Place:

Delete:

Confirm

Выводы: в результате выполнения лабораторной работы были получены практические навыки и умения реализации web-серверов средствами Django; создан простой сайт для туристической фирмы.