Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.

Факультет инфокоммуникационных технологий.

Лабораторная работа №2 по теме: «Реализация простого сайта средствами Django» по дисциплине Web-программирование.

Выполнила: Тихонова Е.К.

Группа: К33401

Вариант: 15 (3)

Преподаватель: Говоров Антон Игоревич

Цель работы: овладеть практическими навыками и умениями реализации web-сервисов средствами Django 2.2.

Вариант 3. Табло отображения информации об авиаперелетах.

Должна храниться информация о номере рейса, авиакомпании, отлете, прилете, типе (отлет, прилет), номере гейта.

Также необходимо реализовать следующий функционал:

- Регистрация новых пользователей
- Просмотр и резервирование мест на рейсах. Пользователь должен иметь возможность редактирования и удаления своих резервирований.
- Администратор должен иметь возможность зарегистрировать на рейс пассажира и вписать в систему номер его билета средствами Django-admin.
- В клиентской части должна формироваться таблица, отображающая всех пассажиров рейса.
- Написание отзывов к рейсам. При добавлении комментариев, должны сохраняться дата рейса, текст комментария, рейтинг (1-10), информация о комментаторе.

Модель данных

Flight(django.db.models.Model)

Поля: number, airline, is_departure (тот самый тип: если True, то рейс отображается среди Departures в таблице с полем gate, иначе без него), gate, dep_time, arr_time.

Также есть поле passengers (ManyToManyField с моделью User через Resrvation, указано как related_name)

В классе Meta описан порядок сортировки – ['dep_time', 'number']

Описаны функции get_absolute_url, get_reservation_url, get_comment_url, get_passenger_list_url, для каждой из них возвращается reverse со словарем именнованых аргументов, который содержит number объекта.

User(django.contrib.auth.models.AbstractUser)

Переопределены поля класса AbstractUser – first_name, а также last_name больше не могут быть пустыми.

Также есть поле flights (ManyToManyField с моделью Flight через Reservation)

Reservation(django.db.models.Model)

С помощью этой модели реализуется отношение многие ко многим: поля flight и passenger(User) — внешние ключи.

Остальные поля – seat, ticket_number (unique, присваивается администратором, значит может быть null), date.

Порядок сортировки, указанный в классе Meta: ['date', 'seat']

Функция get_edit_reservation_url передает словарь kwargs, содержащий self.id, а также self.flight.number

Comment(django.db.models.Model)

Содрежит поля flight (FK), user (FK), text, rating, fligh date.

Через объекты Flight или User к полю comment_set нигде обращаться не было необходимости, поэтому в моделях Flight и User поле Comments нет.

Роутеры файла urls.py

```
urlpatterns = [
    path('signup/', UserCreate.as_view(), name='register_url'),
    path('flights/', FlightList.as_view(), name='flight_list_url'),
    path('flight/<str:number>', FlightDetail.as_view(), name='flight_detail_url')
    path('flight/<str:number>/make_a_reservation',
         ReservationCreate.as_view(), name='create_reservation_url'),
    path('my_reservations/', ReservationList.as_view(),
         name='reservation list url'),
    path('flight/<str:number>/edit_a_reservation/<int:pk>',
         ReservationEdit.as_view(), name="edit_reservation_url"),
    path('sign_in/', signin, name='login_url'),
    path('sign_off/', signoff, name='logout_url'),
    path('flight/<str:number>/comment',
         CommentCreate.as_view(), name='comment_url'),
    path('flight/<str:number>/passenger_list',
         PassangerList.as_view(), name="passenger_list_url"),
    path('', redirect_board),
```

В случае с описанными выше функциями, в шаблонах указываю не {% url 'name' %}, а {{ obj.func_name }}

И urlpaterns из project_name/urls.py выглядит следующим образом:

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

Контроллеры

Как можно видеть выше, почти все Views основаны на классах. Исключениями являются только

- redirect_board возвращает редиркт на страницу с рейсами)
- signoff
 вызывает django.contrib.auth.logout
- signin в случае get запроса (или неудачной попытки авторизации) возвращает страницу с формой

UserLoginForm. В противном случае используя данные из связанной формы происходит попытка получить пользователя с помощью функции **django.contrib.auth.authenticate**. Здесь же происходит дополнительная проверка (если пользователя с username не существует, будет показано соответствующее сообщение, добавленное в форму с помощью функции **add_error('field', 'error text')**)

Среди class based views только одна наследует django.views.generic.edit.UpdateView – ReservationEdit.

 ReservationEdit
 в данном случае, вся логика используется из родительского класса, требуется только указать значения некоторых полей.

```
login_url = '/sign_in/'
redirect_field_name = None
model = Reservation
form_class = ReservationEditForm
template_name = 'board/edit_reservation.html'
context_object_name = 'reservation'
success_url = '/my_reservations/'
```

В остальных случаях происходит наследование класса **django.views.generic.View**, так как требуются дополнительные проверки валидных данных.

- UserCreate
 - использует форму UserForm. После успешного создания нового пользователя сразу происходит **login(request, new_user)**. Переместив решетку комментария на строку выше можно реализовать другое поведение после успешной регистрации отправлять пользователя на страницу входа в учетную запись.
- FlightList
 - шаблону отправляется два QuerySet-а для отправлений и прибытий (чтобы были сформированы две таблицы, только одна из которых содержит колонку gate)
- FlightDetail
 - на этой странице странице можно увидеть комментарии к рейсу, а также кнопки:
 - Зарезервировать место (если not user.is_authenticated, отправит на страницу входа в учетную запись)
 - Посмотреть список пассажиров рейса (также, доступно только аутентифицированным пользователям)
 - Oставить комментарий эта кнопка будет видна только тем, кто зарегестрирован на рейс (есть резервирование и администратор вписал в базу номер его билета)

Остальные views наследуют **django.contrib.auth.mixins.LoginRequiredMixin.** Что дает им все пару общих строчек кода:

```
login_url = '/sign_in/'
redirect field name = None
```

- CommentCreate
 - возможность комментировать есть только у авторизованных пользователей, у которых есть Reservation с номером билета на данный рейс (то есть администратор их зарегистрировал). Используется CommentCreateForm с дополнительной проверкой поля rating (в модели указано, как DecimalField с двумя знаками) значение не может быть более 10.

- PassengerList
 - отображает список пассажиров рейса, если таковой, конечно, может быть найден. В противном случае, обращается к уже описанной функции redirect_board.
- ReservationList
 - отображает резервирование в виде таблицы, с дополнительным полями Edit и Delete, содержащими кнопки. Кнопки disabled, если есть номер билета (произошла регистрация администратором). Если резервирование не превратилось в регистрацию на рейс, его можно удалить произойдет мнгновенно и тут же отобразит обновленную таблицу, либо же редактировать, что отсылает нас к ReservationEdit, описанному выше.
- ReservationCreate
 - сидения на странице представлены radio, которые выглядят как обычные небольшие кнопки, выведенные в виде таблицы 4 на 9. Номера сидений прописаны вложенным списком, содержащим внутри словарь 'seat_number': цифрыБуква. Используется форма ReservationForm, перед сохранением валидные данные проверяются дополнительно. Дата может быть от «завтра» до «через две недели». Номер места выбирать обязательно. Если что-то из этого не соответствует действительности, в bound_form будет добавлено соответствующее сообщение об ошибке.

Стоит добавить, что во всех формах, user и flight передаются через input with hidden type. А в случае с написание комментария, таким же образом передается дата полета.

Почти всем полям форм (исключениями являются HiddenInput) из файла forms.py добавлен словарь widget, содержащий словарь {'class': 'form-control'}. В некоторых случаях, добавлен класс, увеличивающий внешние отступы от Input.

Шаблоны

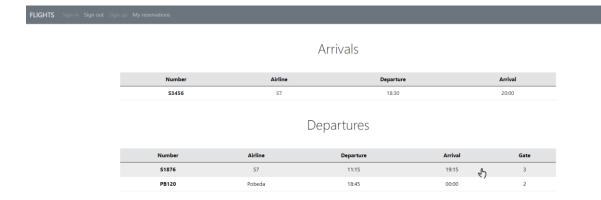
Все шаблоны наследуют base.html, который содержит navbar.

При работе над шаблонами активно использовался Bootstrap.

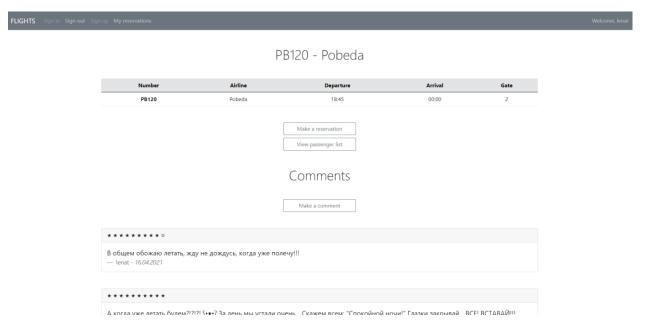
В случае с таблицей рейсов (где по задумке автора, при нажатии на строку должна была происходить переадресация на страницу с детальной информацией и комментариями к рейсу) изза использования Bootstrap, не было возможности обернуть строки ссылками. В результате, все работает, как планировалось, но для этого в каждой ячейке (и на весь размер ячейки) были добавлены прозрачные кнопки.

Время отображалось в формате "midnight" or "8 p.m.", дата в виде "April 14, 2021". Приведено к надлежащему виду с помощью фильтров

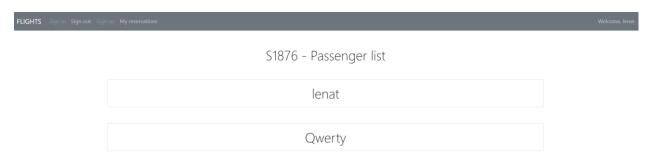
time time:"H:i"	
date date:"d.m.Y"	
Выбор мест, как было описано выше, представляет из себя таблицу с элементами radio.	
Скриншоты	
,	



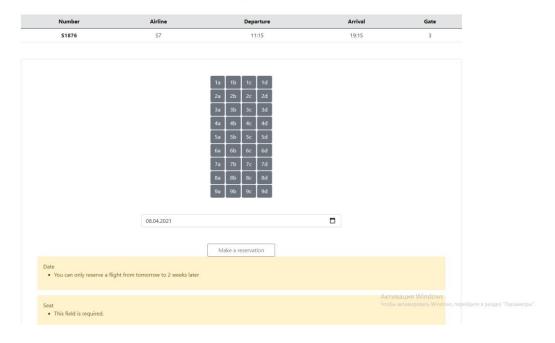
B navbar доступность кнопок зависит от того, авторизован ли пользователь, это же влияет на наличие надписи в правом верхнем углу. Строки таблицы целиком кликабельны.

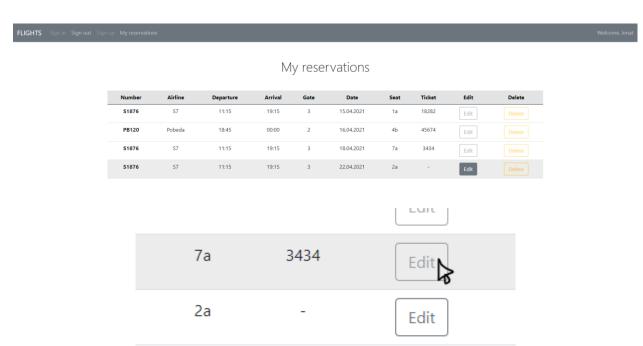


Как зарегестрированный на рейс авторизованный пользователь могу оставить комментарий. Кнопки Make a comment нет у анонимного пользователя.



S1876 - Making a reservation





Редактировать или удалять можно только те резервирования, на которые администратор не зарегистрировал.

FLIGHTS Sign in Sign off Sign up My reservations					
		Sign in			
	Username Password	lenat			
		Sign in!			

Если пользователь не авторизован, ему не доступны кнопки Sign out, My reservations, зато доступна авторизация. На эту страницу он будет отправлен при попытке создать резервирование или смотреть список пассажиров рейса.