ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет «Инфокоммуникационных технологий»
Направление подготовки «09.03.03 Мобильные и сетевые технологии»

О Т Ч Е Т

Тема задания: РЕАЛИЗАЦИЯ ПРОСТОГО САЙТА СРЕДСТВАМИ DJANGO

Выполнил:
Студент Малинина А.В _K33402
(Фамилия И.О.) номер группы

Проверил:
Преподаватель Говоров А.И
(Фамилия И.О)

Санкт-Петербург
2020

Вариант 6.

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.

Реализованные интерфейсы.

Реализованы 4 вида пользователей: администратор, гонщик, зарегистрированный посетитель сайта, посетитель.

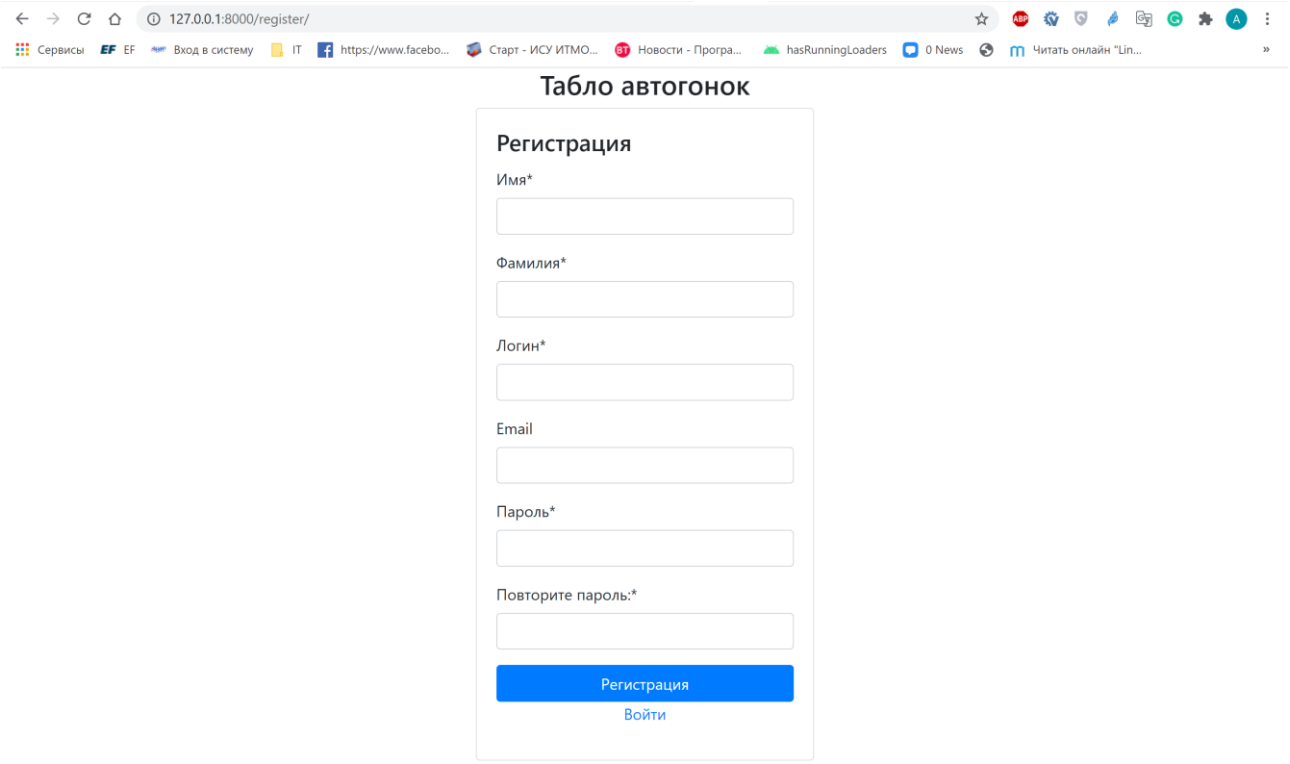
Администратор имеет доступ к Django admin и имеет возможность напрямую работать с базой данных.

Гонщик – зарегистрированный пользователь, который так же зарегистрировался как гонщик. Имеет возможность добавлять и удалять регистрации на автогонки, изменять информацию о себе, как о гонщике.

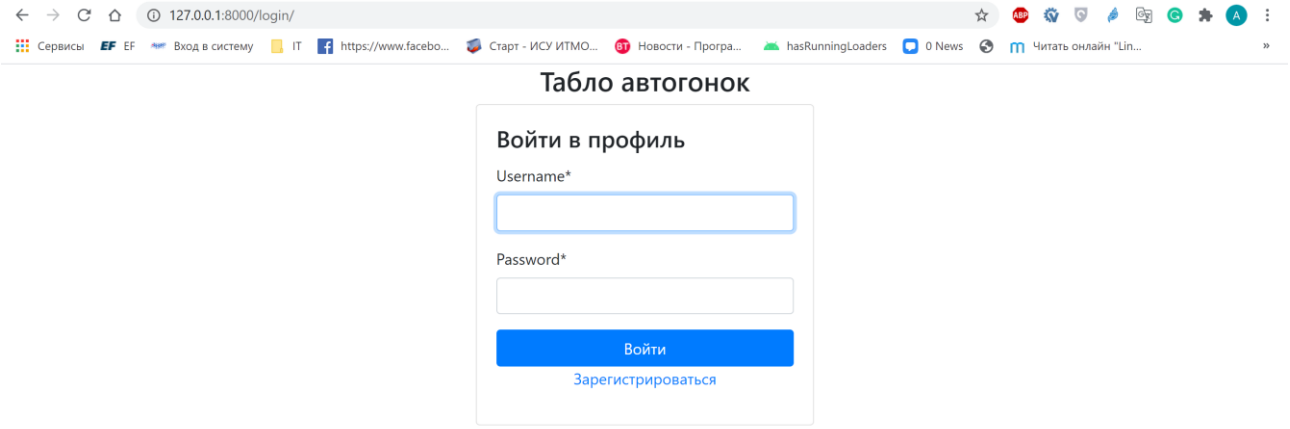
Зарегистрированный пользователь имеет возможность просматривать и комментировать автогонки, зарегистрироваться как гонщик.

Посетитель имеет возможность просматривать автогонки и зарегистрироваться.

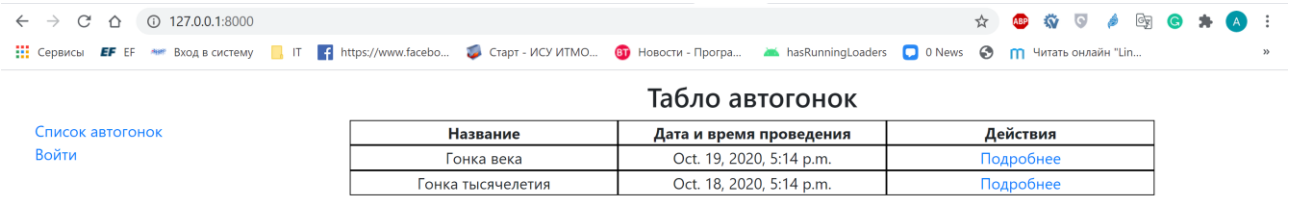
Экран регистрации:



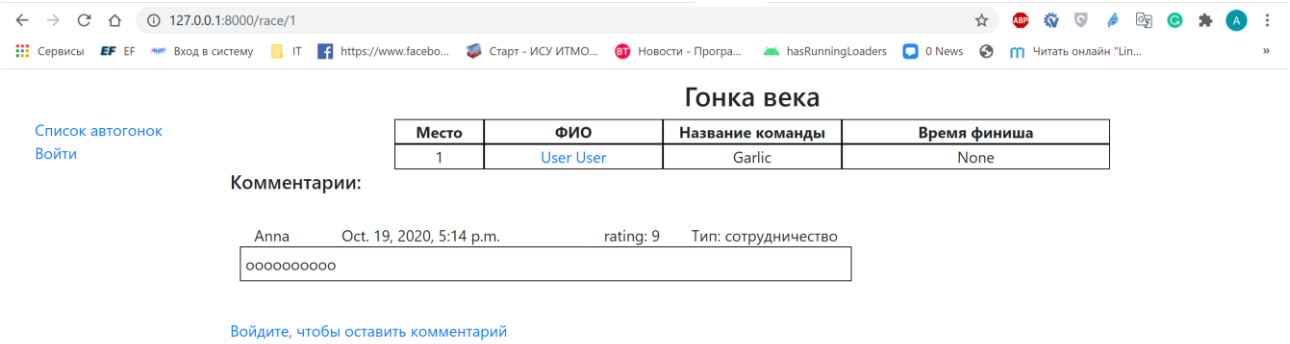
Экран входа:



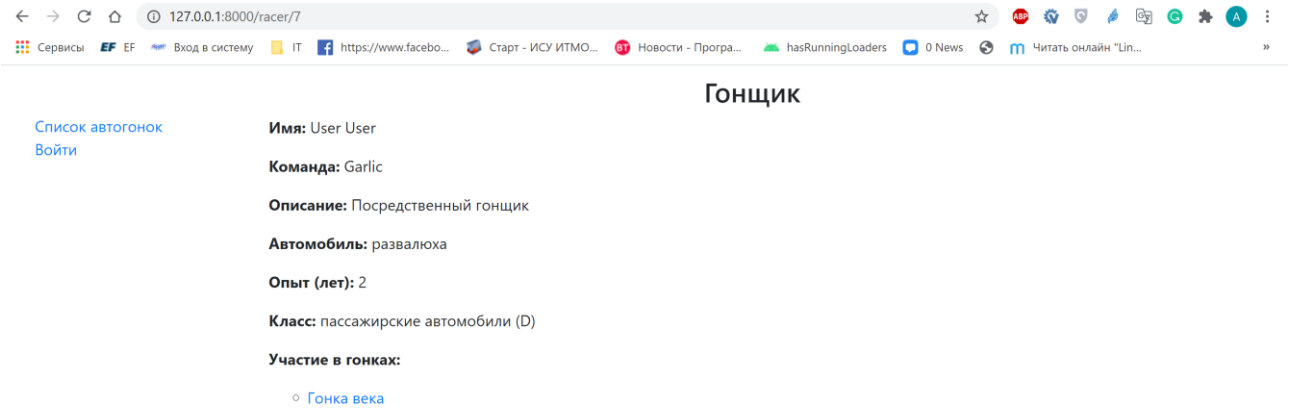
Главный экран (вид незарегистрированного пользователя)



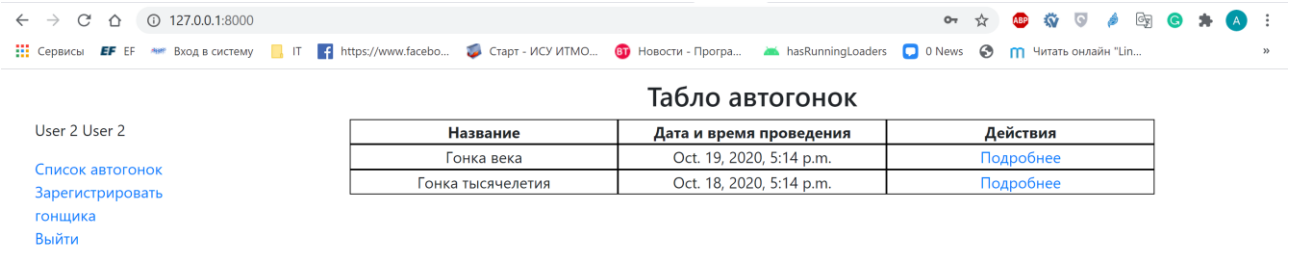
Экран гонки (вид незарегистрированного пользователя)



Экран информации о гонщике:

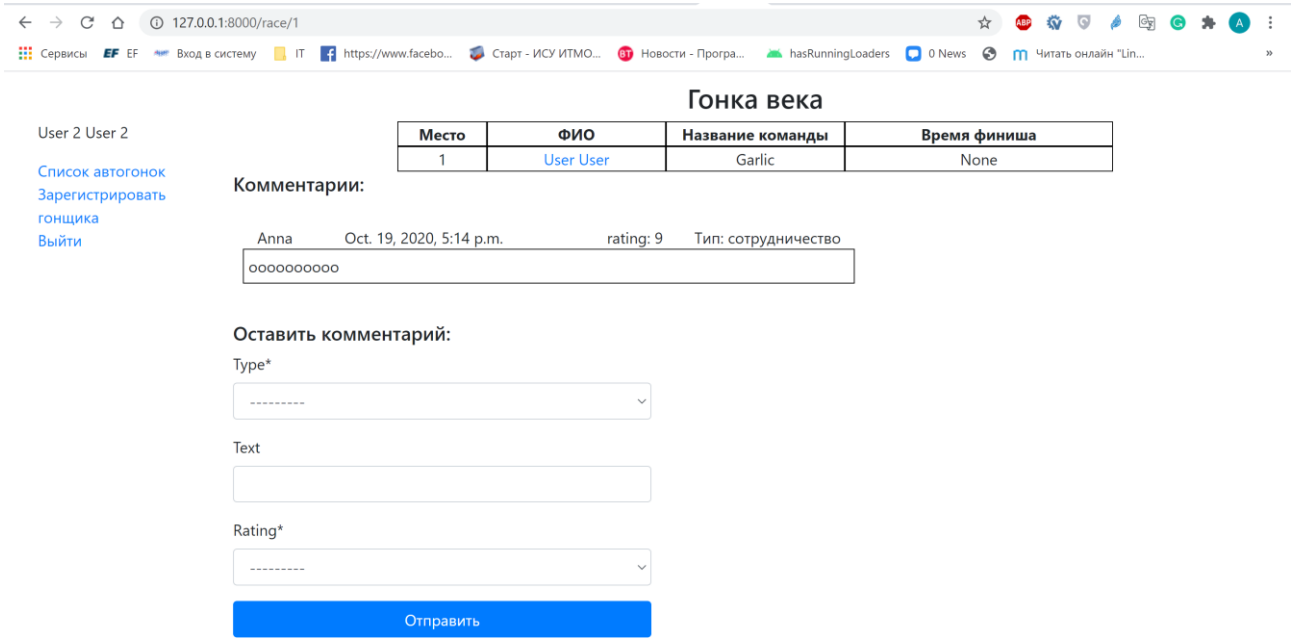


Главный экран (зарегистрированный пользователь, не гонщик)



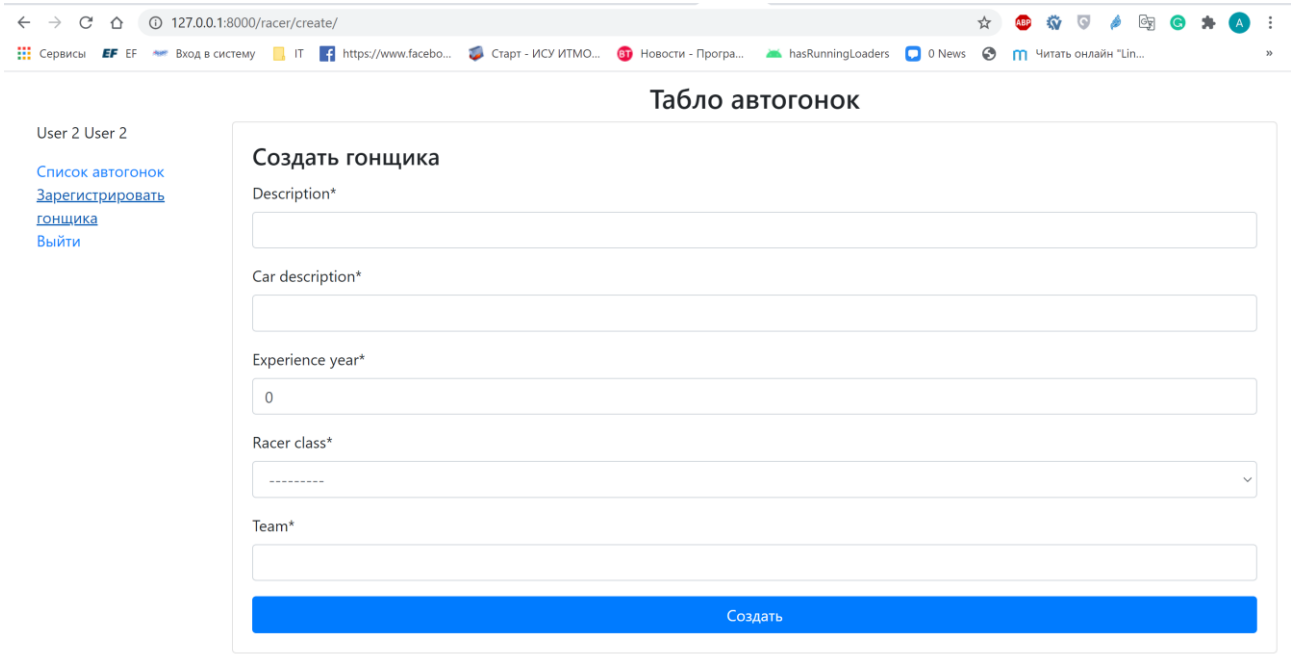
Примечание: появилась слева возможность зарегистрироваться как гонщик.

Экран гонки (зарегистрированный пользователь)

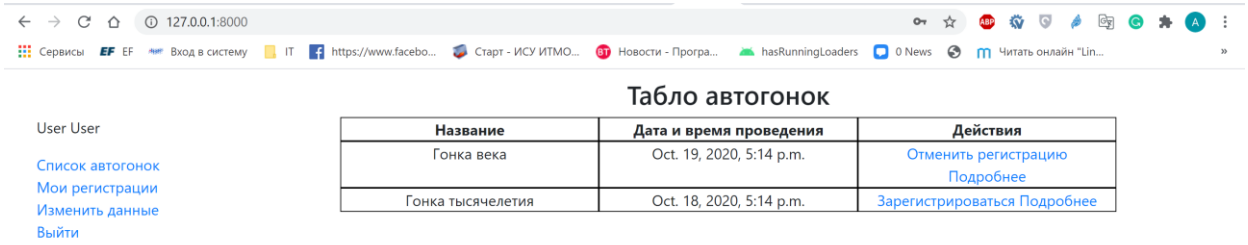


Примечание: появились возможность оставить комментарий.

Экран регистрации как гонщика

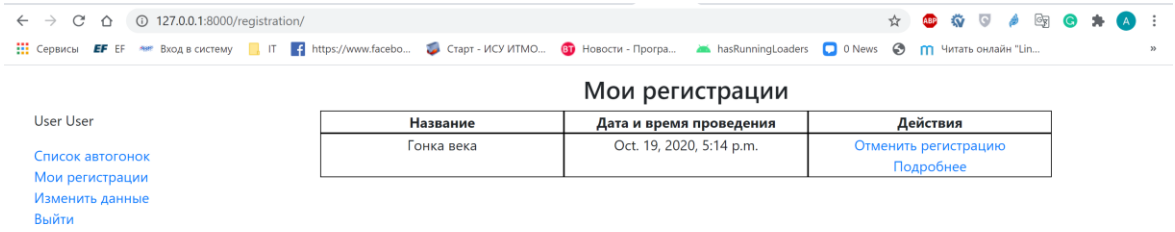


Главный экран (гонщик)

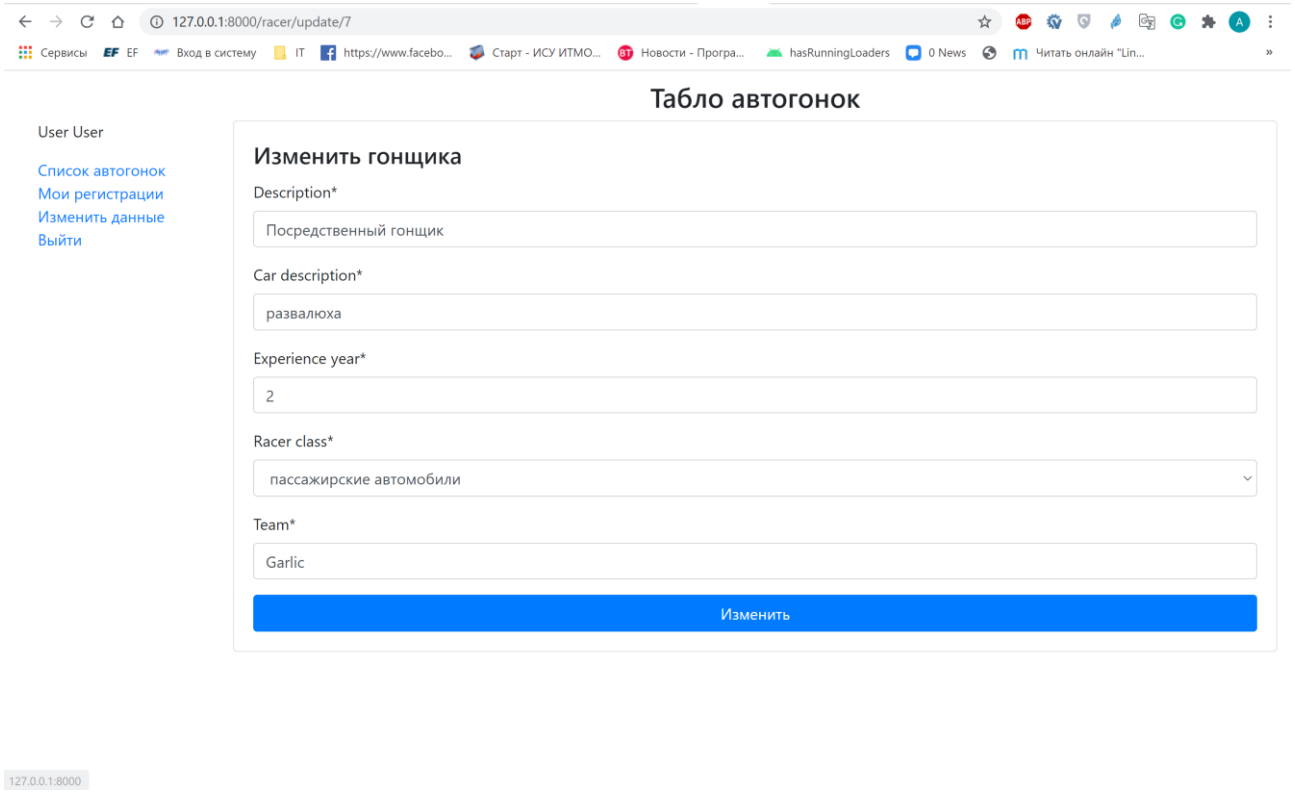


Примечание: появилась возможность регистрироваться или отменять регистрацию на гонку (если зарегистрирован), просматривать регистрации, изменять данные.

Экран моих регистраций (гонщик)



Экран изменения данных



Файл views.py

```
from django.contrib.auth import get_user_model
from django.contrib.auth.forms import UserCreationForm
from django.db.models import F
from django.shortcuts import render, redirect

# Create your views here.
from django.urls import reverse
from django.views.generic import FormView, TemplateView, ListView, DetailView,
CreateView, UpdateView

from racing_scoreboard.forms import RegisterForm, AddCommentForm,
CreateRacerForm
from racing_scoreboard.models import Race, Comment, Racer, RaceRacer

User = get_user_model()


# Регистрация пользователя
def register(request):
    register_form = RegisterForm(request.POST or None)
    context = {
        "register_form": register_form
    }

    if register_form.is_valid():
        username = register_form.cleaned_data.get("username")
        first_name = register_form.cleaned_data.get("first_name")
        last_name = register_form.cleaned_data.get("last_name")
```

```
        email = register_form.cleaned_data.get("email")
        password = register_form.cleaned_data.get("password")

        User.objects.create_user(username, email, password,
first_name=first_name, last_name=last_name)

        return redirect('/login/')

    return render(request, "registration/register.html", context)


# Вывод информации об автогонках (плюс регистрация и удаление регистрации на
гонку)
class RaceListView(ListView):
    template_name = 'main.html'
    model = Race

    def get_queryset(self):
        register_id = self.request.GET.get('register_id')
        delete_id = self.request.GET.get('delete_id')

        if register_id:
            try:
                register_id = int(register_id)
                RaceRacer.objects.create(race_id=register_id,
racer_id=self.request.user.racer.id)

            except:
                pass

        if delete_id:
            try:
                delete_id = int(delete_id)
                RaceRacer.objects.get(race_id=delete_id,
racer_id=self.request.user.racer.id).delete()

            except:
                pass

        return Race.objects.all()


# Детали гонки
class RaceDetailView(DetailView, CreateView):
    template_name = 'race_detail.html'
    model = Race
    form_class = AddCommentForm

    def get_success_url(self):
        return reverse('race-detail', args=[str(self.kwargs.get('pk'))])

    def form_valid(self, form):
        comment = form.save(commit=False)
        comment.commentator = self.request.user
        comment.race_id = self.kwargs.get('pk')

        return super(RaceDetailView, self).form_valid(form)


# Детали о гонщике
class RacerDetailView(DetailView):
```

```
    template_name = 'racer_detail.html'
    model = Racer


# Создание гонщика
class RacerCreateView(CreateView):
    template_name = 'racer_create.html'
    model = Racer
    form_class = CreateRacerForm

    def get_success_url(self):
        return reverse('racer-detail', args=[str(self.object.id)])

    def form_valid(self, form):
        racer = form.save(commit=False)
        racer.user_info = self.request.user

        return super(RacerCreateView, self).form_valid(form)


# Обновление гонщика
class RacerUpdateView(UpdateView):
    model = Racer
    template_name = 'racer_create.html'
    form_class = CreateRacerForm

    def get_success_url(self):
        return reverse('racer-detail', args=[str(self.object.id)])


# Мои регистрации
class RegistrationListView(ListView):
    model = RaceRacer
    template_name = 'registration.html'

    def get_queryset(self):
        delete_id = self.request.GET.get('delete_id')

        if delete_id:
            try:
                delete_id = int(delete_id)
                RaceRacer.objects.get(race_id=delete_id,
racer_id=self.request.user.racer.id).delete()

            except:
                pass

        return RaceRacer.objects.filter(racer_id=self.request.user.racer.id)
```

Файл urls.py

```
from django.contrib import admin
from django.contrib.auth import views
from django.urls import path

from racing_scoreboard.views import *

urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', views.LoginView.as_view(), name='login'),
```

```
    path('logout/', views.LogoutView.as_view(), name='logout'),
    path('register/', register, name='register'),
    path('', RaceListView.as_view(), name='race-list'),
    path('registration/', RegistrationListView.as_view(), name='registration-
list'),
    path('race/<int:pk>', RaceDetailView.as_view(), name='race-detail'),
    path('racer/<int:pk>', RacerDetailView.as_view(), name='racer-detail'),
    path('racer/update/<int:pk>', RacerUpdateView.as_view(), name='racer-
update'),
    path('racer/create/', RacerCreateView.as_view(), name='racer-create'),
]
```

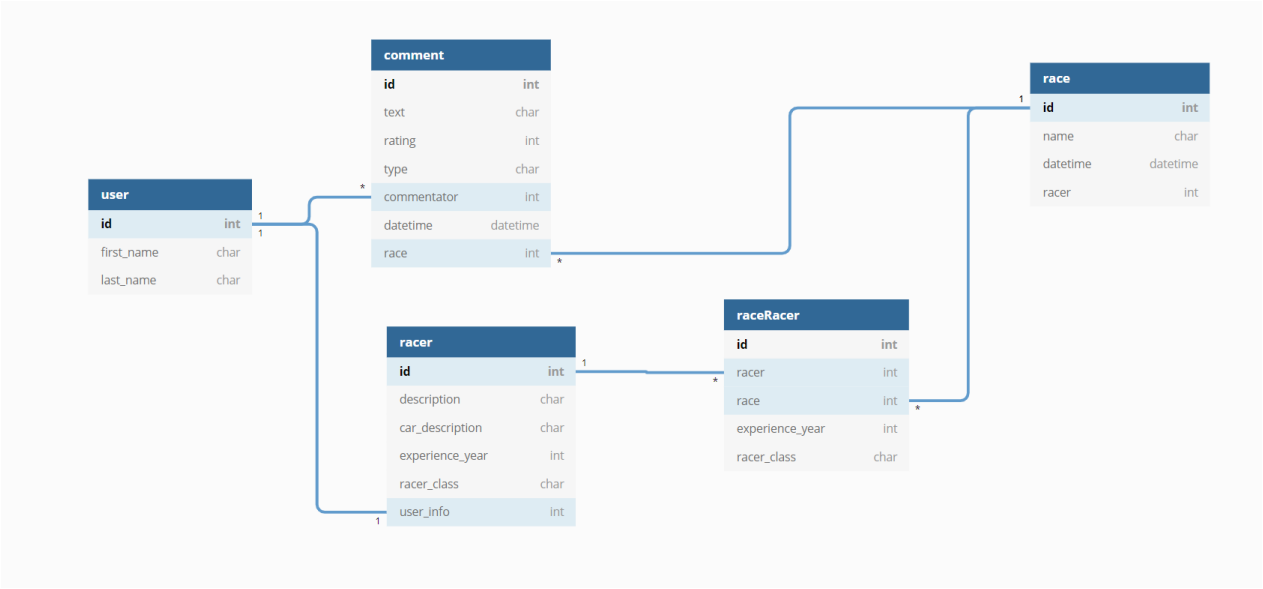
Модель данных models.py.

```
from django.contrib.auth.models import AbstractUser
from django.contrib.auth.models import Group
from django.db import models

# Create your models here.
from django.db.models import F
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.urls import reverse

from laboratory_work_2 import settings


class User(AbstractUser):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)

    def __str__(self):
        return "{} {}".format(self.first_name, self.last_name)


class Racer(models.Model):
    RACER_CLASS = [
        ('A', 'мотоциклы'),
        ('B', 'легковые автомобили'),
        ('C', 'грузовые автомобили'),
        ('D', 'пассажирские автомобили'),
        ('M', 'мопеды и легкие квадроциклы'),
    ]
    description = models.CharField(max_length=200)
    car_description = models.CharField(max_length=200)
    experience_year = models.IntegerField(default=0)
    racer_class = models.CharField(max_length=30, choices=RACER_CLASS)
    user_info = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, null=True, blank=True)
    team = models.CharField(max_length=100)

    def __str__(self):
        return self.user_info.__str__()

    def get_absolute_url(self):
        """
        Returns the url to access a particular instance of the model.
        """
        return reverse('racer-detail', args=[str(self.id)])


@receiver(post_save, sender=Racer)
```

```
def create_racer(sender, instance, created, **kwargs):
    if created:
        g = Group.objects.get(name='racer')
        g.user_set.add(instance.user_info)


class Race(models.Model):
    name = models.CharField(max_length=200)
    datetime = models.DateTimeField(blank=True, null=True)
    racer = models.ManyToManyField(Racer, through='RaceRacer')

    class Meta:
        ordering = ['-datetime']

    def __str__(self):
        return self.name

    @property
    def sorted_racers_set(self):
        return
self.raceracer_set.order_by(F('finish_time').asc(nulls_last=True))

    def get_absolute_url(self):
        """
        Returns the url to access a particular instance of the model.
        """
        return reverse('race-detail', args=[str(self.id)])


class Comment(models.Model):
    CATEGORIES = [
        ('cooperation', 'сотрудничество'),
        ('race', 'вопрос о гонках'),
        ('other', 'иное'),
    ]
    RATING = [
        (1, '1'),
        (2, '2'),
        (3, '3'),
        (4, '4'),
        (5, '5'),
        (6, '6'),
        (7, '7'),
        (8, '8'),
        (9, '9'),
        (10, '10'),
    ]
    type = models.CharField(max_length=15, choices=CATEGORIES)
    text = models.CharField(max_length=300, null=True, blank=True)
    rating = models.IntegerField(choices=RATING)
    commentator = models.ForeignKey(User, on_delete=models.CASCADE)
    datetime = models.DateTimeField(auto_now_add=True)
    race = models.ForeignKey(Race, on_delete=models.CASCADE)


class RaceRacer(models.Model):
    racer = models.ForeignKey(Racer, on_delete=models.CASCADE)
    race = models.ForeignKey(Race, on_delete=models.CASCADE)
    finish_time = models.DateTimeField(null=True, blank=True)
```

```
    class Meta:
        ordering = ['finish_time']
```

Схема:



Вывод: в ходе работы были изучены принципы проектирования простого сайта средствами Django и его реализация.