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

Факультет «Инфокоммуникационных технологий»
Направление подготовки «45.03.04 Интеллектуальные системы в гуманитарной среде»

О Т Ч Е Т

ЛАБОРАТОРНАЯ РАБОТА №2

Выполнил:
Студент Янов Ф.А. **К33422**
(Фамилия И.О.) номер группы

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

Санкт-Петербург
2021

Цель:

Овладеть практическими навыками и умениями реализации web-сервисов средствами Django 2.2.

Практическое задание:

Реализовать сайт используя фреймворк Django 3 и СУБД PostgreSQL *, в соответствии с вариантом задания лабораторной работы.

Описание:

Необходимо учитывать название отеля, владельца отеля, адрес, описание, типы номеров, стоимость, вместимость, удобства. Необходимо реализовать следующий функционал:

- Регистрация новых пользователей.
- Просмотр и резервирование номеров. Пользователь должен иметь возможность редактирования и удаления своих резервирований.
- Написание отзывов к номерам. При добавлении комментариев, должны сохраняться период проживания, текст комментария, рейтинг (1-10), информация о комментаторе.
- Администратор должен иметь возможность заселить пользователя в отель и выселить из отеля средствами Django-admin.
- В клиентской части должна формироваться таблица, отображающая постояльцев отеля за последний месяц.

Модель данных:

Модель состоит из 5 сущностей: клиент (с перечнем своих возможностей), отель, номер (комната отеля), резервирование, отзыв.

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


class Hotel(models.Model):
    name = models.CharField(max_length=40)
    owner = models.CharField(max_length=40)
    address = models.CharField(max_length=60)

    def __str__(self):
        return self.name


class Room(models.Model):
    hotel = models.ForeignKey('Hotel', on_delete=models.CASCADE)
    number = models.CharField(max_length=4)

    class RoomType(models.TextChoices):
        std = 'STD', "Standard"
        bdr = 'BDR', "With bedroom"
        brt = '2BDR', "With 2 bedrooms"
        sui = 'SUI', "Suite"
        ksu = 'KSU', "King Suite"
        vll = 'VLL', "Villa"
        bgg = 'BGG', "Bungalo"

    room_type = models.CharField(choices=RoomType.choices, max_length=4)
    price = models.CharField(max_length=10)

    def __str__(self):
        return self.number
```

Выше расположены модели Room и Hotel. У Room можно выбрать тип номера.

Модели Client и Booking (резервирование):

```
class Client(AbstractUser):
    id = models.AutoField(unique=True, primary_key=True)
    first_name = models.CharField(max_length=50, null=True)
    last_name = models.CharField(max_length=50, null=True)
    birthday = models.DateField(blank=True, null=True)
    passport = models.CharField(max_length=20, blank=True, null=True)
    phone_number = models.CharField(max_length=11, blank=True, null=True)
    room = models.ManyToManyField('Room', through='Booking')


class Booking(models.Model):
    client = models.ForeignKey('Client', on_delete=models.CASCADE)
    room = models.ForeignKey('Room', on_delete=models.CASCADE)
    people = models.IntegerField()
    start_date = models.DateField()
    end_date = models.DateField()
    check_in = models.BooleanField(default=False)
```

Модель Review (с выбором рейтинга отеля по пятибалльной шкале)

```
class Review(models.Model):
    user = models.ForeignKey('Client', on_delete=models.SET_NULL, null=True)
    hotel = models.ForeignKey('Hotel', on_delete=models.SET_NULL, null=True)
    room = models.ForeignKey('Room', on_delete=models.SET_NULL, null=True)

    class Rating(models.IntegerChoices):
        very_bad = 1
        bad = 2
        neutral = 3
        good = 4
        excellent = 5

    rating = models.IntegerField(choices=Rating.choices, null=True)
    date_start = models.DateField()
    date_end = models.DateField()
    text = models.CharField(max_length=1000)
```

Ссылки:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', HomePageView.as_view(), name='home'),
    path('hotels/', ListHotels.as_view(), name='hotels'),
    path('hotels/<int:pk>/', ListHotelRooms.as_view()),
    path('rooms/<int:pk>/book', CreateBooking.as_view(), name='booking'),
    path('register/', registration),
    path('login/', LoginView.as_view(template_name='login.html')),
    path('logout/', logout_view),
    path('account/login/', profile, name='profile'),
    path('profile/bookings', ListBookings.as_view(), name='bookings'),
    path('profile/bookings/delete/<int:pk>/', DeleteBooking.as_view(), name='delete_booking'),
    path('reviews/', ListReviews.as_view(), name='review_list'),
    path('rooms/<int:pk>/add_review', CreateReview.as_view(), name='add_review'),
    path('last_guests/', ListGuests.as_view(), name='last_guests')
]
```

Views:

Главная страница сайта, перенаправление на профиль, регистрация.

```
class HomePageView(TemplateView):
    template_name = 'home.html'


def profile(request):
    return render(request, 'profile.html')


def registration(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('/')

    else:
        form = RegistrationForm()
    return render(request, 'register.html', {'form': form})
```

Возможность выхода из аккаунта, а также страницы сайта: список отелей, список доступных номеров отеля.

```
def logout_view(request):
    logout(request)
    return redirect('/')


class ListHotels(ListView):
    model = Hotel
    template_name = 'hotels.html'
    context_object_name = 'hotels'


class ListHotelRooms(ListView):
    template_name = 'hotelrooms.html'
    context_object_name = 'hotel'

    def get_queryset(self):
        self.hotel = get_object_or_404(Hotel, pk=self.kwargs['pk'])
        return Room.objects.filter(hotel=self.hotel)
```

Создать бронь и посмотреть список «моих» бронирований.

```
class CreateBooking(CreateView):
    form_class = CreateBookingForm
    model = Booking
    template_name = 'newbooking.html'
    context_object_name = 'booking'
    success_url = '/profile/bookings'

    def get_initial(self):
        initial = super(CreateBooking, self).get_initial()
        initial = initial.copy()
        initial['client'] = self.request.user.pk
        initial['room'] = get_object_or_404(Room, pk=self.kwargs['pk'])
        return initial


class ListBookings(ListView):
    model = Booking
    template_name = 'bookings_list.html'
    context_object_name = 'bookings_list'

    def get_queryset(self):
        self.client = self.request.user.pk
        return Booking.objects.filter(client=self.client)
```

Удалить бронь, список отзывов, создать отзыв.

```
class DeleteBooking(DeleteView):
    model = Booking
    template_name = 'delete_booking.html'
    context_object_name = 'booking'
    success_url = '/profile/bookings'


class ListReviews(ListView):
    model = Review
    template_name = 'reviews.html'
    context_object_name = 'reviews'
```

```python
class CreateReview(CreateView):
    model = Review
    form_class = CreateReviewForm
    template_name = 'create_review.html'
    context_object_name = 'review'
    success_url = '/reviews'

    def get_initial(self):
        initial = super(CreateReview, self).get_initial()
        initial = initial.copy()
        room = Room.objects.get(pk=self.kwargs['pk'])
        room_hotel = getattr(room, 'hotel')

        initial['user'] = self.request.user.pk
        initial['hotel'] = room_hotel
        initial['room'] = room

        return initial
```

Список гостей отеля за последний месяц.

```python
class ListGuests(ListView):
    template_name = 'lastguests.html'
    context_object_name = 'guests_list'

    def get_queryset(self):
        last_month = datetime.date.today() - datetime.timedelta(days=30)
        booking_queries = Booking.objects.filter(check_in=True).filter(start_date__gte=last_month)

        return booking_queries
```

Формы:

Наш сайт предполагает несколько встроенных форм таких как: регистрация, создание брони и создание отзыва.

```python
from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import Client, Booking, Review
from datetime import date


class RegistrationForm(UserCreationForm):
    class Meta:
        model = Client
        fields = ('username', 'email', 'first_name', 'last_name', 'birthday', 'phone_number', 'password1', 'password2')
```

В создании брони мы ограничиваем возможность редактирования «клиента» и «комнаты», дабы усовершенствовать систему выбора отеля. Также учитываем варианты, при которых пользователь бронируют номер в прошлом или дата окончания брони раньше даты заезда. В таких случаях будет ошибка.

```python
class CreateBookingForm(forms.ModelForm):
    class Meta:
        model = Booking
        fields = ('client', 'room', 'people', 'start_date', 'end_date')
        widgets = {
            'start_date': forms.DateInput(attrs={'type': 'date'},
                                          format='d-%m-%Y'),
            'end_date': forms.DateInput(attrs={'type': 'date'},
                                        format='d-%m-%Y'),
        }

    def __init__(self, *args, **kwargs):
        super(CreateBookingForm, self).__init__(*args, **kwargs)
        self.fields['client'].disabled = True
        self.fields['room'].disabled = True

    def clean(self):
        cleaned_data = super().clean()
        start_date = cleaned_data.get("start_date")
        end_date = cleaned_data.get("end_date")
        today = date.today()

        if start_date > end_date:
            raise forms.ValidationError("Your meeting shoud be earlier than last day!")

        if end_date < today:
            raise forms.ValidationError("Your date is in the past")
```

Примерно те же ограничения в форме создания отзыва. Отметим, что имя пользователя на сайте, оставившего отзыв, будет выводиться с помощью никнейма.

```python
class CreateReviewForm(forms.ModelForm):
    class Meta:
        model = Review
        fields = ('user', 'hotel', 'room', 'rating', 'date_start', 'date_end', 'text')
        widgets = {
            'date_start': forms.DateInput(attrs={'type': 'date'},
                                          format='d-%m-%Y'),
            'date_end': forms.DateInput(attrs={'type': 'date'},
                                        format='d-%m-%Y'),
            'text': forms.Textarea(),
        }

    def __init__(self, *args, **kwargs):
        super(CreateReviewForm, self).__init__(*args, **kwargs)
        self.fields['user'].disabled = True
        self.fields['room'].disabled = True
        self.fields['hotel'].disabled = True
```

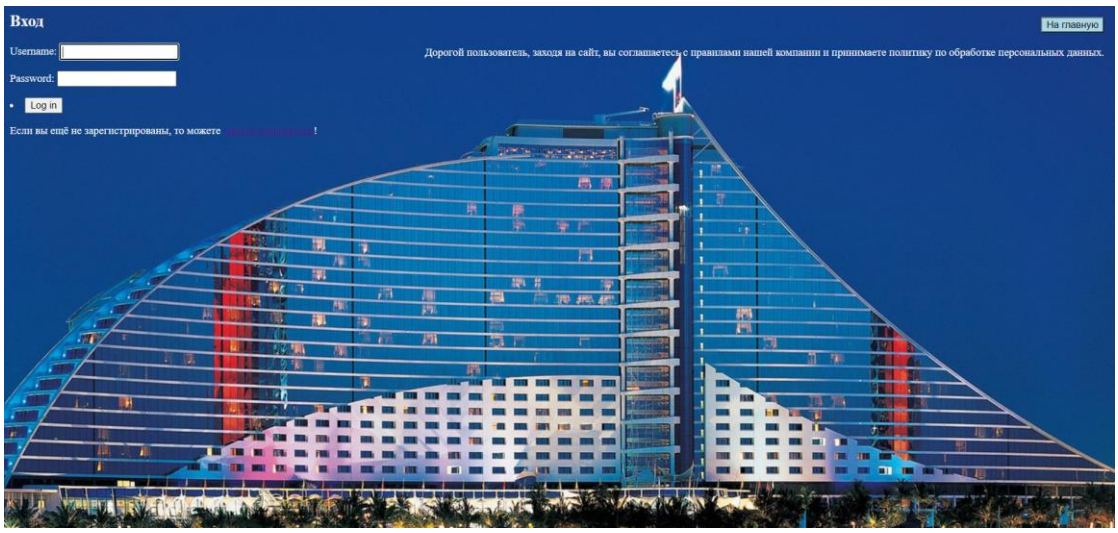
Теперь зайдём на сайт и просмотрим всю работу в действии!

Скрины работы сайта с клиентской стороны:

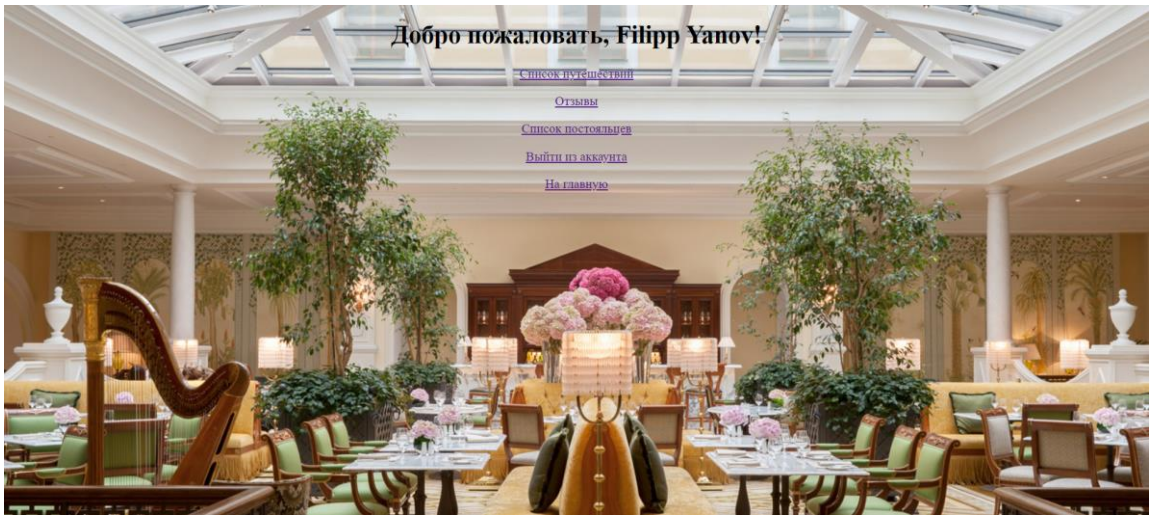
Главная страница сайта:

Предусмотрена возможность зайти в панель администратора (отдельная кнопка), возможности войти в профиль/зарегистрироваться, а также посмотреть список отелей. Но если мы зайдём в список отелей, то увидим, что информация от нас будет скрыта, пока мы не авторизуемся в системе.

Давайте войдём в систему, чтобы попробовать «скрытые» функции. «На главную» -> «Войти».

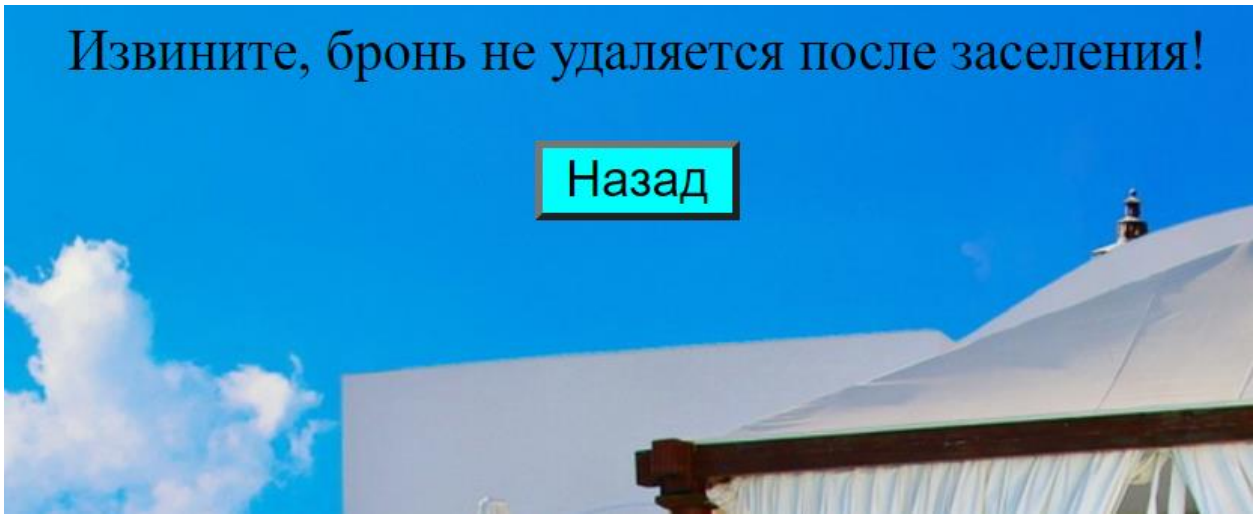


Захожу в систему под своими данными (можно зарегистрироваться, если до этого не было аккаунта) и вижу приветствующую страницу.

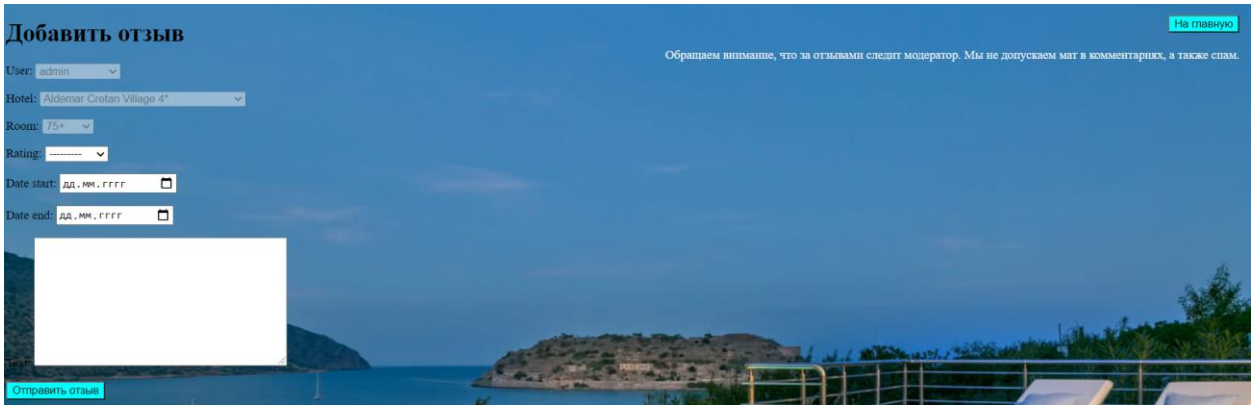


Здесь мы можем посмотреть историю наших путешествий, отзывы к отелям, список гостей последнего месяца, зарегистрированных на сайте, а также, конечно, выйти из аккаунта или вернуться на главную. Давайте посмотрим список путешествий.

Мы уже «заселены» – попробуем удалить бронь – уже не получится, поскольку бронь утверждена, а мы уже въехали.

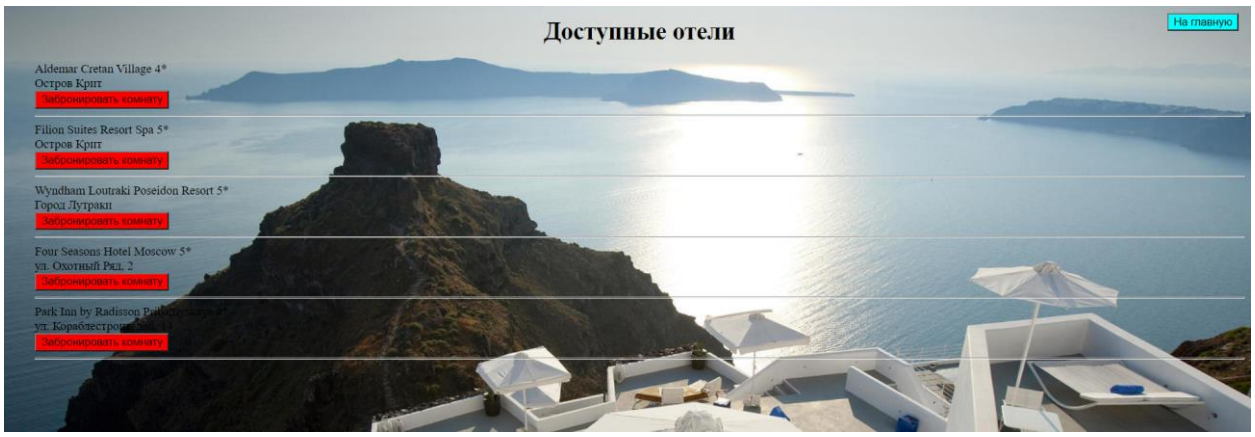


Поскольку мы въехали – уже можно оставить отзыв.



Дальше отправляем отзыв и выведет на страницу с отзывами (в неё также можно перейти из профиля)

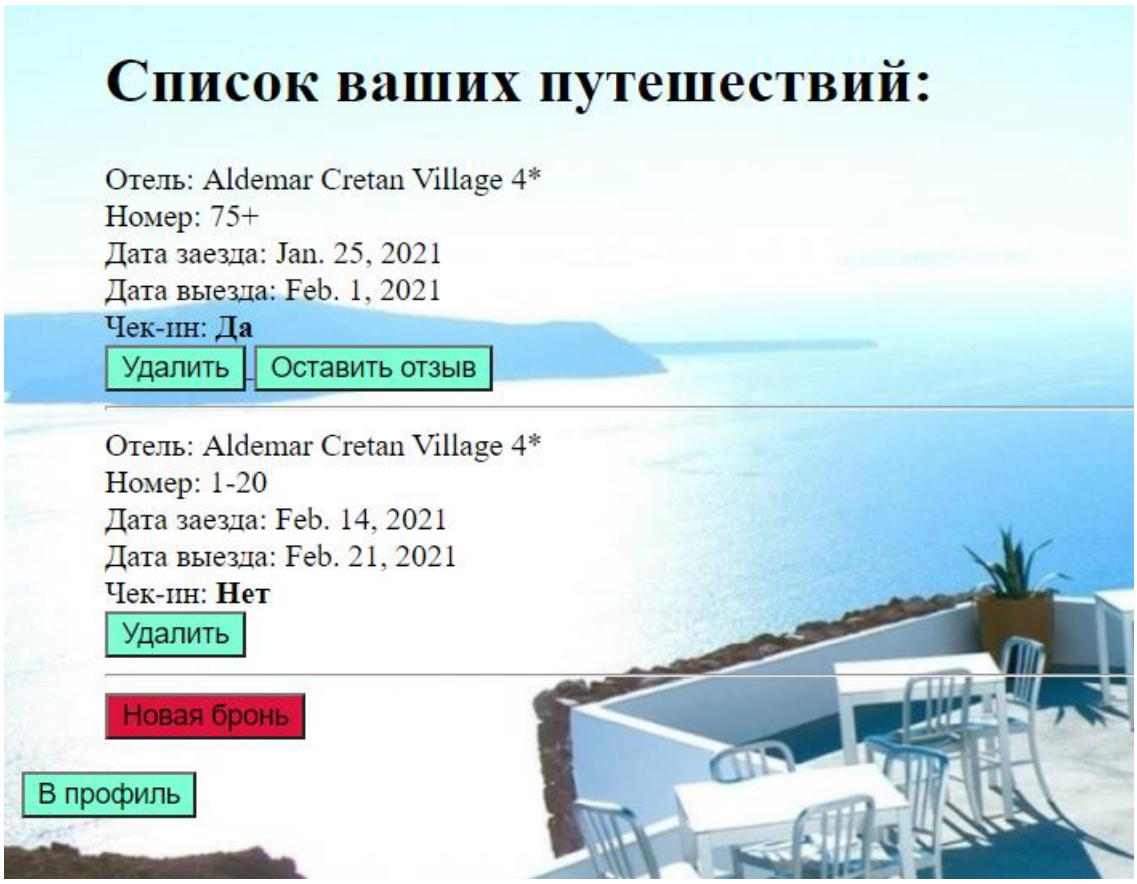
Если мы хотим сделать бронь на следующее путешествие, то идём в список отелей. Поскольку мы уже вошли в профиль, то можем увидеть следующую картину:



Забронируем комнату в первом отеле.

Бронируем стандартный номер.

Бронь отобразится в нашем профиле, но мы ещё не заселены, так что есть возможность удалить.



И посмотрим список постояльцев за последние 30 дней.

Выводы:

В ходе выполнения данной лабораторной работы мною были получены навыки работы с Django, был создан сайт-аналог «Триваго», были созданы регистрационная форма и предусмотрена возможность входа.