

МГТУ им. Н. Э. Баумана, кафедра ИУ5
курс “Разработка интернет-приложений”

Лабораторная работа №4 Django. Шаблонизация

ВЫПОЛНИЛ:

Матюнин да Вейга Р.А.

Группа: ИУ5-51Б

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2019

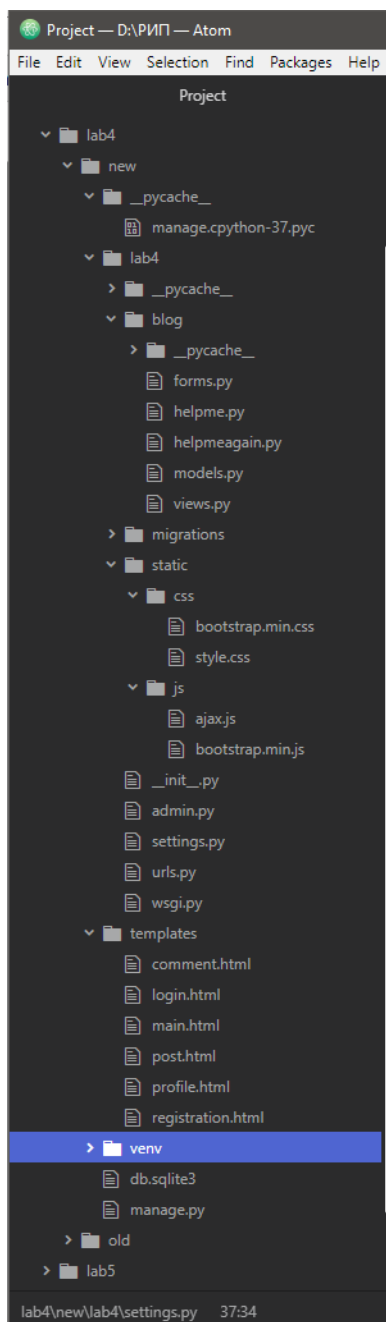
Задание и порядок выполнения

В этой ЛР вы создадите Django-проект, покажете пользователю статичную страницу, познакомитесь с конструкциями шаблонизаторов: переменные, теги, наследование шаблонов.

1. Создать проект
2. Реализовать view, в которых генерируются html-страницы
3. В шаблонах должны быть использованы рассмотренные конструкции: переменные, вложенные значения, циклы, условия
4. Все шаблоны должны расширять базовый шаблон
5. Для элементов списка использовать тег include
6. По нажатии на элемент списка должна открываться страница информации об элементе
7. Для верстки необходимо использовать Bootstrap

Выполненная работа

Структура проекта



models.py

Project — D:\PMT — Atom

File Edit View Selection Find Packages Help

Project

- lab4
 - new
 - __pycache__
 - manage.cpython-37.pyc
lab4
 - __pycache__
blog
 - __pycache__
forms.py
helpme.pyhelpmeagain.py**models.py**views.pymigrationsstatic

 - css
 - bootstrap.min.css
style.css

js

 - ajax.js
bootstrap.min.js
__init__.pyadmin.pysettings.pyurls.pywsgi.pytemplates

 - comment.html
login.htmlmain.htmlpost.htmlprofile.htmlregistration.html
venvdb.sqlite3manage.pyoldlab5

models.py

```
1 from django.db import models
2
3 import pymysql
4
5
6 class Post(models.Model):
7     post_head = models.CharField(max_length=70)
8     post_text = models.CharField(max_length=255)
9     publication_date = models.DateField('Date published')
10
11     def __str__(self):
12         return self.post_head
13
14
15 class Comment(models.Model):
16     id_post = models.ForeignKey(Post, on_delete=models.CASCADE)
17     answer_text = models.CharField(max_length=70)
18     rating = models.IntegerField(default=0)
19
20     def __str__(self):
21         return self.answer_text
22
23
24 class Connection:
25     def __init__(self, user, password, db, host='localhost'):
26         self.user = user
27         self.host = host
28         self.password = password
29         self.db = db
30         self._connection = None
31
32     @property
33     def connection(self):
34         return self._connection
35
36     def __enter__(self):
37         self.connect()
38
39     def __exit__(self, exc_type, exc_val, exc_tb):
40         self.disconnect()
41
42
```

lab4\new\lab4\blog\models.py 1:1

CRLF UTF-8 Python GitHub Git (0)

Project — D:\PMT — Atom

File Edit View Selection Find Packages Help

Project

- lab4
 - new
 - __pycache__
 - manage.cpython-37.pyc
lab4
 - __pycache__
blog
 - __pycache__
forms.py
helpme.pyhelpmeagain.py**models.py**views.pymigrationsstatic

 - css
 - bootstrap.min.css
style.css

js

 - ajax.js
bootstrap.min.js
__init__.pyadmin.pysettings.pyurls.pywsgi.pytemplates

 - comment.html
login.htmlmain.htmlpost.htmlprofile.htmlregistration.html
venvdb.sqlite3manage.pyoldlab5

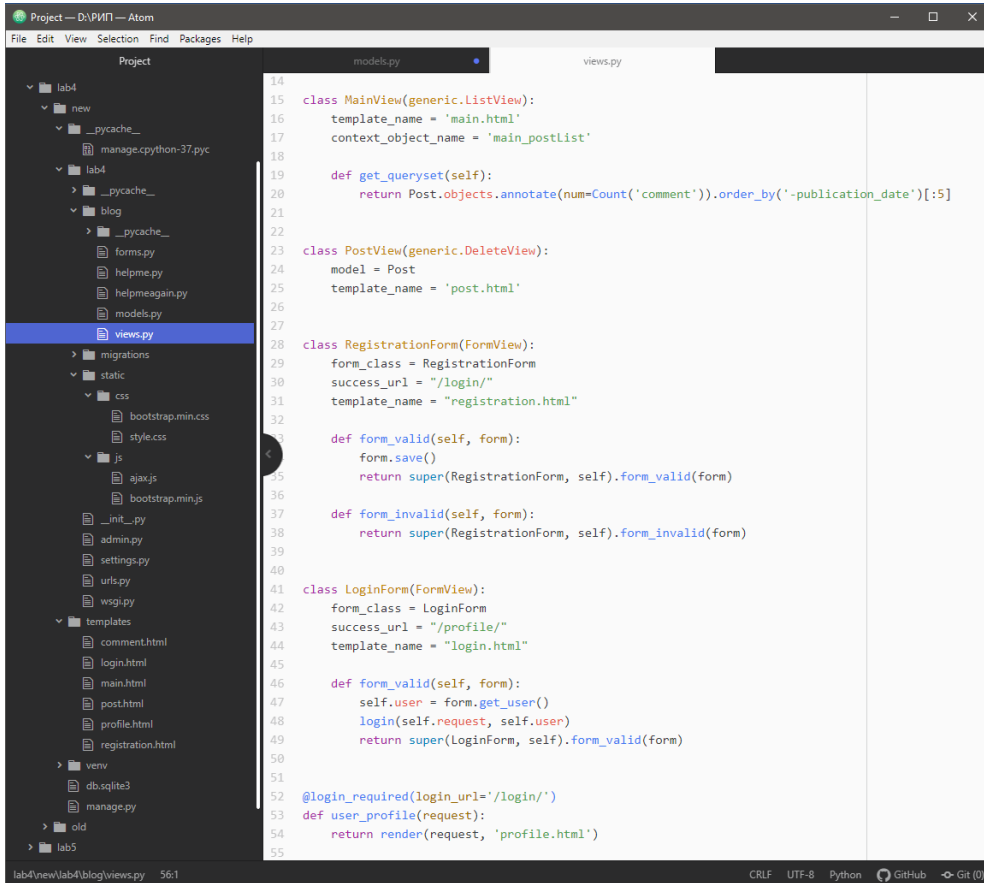
models.py

```
39 def __exit__(self, exc_type, exc_val, exc_tb):
40     self.disconnect()
41
42 def connect(self):
43     if not self._connection:
44         self._connection = pymysql.connect(
45             host=self.host,
46             user=self.user,
47             passwd=self.password,
48             db=self.db
49         )
50
51 def disconnect(self):
52     if self._connection:
53         self._connection.close()
54
55
56 class Userrr:
57
58     def __init__(self, db_connection, name, age, email):
59         self.db_connection = db_connection.connection,
60         self.name = name,
61         self.age = age,
62         self.email = email
63
64     def save(self):
65         c = self.db_connection.cursor()
66         c.execute("INSERT INTO users (name, age, email) VALUES (%s, %s, %s)",
67             ('Патри', '11', '41@mail.ru'))
68         self.db_connection.commit()
69         c.close()
70
71
72 class UserModel(models.Model):
73     name = models.CharField(max_length=30)
74     age = models.IntegerField(default=0)
75     email = models.CharField(max_length=30)
76
77
78
79
```

lab4\new\lab4\blog\models.py* 67:21

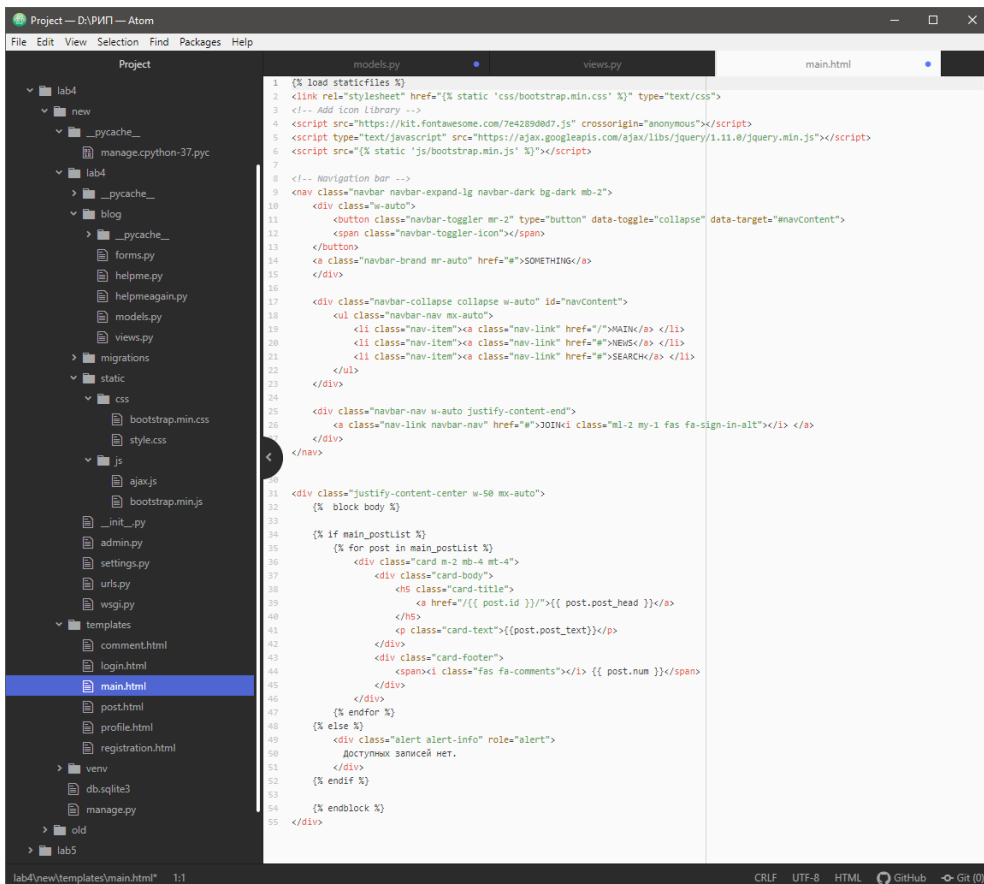
CRLF UTF-8 Python GitHub Git (0)

views.py

The screenshot shows the Atom editor interface with a project named 'lab4'. The left sidebar displays a file tree with folders 'lab4', 'new', and 'old'. The 'lab4' folder contains subfolders like 'blog' and 'static', and files like 'views.py' which is currently selected. The main editor pane shows the code for 'views.py'. It defines three Django view classes: 'MainView' (a generic ListView), 'PostView' (a generic DeleteView), and 'RegistrationForm' (a FormView). It also includes a 'LoginForm' class and a 'user_profile' function. The code is written in Python and uses Django's ORM and rendering functions.

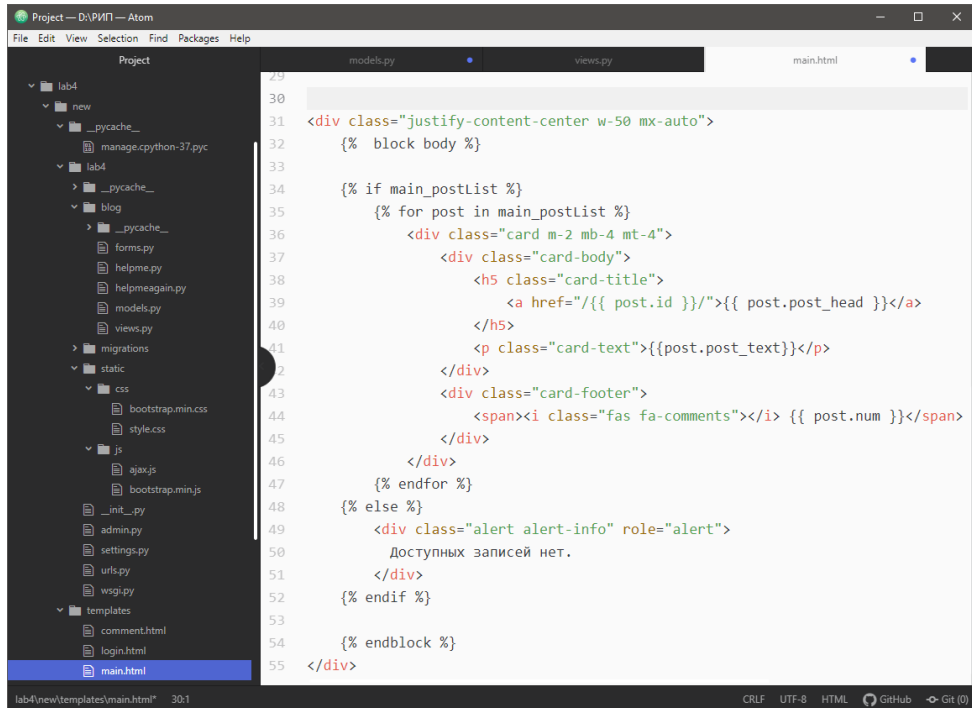
```
14
15 class MainView(generic.ListView):
16     template_name = 'main.html'
17     context_object_name = 'main_postlist'
18
19     def get_queryset(self):
20         return Post.objects.annotate(num=Count('comment')).order_by('-publication_date')[:5]
21
22
23 class PostView(generic.DeleteView):
24     model = Post
25     template_name = 'post.html'
26
27
28 class RegistrationForm(FormView):
29     form_class = RegistrationForm
30     success_url = "/login/"
31     template_name = "registration.html"
32
33     def form_valid(self, form):
34         form.save()
35         return super(RegistrationForm, self).form_valid(form)
36
37     def form_invalid(self, form):
38         return super(RegistrationForm, self).form_invalid(form)
39
40
41 class LoginForm(FormView):
42     form_class = LoginForm
43     success_url = "/profile/"
44     template_name = "login.html"
45
46     def form_valid(self, form):
47         self.user = form.get_user()
48         login(self.request, self.user)
49         return super(LoginForm, self).form_valid(form)
50
51
52 @login_required(login_url='/login/')
53 def user_profile(request):
54     return render(request, 'profile.html')
55
```

main.html

The screenshot shows the Atom editor interface with the same project 'lab4'. The left sidebar shows the file tree, and 'main.html' is now selected under the 'templates' folder. The main editor pane displays the HTML code for 'main.html'. It uses Django's template language to load static files, include Bootstrap and jQuery, and render a navigation bar. The main content area uses a loop to iterate over 'main_postlist' and render each post as a card. The footer includes a Django debug message.

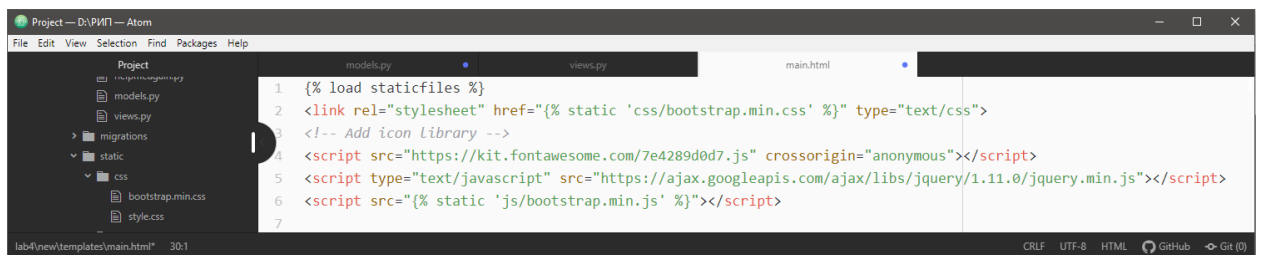
```
1 {% load staticfiles %}
2 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}" type="text/css">
3 <!-- Add icon library -->
4 <script src="https://kit.fontawesome.com/7e4289d0d7.js" crossorigin="anonymous"></script>
5 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
6 <script src="{% static 'js/bootstrap.min.js' %}"></script>
7
8 <!-- Navigation bar -->
9 <nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-2">
10     <div class="w-auto">
11         <button class="navbar-toggler mr-2" type="button" data-toggle="collapse" data-target="#navContent">
12             <span class="navbar-toggler-icon"></span>
13         </button>
14         <a class="navbar-brand mr-auto" href="#">SOMETHING</a>
15     </div>
16
17     <div class="navbar-collapse collapse w-auto" id="navContent">
18         <ul class="navbar-nav mx-auto">
19             <li class="nav-item"><a class="nav-link" href="/#MAIN/"></li>
20             <li class="nav-item"><a class="nav-link" href="/#NEWS/"></li>
21             <li class="nav-item"><a class="nav-link" href="/#SEARCH/"></li>
22         </ul>
23     </div>
24
25     <div class="navbar-nav w-auto justify-content-end">
26         <a class="nav-link navbar-nav" href="/#JOIN/" class="ml-2 my-1 fas fa-sign-in-alt"></a>
27     </div>
28 </nav>
29
30 <div class="justify-content-center w-50 mx-auto">
31     {% block body %}
32
33     {% if main_postlist %}
34         {% for post in main_postlist %}
35             <div class="card m-2 mb-4 mt-4">
36                 <div class="card-body">
37                     <h5 class="card-title">
38                         <a href="{% post.post_head %}">
39                     </h5>
40                     <p class="card-text">{{post.post_text}}</p>
41                 </div>
42                 <div class="card-footer">
43                     <span><i class="fas fa-comments"></i> {{ post.num }}</span>
44                 </div>
45             </div>
46         {% endfor %}
47     {% else %}
48         <div class="alert alert-info" role="alert">
49             Django not running.
50         </div>
51     {% endif %}
52
53     {% endblock %}
54 </div>
55
```

Наследования шаблонов, тегов и переменных:



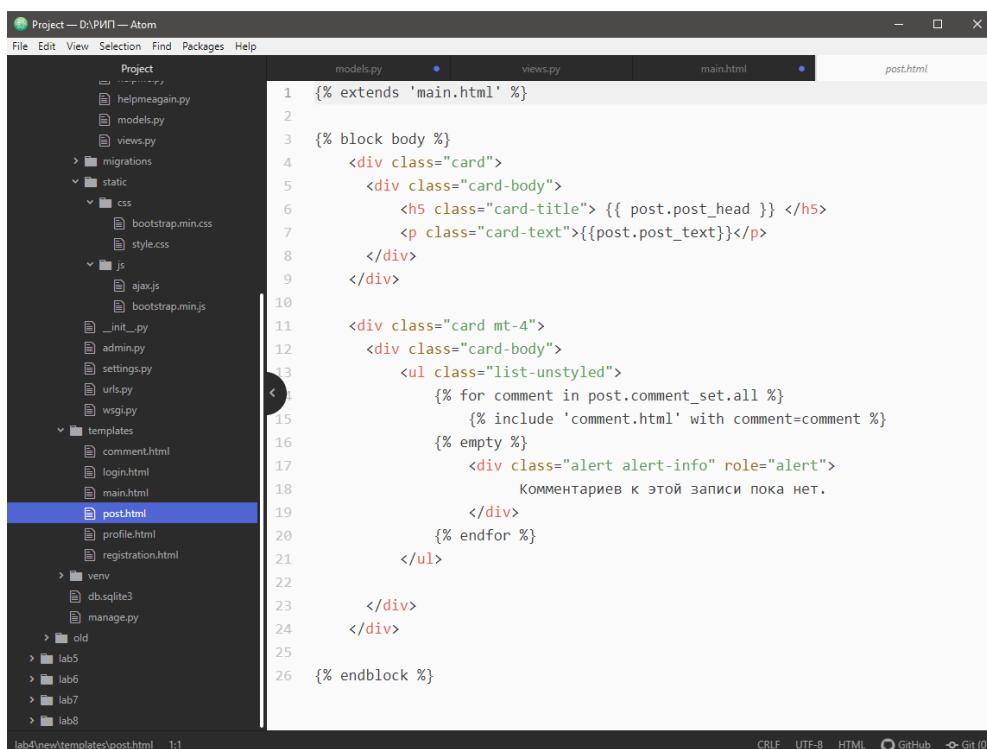
```
30
31 <div class="justify-content-center w-50 mx-auto">
32     {% block body %}
33
34     {% if main_postList %}
35         {% for post in main_postList %}
36             <div class="card m-2 mb-4 mt-4">
37                 <div class="card-body">
38                     <h5 class="card-title">
39                         <a href="{% post.id %}"{{ post.post_head }}</a>
40                     </h5>
41                     <p class="card-text">{{post.post_text}}</p>
42                 </div>
43                 <div class="card-footer">
44                     <span><i class="fas fa-comments"></i> {{ post.num }}</span>
45                 </div>
46             </div>
47         {% endfor %}
48     {% else %}
49         <div class="alert alert-info" role="alert">
50             Доступных записей нет.
51         </div>
52     {% endif %}
53
54     {% endblock %}
55 </div>
```

Подключение статических файлов:



```
1 {% load staticfiles %}
2 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}" type="text/css">
3 <!-- Add icon Library -->
4 <script src="https://kit.fontawesome.com/7e4289d0d7.js" crossorigin="anonymous"></script>
5 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
6 <script src="{% static 'js/bootstrap.min.js' %}"></script>
7
```

post.html



```
1 {% extends 'main.html' %}
2
3 {% block body %}
4     <div class="card">
5         <div class="card-body">
6             <h5 class="card-title"> {{ post.post_head }} </h5>
7             <p class="card-text">{{post.post_text}}</p>
8         </div>
9     </div>
10
11     <div class="card mt-4">
12         <div class="card-body">
13             <ul class="list-unstyled">
14                 {% for comment in post.comment_set.all %}
15                     {% include 'comment.html' with comment=comment %}
16                 {% empty %}
17                     <div class="alert alert-info" role="alert">
18                         Комментариев к этой записи пока нет.
19                     </div>
20                 {% endfor %}
21             </ul>
22         </div>
23     </div>
24 </div>
25
26 {% endblock %}
```

comment.html



The screenshot shows the Atom editor interface with a project named "Project — D:\PIП". The left sidebar displays a file tree with the following structure:

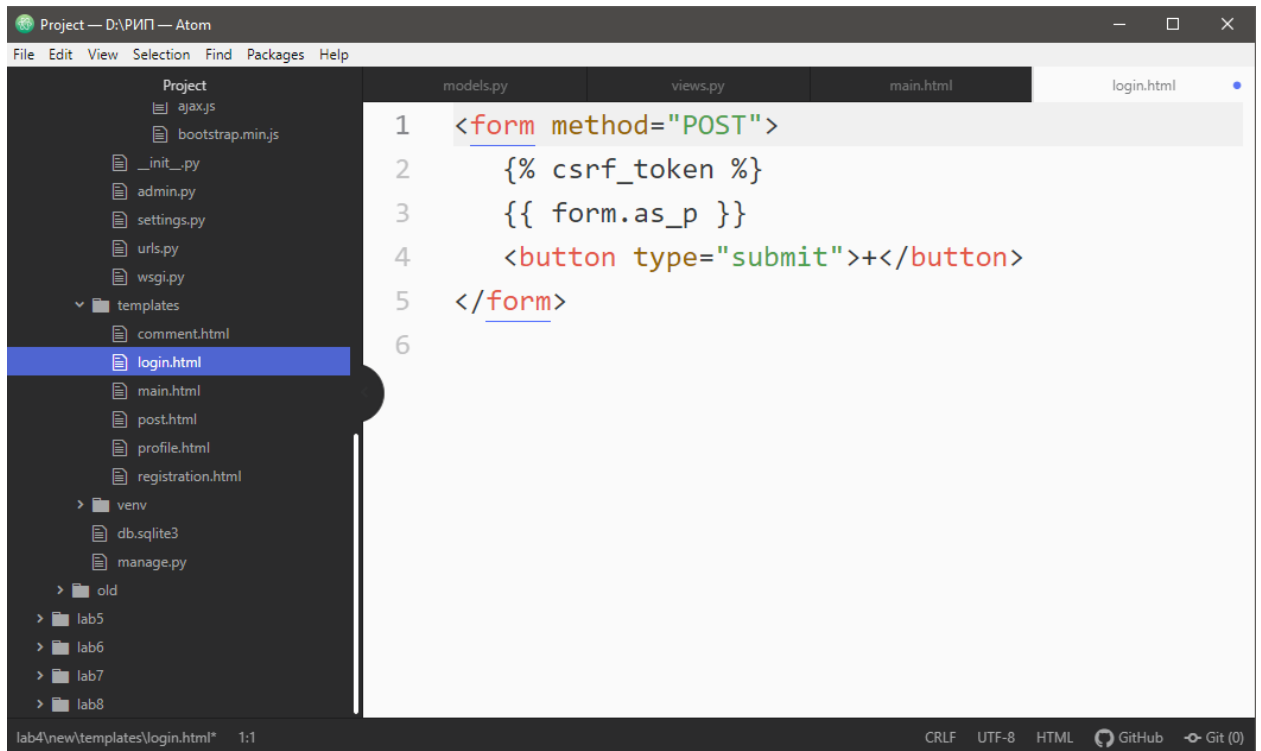
- Project
 - admin.py
 - settings.py
 - urls.py
 - wsgi.py
 - templates
 - comment.html**
 - login.html
 - main.html
 - post.html
 - profile.html
 - registration.html
 - venv
 - db.sqlite3

The main editor area shows the content of `comment.html` with the following code:

```
1 {% load static %}
2 <link rel="stylesheet" href="{% static 'css/style.css' %}" type="text/css">
3 <script src="{% static 'js/ajax.js' %}"></script>
4
5 <li class="mb-3">
6     <div class="col-4 h-25 p-0 text-muted">
7         <span id="{{ comment.id }}">{{ comment.rating }}</span>
8         <span class="commentRating m-0 ml-1 ajax" onclick="loadRating('{{ comment.id }}')"><i class="fas fa-caret-up"></i></span>
9         <span class="commentRating m-0 ajax" onclick="loadRating('{{ comment.id }}')"><i class="fas fa-caret-down"></i></span>
10     </div>
11     {{ comment.answer_text }}
12 </li>
```

The status bar at the bottom indicates the file path `lab4\new\templates\comment.html` and the line number `1:1`. It also shows encoding options (CRLF, UTF-8, HTML) and GitHub integration links.

login.html



The screenshot shows the Atom editor interface with the same project. The left sidebar file tree is expanded to show the `templates` directory, with `login.html` selected. The main editor area shows the content of `login.html` with the following code:

```
1 <form method="POST">
2     {% csrf_token %}
3     {{ form.as_p }}
4     <button type="submit"></button>
5 </form>
6
```

The status bar at the bottom indicates the file path `lab4\new\templates\login.html` and the line number `1:1`. It also shows encoding options (CRLF, UTF-8, HTML) and GitHub integration links.

И т.д.