

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

ПО ДИСЦИПЛИНЕ:

«Web-программирование»

НА ТЕМУ:

«Реализация простого сайта средствами Django»

Работу выполнил:

Студент гр. К33422

Кириллов Николай Александрович

Преподаватель:

Говоров Антон Игоревич

Санкт-Петербург

2020

Цель

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

Задание

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

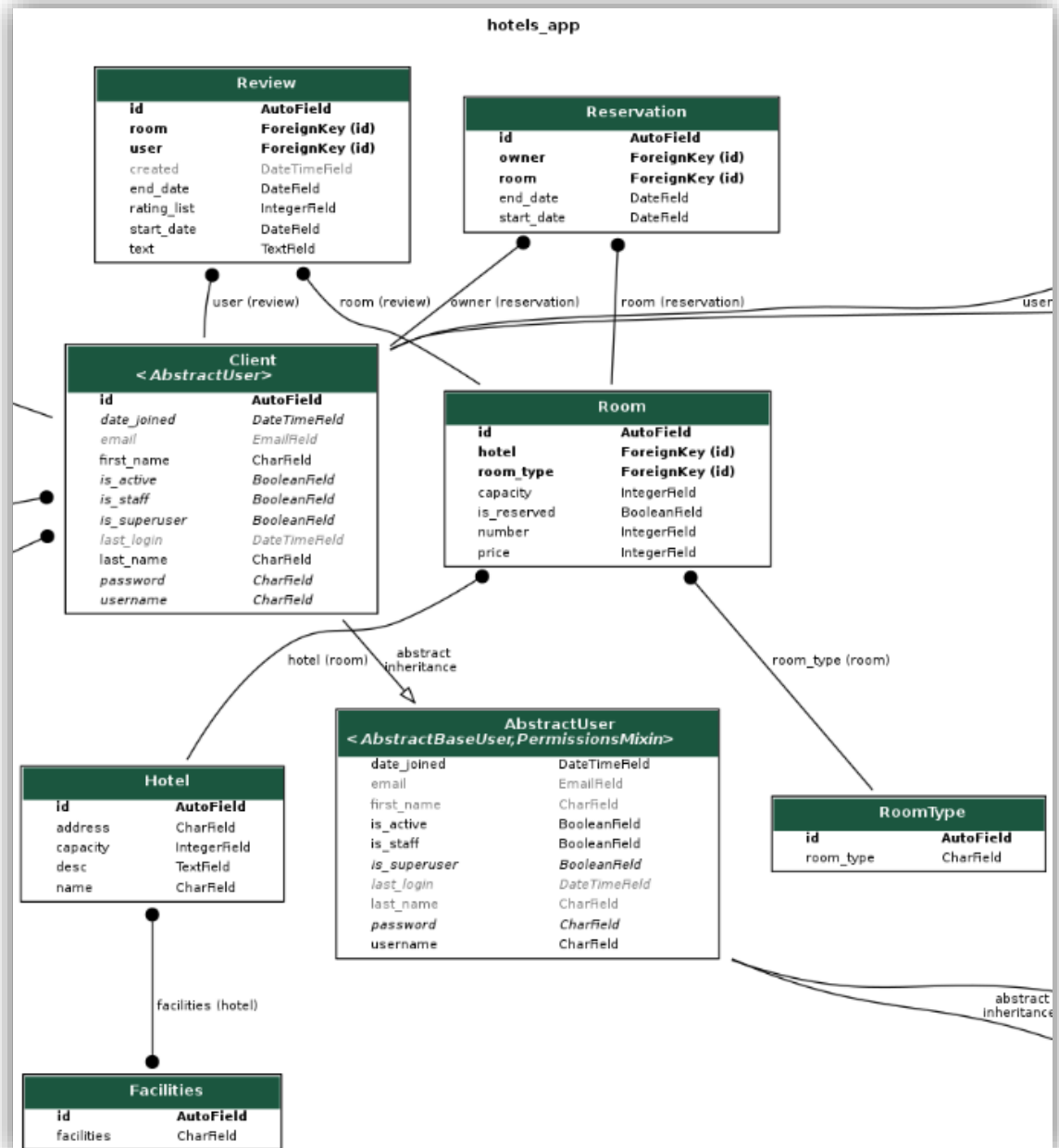
Вариант 1: Список отелей

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.

Описание модели данных



Сущность «Клиент»

Хранит имя и фамилию пользователя

```
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)
```

Сущность «Тип комнаты»

Хранит тип комнаты (люкс, одиночная, президентская...)

```
class RoomType(models.Model):
    room_type = models.CharField("Type", max_length=50)
```

Сущность «Удобства»

Хранит удобства (spa, бар, бассейн...)

```
class Facilities(models.Model):  
    facilities = models.CharField("Facilities", max_length=50)
```

Сущность «Комната»

Хранит владельца комнаты, номер, тип комнаты (FK), вместимость, цену, отель (FK) и статус (зарезервирована или нет)

```
class Room(models.Model):  
    is_reserved = models.BooleanField("Reservation", default=False)  
    owner = models.ManyToManyField('Client', verbose_name="Owner", through='Reservation')  
    hotel = models.ForeignKey('Hotel', verbose_name="Hotel", on_delete=models.SET_NULL, null=True)  
    number = models.IntegerField("Number")  
    room_type = models.ForeignKey('RoomType', verbose_name="Type", on_delete=models.SET_NULL, null=True)  
    capacity = models.IntegerField("Capacity")  
    price = models.IntegerField("Price")
```

Сущность «Отель»

Хранит название отеля, адрес, описание, вместимость, удобства

```
class Hotel(models.Model):  
    name = models.CharField("Name", max_length=50)  
    address = models.CharField("Address", max_length=100)  
    desc = models.TextField("Description", max_length=300)  
    capacity = models.IntegerField("Capacity")  
    facilities = models.ManyToManyField('Facilities', verbose_name="Facili
```

Сущность «Отзыв»

Хранит период проживания, пользователя (FK), комнату (FK), текст отзыва, дату создания отзыва и рейтинг (1-5)

```
class Review(models.Model):
    start_date = models.DateField("StartDate", null=True)
    end_date = models.DateField("EndDate", null=True)
    user = models.ForeignKey('Client', verbose_name="User", on_delete=models.CASCADE)
    room = models.ForeignKey('Room', verbose_name="Room", on_delete=models.CASCADE)
    text = models.TextField("Review")
    created = models.DateTimeField("Date", auto_now_add=True, null=True)

    class Rating(models.IntegerChoices):
        very_unsatisfied = 1
        unsatisfied = 2
        neutral = 3
        satisfied = 4
        very_satisfied = 5

    rating_list = models.IntegerField(choices=Rating.choices, null=True)
```

Связующая сущность «Резервация»

Хранит пользователя (FK), комнату (FK), дату заезда и дату выселения

```
class Reservation(models.Model):
    owner = models.ForeignKey('Client', verbose_name="Owner", on_delete=models.CASCADE)
    room = models.ForeignKey('Room', verbose_name="Room", on_delete=models.CASCADE)
    start_date = models.DateField("StartDate", null=True)
    end_date = models.DateField("EndDate", null=True)
```

Описание контролеров в views

```
# стартовая страница
def Index(request):
    template_name = 'index.html'
    return render(request, "index.html")

# вывод списка постояльцев за последний месяц
def HotelListView(request):
    today = datetime.date.today()
    today_year = today.year
    today_day = today.day
    last_month = today.month - 1 if today.month > 1 else 12
    month_ago = datetime.date(today_year, last_month, today_day)
    context = {}
    context["guests"] = Reservation.objects.all().order_by('room__hotel').filter(end_date__range=(month_ago, today))
    return render(request, "hotel_list.html", context)

# вывод списка пользователей
class UserListView(ListView):
    model = Client
    template_name = 'user_list_view.html'

# вывод списка комнат
class RoomListView(ListView):
    model = Room
    template_name = 'room_list_view.html'
```

```
# вывод информации о комнате
def RoomView(request, pk):
    room = get_object_or_404(Room, id=pk)
    review = Review.objects.filter(room=pk)
    form = ReviewForm(request.POST or None)
    context = {"room": room, "reviews": review, "form": form}
    if form.is_valid():
        form = form.save(commit=False)
        form.user = request.user
        form.room = room
        form.save()
        return redirect('room_view', pk)

    return render(request, "room_view.html", context)
```

```
# страница резервации
def ReservationView(request, pk):
    room = get_object_or_404(Room, id=pk)
    reservation = Reservation.objects.filter(room=pk)
    res_form = ReservationForm(request.POST or None)
    context = {"room": room, "reservation": reservation, "res_form": res_form}
    if res_form.is_valid():
        res_form = res_form.save(commit=False)
        res_form.owner = request.user
        res_form.room = room
        # room.is_reserved = True
        Room.objects.filter(id=pk).update(is_reserved=True)
        res_form.save()
    return render(request, "reservation.html", context)
```

```
# страница удаления
def ResDelete(request, pk):
    query = get_object_or_404(Reservation, room_id=pk)
    owner = query.owner.username
    start_date = query.start_date
    user_now = request.user.username
    today = datetime.date.today()
    if str(owner) == str(user_now):
        if start_date < today:
            return HttpResponse('You cannot delete reservations that have passed')
        else:
            query.delete()
            Room.objects.filter(id=pk).update(is_reserved=False)
            return render(request, "res_delete.html")
    return HttpResponse('Not your reservation or ')
```

Описание роутеров в urls

Urls проекта

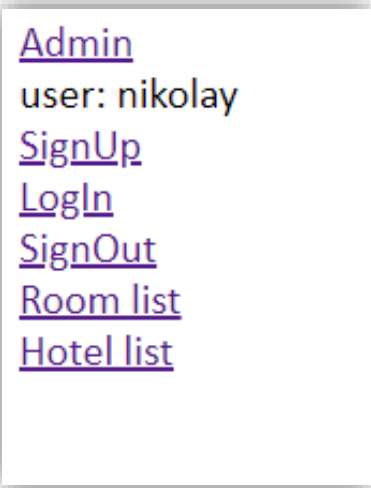
```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', include('hotels_app.urls')),  
    path('accounts/', include('allauth.urls')),  
]
```

Urls приложения

```
urlpatterns = [  
    path('user/list/', UserListView.as_view(), name="user_list"),  
    # path('user/add/', AddUser.as_view(success_url="http://127.0.0.1:8000/user/list/")),  
    path('room/<int:pk>', views.RoomView, name="room_view"),  
    path('room/list', RoomListView.as_view(), name="room_list"),  
    path('hotel/list', views.HotelListView, name="hotel_list"),  
    path('room/<int:pk>/reservation', views.ReservationView, name="reservation"),  
    path('room/<int:pk>/delete/', views.ResDelete, name="res_delete"),  
    path('', views.Index, name="index")  
]
```

Скрины работы программы

Главная страница



[Admin](#)
user: nikolay
[SignUp](#)
[LogIn](#)
[SignOut](#)
[Room list](#)
[Hotel list](#)

Регистрация нового пользователя

Sign Up

Already have an account? Then please [sign in](#).

Username:

E-mail (optional):

Password:

Password (again):

Список комнат

- **Rose_hotel**

Number: 1 **Reserved** [Details](#)

- **Hotel_1_Test**

Number: 667 **Free** [Details](#)

- **Hotel_1_Test**

Number: 555 **Reserved** [Details](#)

- **Hotel_1_Test**

Number: 678 **Reserved** [Details](#)

- **Rose_hotel**

Number: 342 **Reserved** [Details](#)

- **Hotel_1_Test**

Number: 123 **Free** [Details](#)

- **Hotel_1_Test**

Number: 456 **Reserved** [Details](#)

Информация о комнате (свободной), реализован вывод отзывов и можно оставить свой

667

Free [\[Make a reservation\]](#)

Hotel_1_Test

Status:

Type: Single

Capacity: 1

price: 876

Reviews:

User - nikolay
Stay from Nov. 1, 2020 to Jan. 15, 2020
All was ok
Rating - 2
Created - Dec. 2, 2020, 5:59 p.m.

Review:

Rating list:

StartDate:

EndDate:

send

[Get back](#)

Информация о комнате, зарезервированной пользователем, который видит это (есть возможность удалить резервацию)

555

[\[Delete a reservation\]](#) Reserved

Информация о комнате, зарезервированной кем-то другим

1

Reserved

Пользователь должен быть авторизован для получения возможности оставлять отзывы

Reviews:

User - nikolay

Stay from Nov. 1, 2020 to Jan. 15, 2020

All was ok

Rating - 2

Created - Dec. 2, 2020, 5:59 p.m.

You need to be logged in to leave a review

[Get back](#)

Вывод постояльцев за последний месяц

--Hotel_1_Test--

- Name: nikolay
Room: 678

Stayed _____

From: Oct. 31, 2020

To: Nov. 30, 2020

- Name: vova
Room: 123

Stayed _____

From: Nov. 1, 2020

To: Dec. 1, 2020

--Rose_hotel--

- Name: nikolay
Room: 342

Stayed _____

From: Oct. 31, 2020

To: Nov. 19, 2020

Вывод

В результате выполнения лабораторной работы были получены практические навыки и умения реализации web-серверов средствами Django; создан простой сайт для отеля