



Django Projekt anlegen



Einleitung



pythonTM


django

Kapitel II
Projekt anlegen

Lernziele Kapitel II:

- Ich kann Apps zum Django Projekt hinzufügen.
- Ich habe verstanden was eine Django App ist.
- Ich habe die wichtigsten Dateien innerhalb einer App verstanden.
- Ich habe verstanden wie Django eine Website rendert.

Anlegen des Projektes

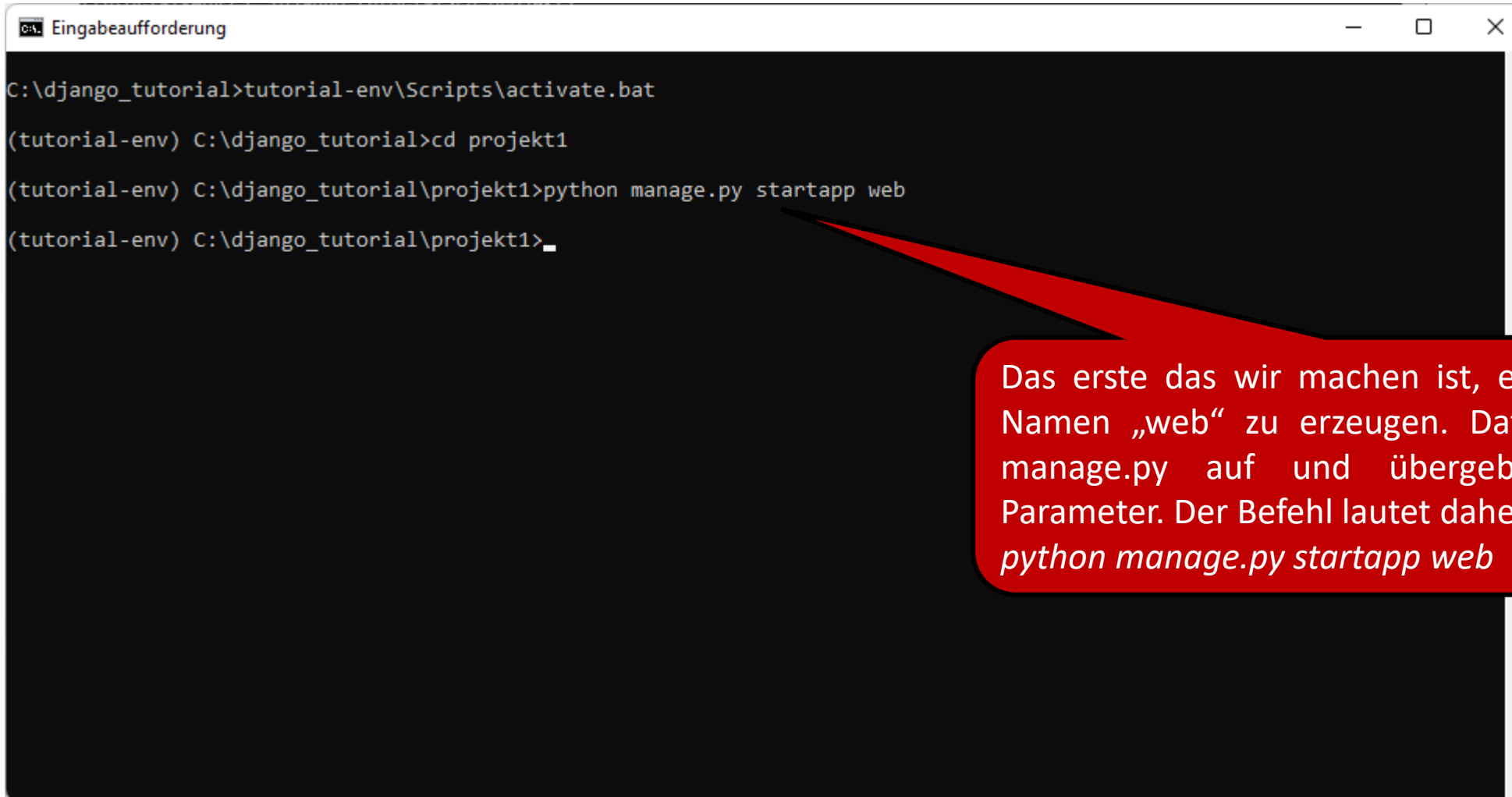
A red speech bubble with a black outline and a tail pointing towards the top-left. It contains white text.

Wir wollen als Einstieg in Django ein kleines Projekt umsetzen schreiben.

Anlegen des Projektes

Aufgabe: Es soll ein Programm geschrieben werden in dem ein Lager verwaltet werden kann. Der Laderbestand soll dabei nur von Registrierten Usern, eingesehen und verändert werden.

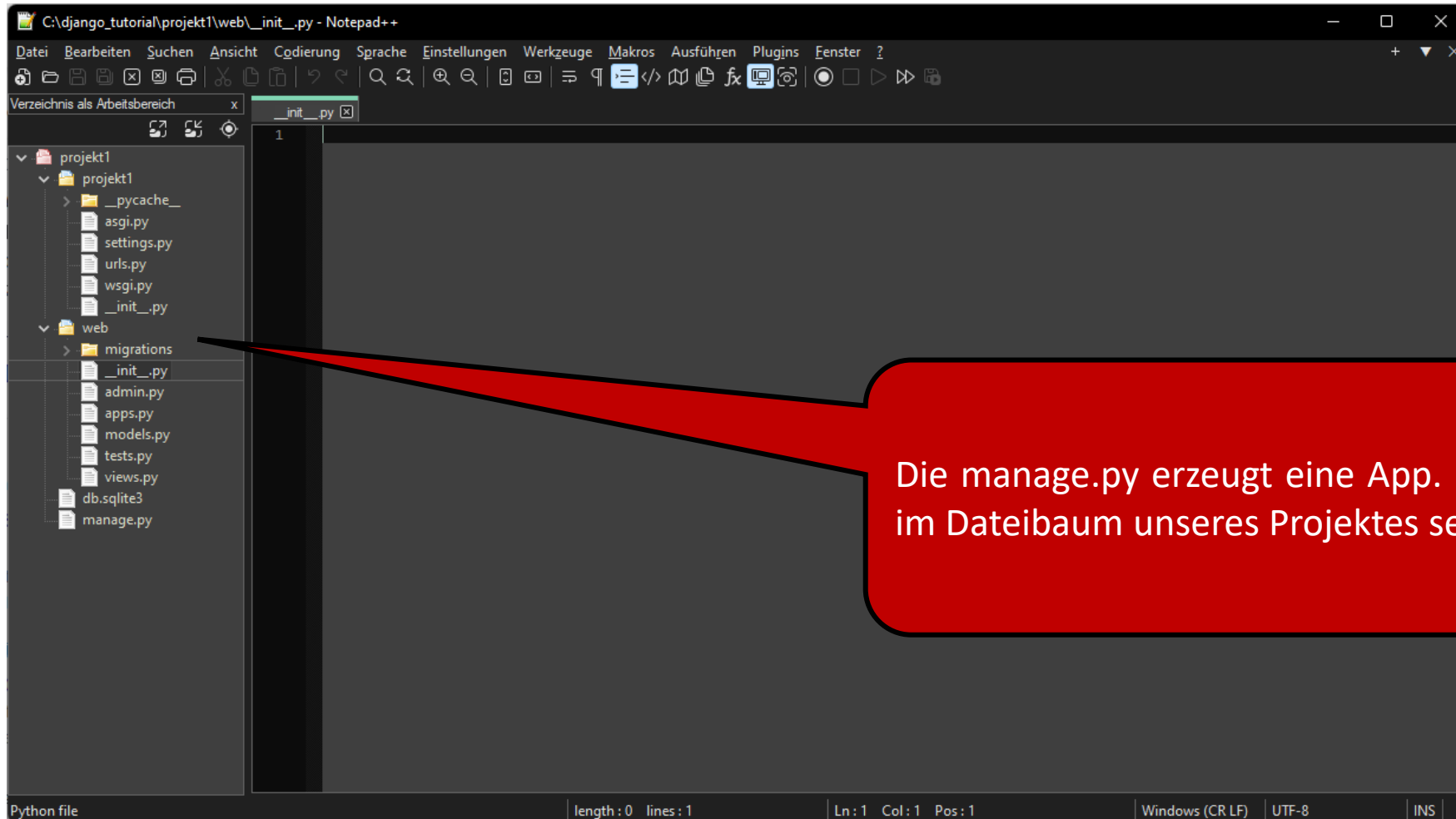
Anlegen des Projektes



```
C:\django_tutorial>tutorial-env\Scripts\activate.bat
(tutorial-env) C:\django_tutorial>cd projekt1
(tutorial-env) C:\django_tutorial\projekt1>python manage.py startapp web
(tutorial-env) C:\django_tutorial\projekt1>_
```

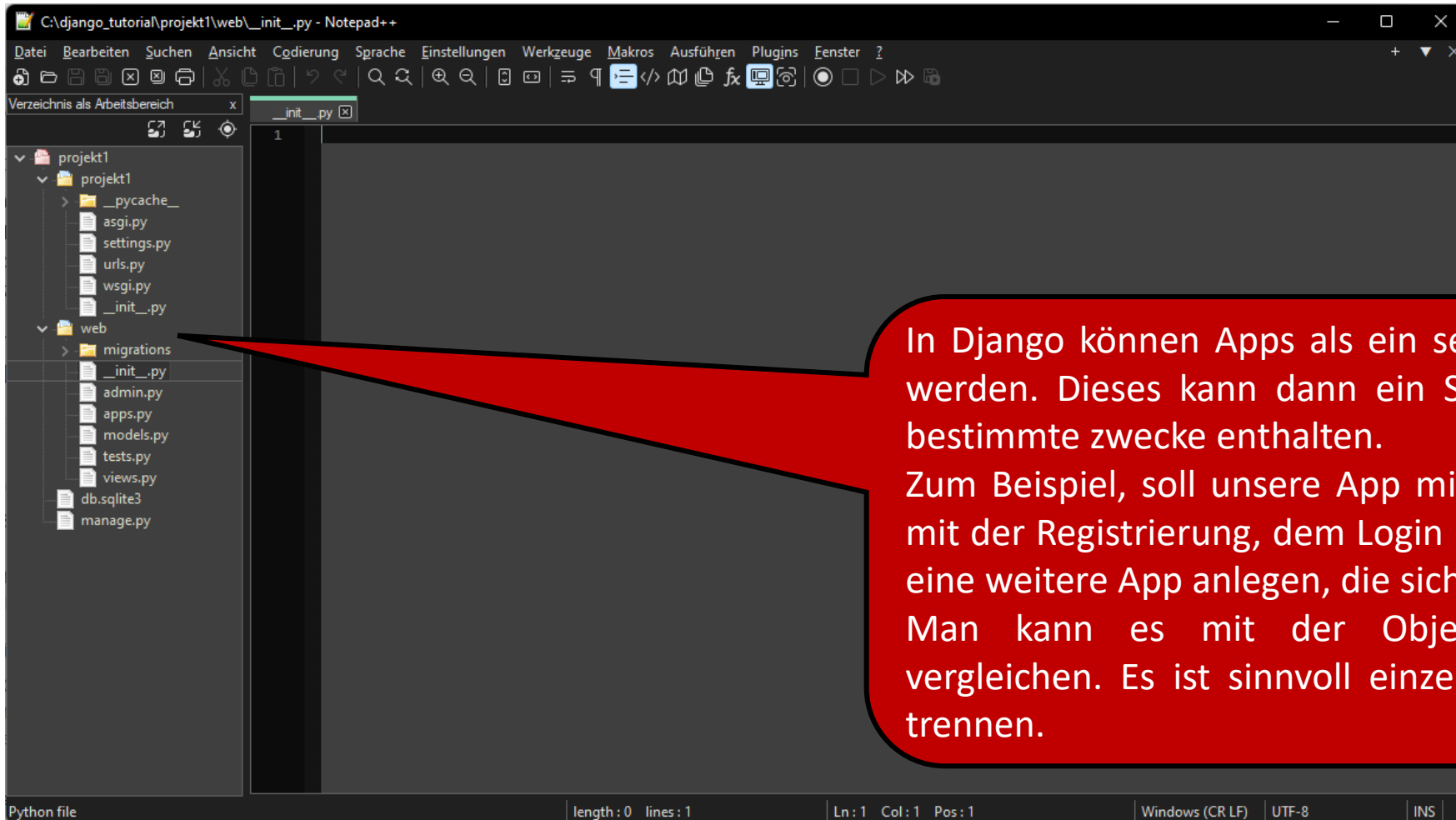
Das erste das wir machen ist, eine App mit dem Namen „web“ zu erzeugen. Dafür rufen wir die `manage.py` auf und übergeben die nötigen Parameter. Der Befehl lautet daher:
python manage.py startapp web

Anlegen des Projektes



Die manage.py erzeugt eine App. Diese können wir im Dateibaum unseres Projektes sehen.

Anlegen des Projektes

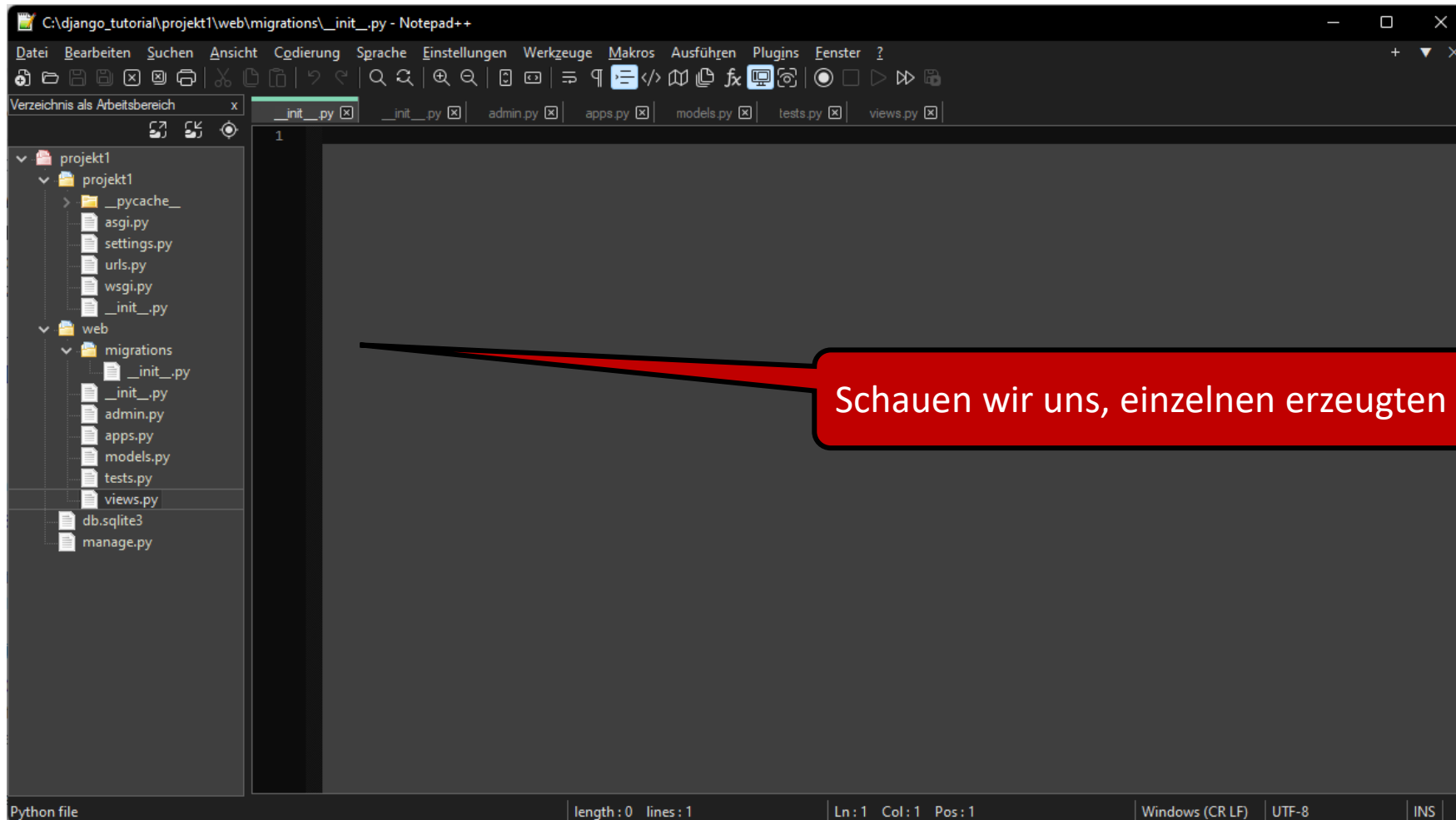


In Django können Apps als ein separates Python-Paket verstanden werden. Dieses kann dann ein Satz von zugehörigen Dateien für bestimmte Zwecke enthalten.

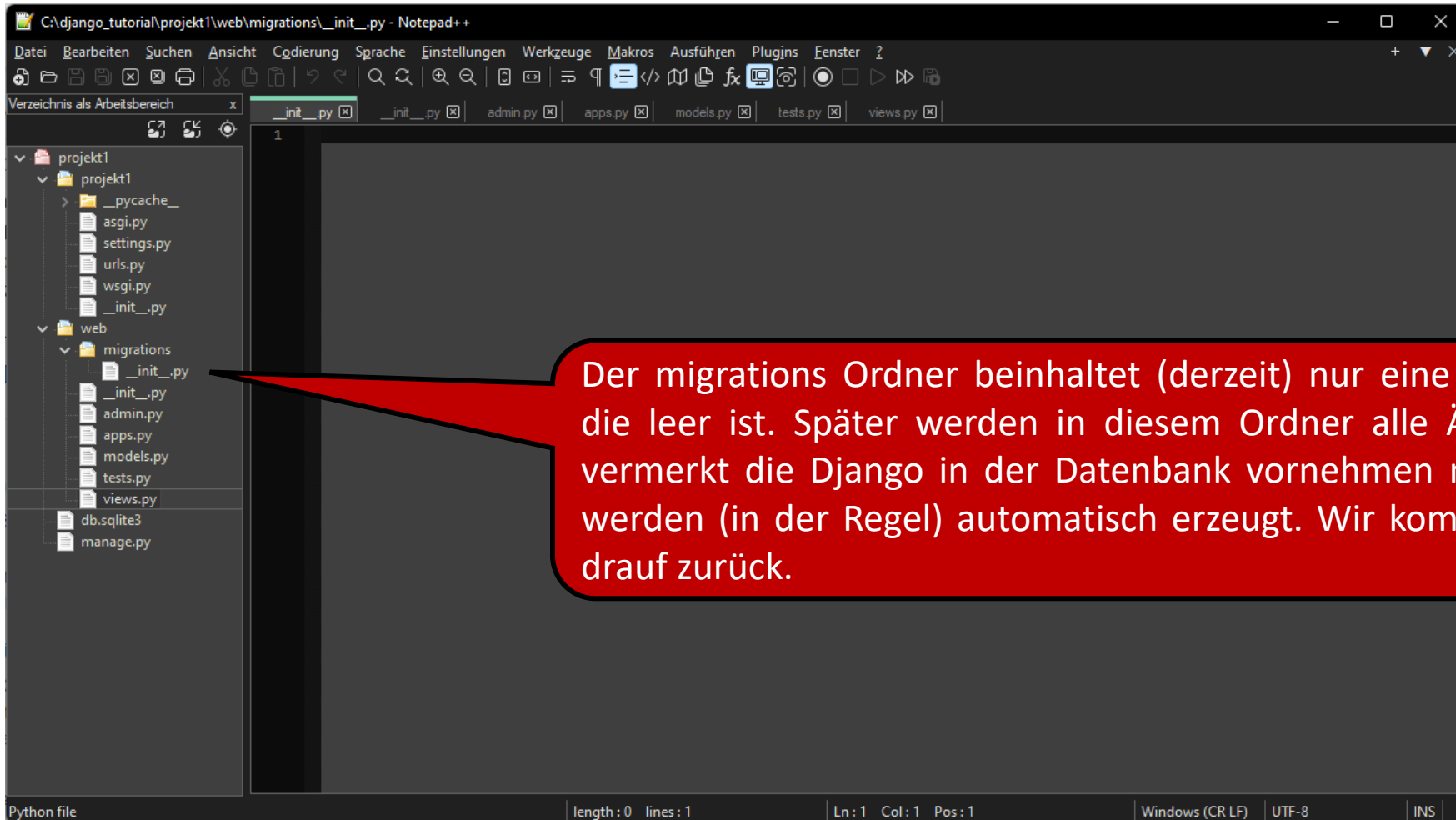
Zum Beispiel, soll unsere App mit dem Namen web, sich um alles mit der Registrierung, dem Login usw. kümmern. Wir werden später eine weitere App anlegen, die sich nur um das Lager kümmert.

Man kann es mit der Objekt Orientierten Programmierung vergleichen. Es ist sinnvoll einzelne Teilbereiche des Projektes zu trennen.

Anlegen des Projektes

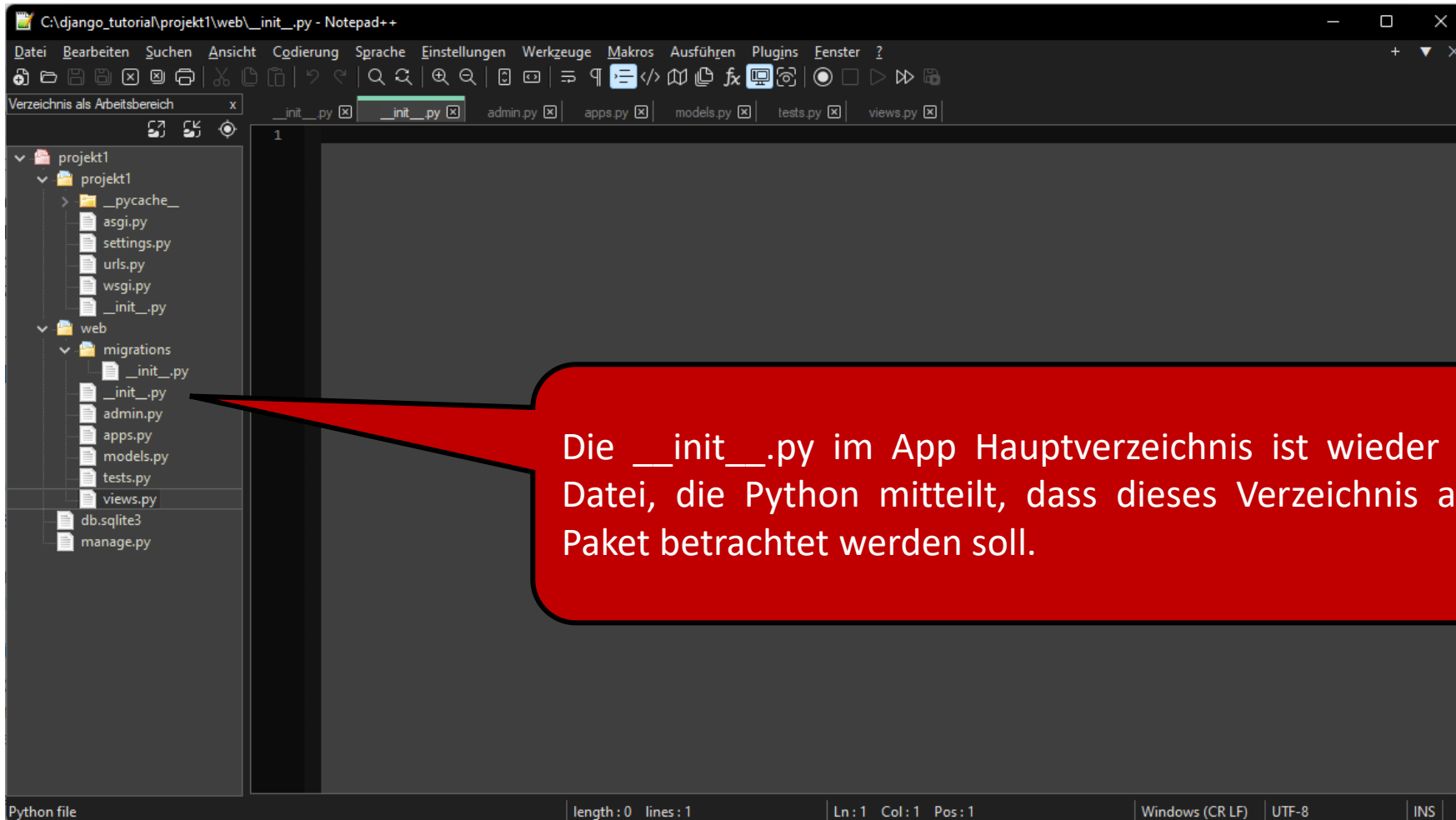


Anlegen des Projektes

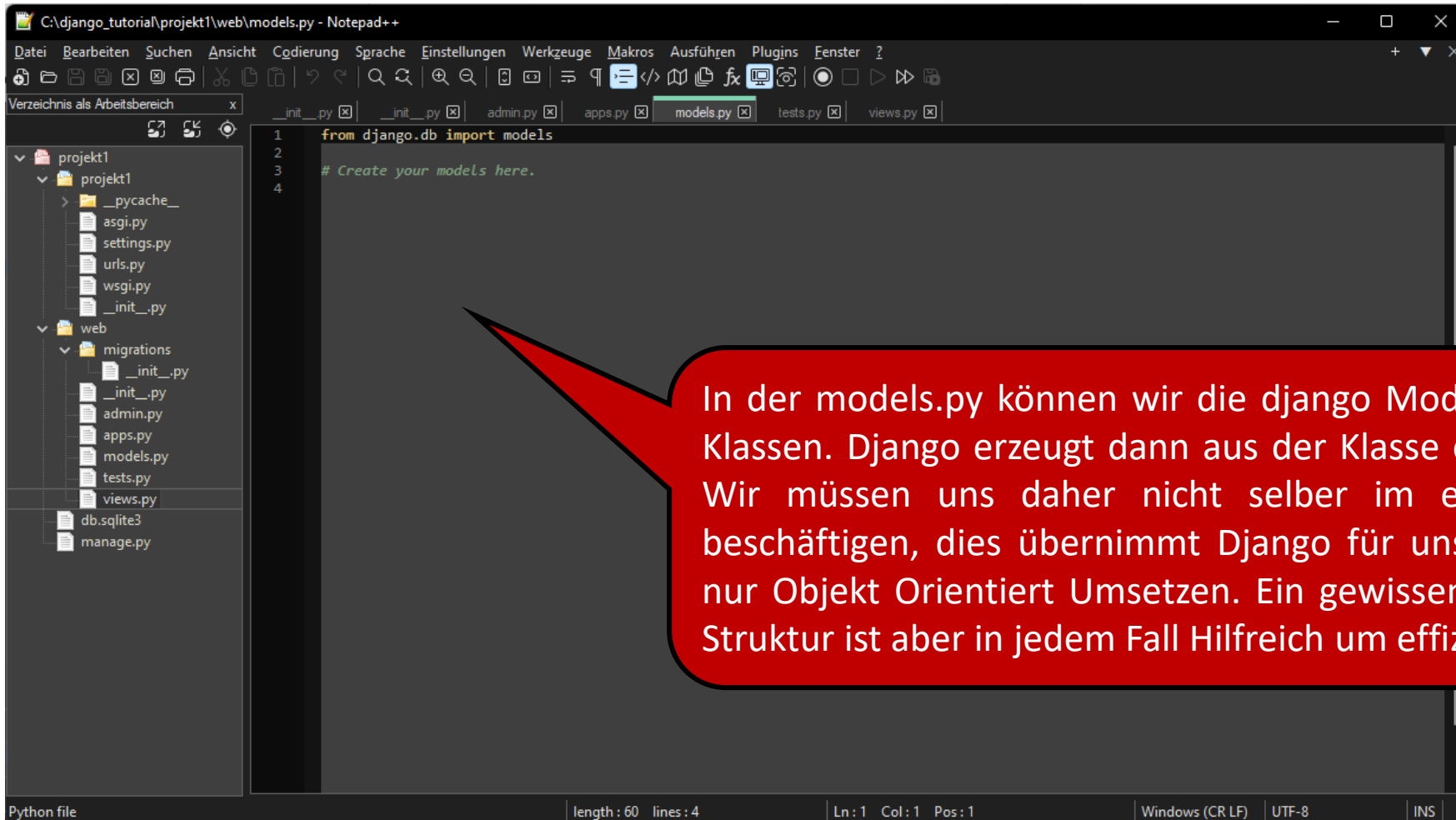


Der migrations Ordner beinhaltet (derzeit) nur eine `__init.py__` die leer ist. Später werden in diesem Ordner alle Änderungen vermerkt die Django in der Datenbank vornehmen muss. Diese werden (in der Regel) automatisch erzeugt. Wir kommen später drauf zurück.

Anlegen des Projektes



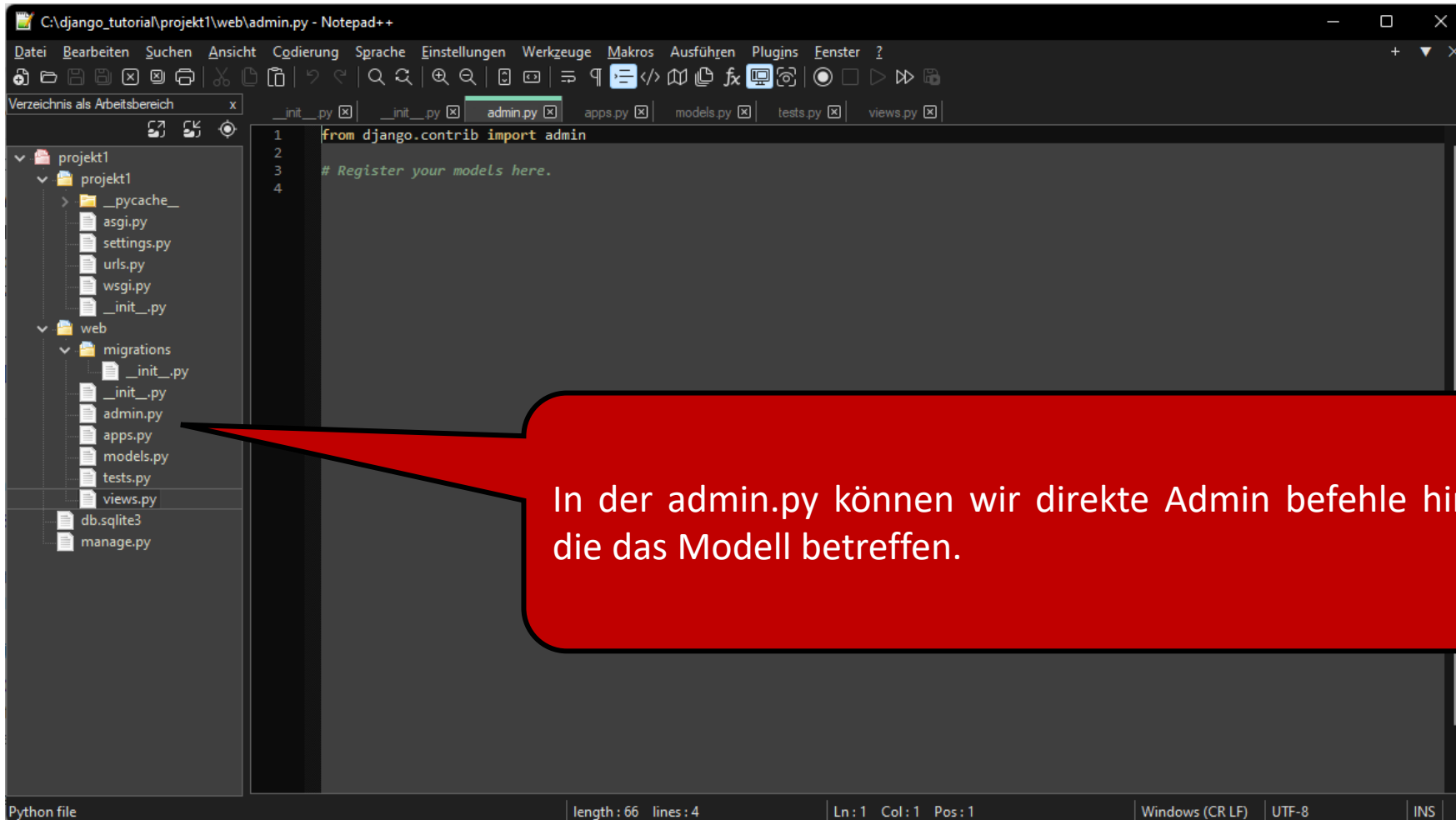
Anlegen des Projektes



```
1 from django.db import models
2
3 # Create your models here.
4
```

In der models.py können wir die django Modells anlegen. Dies sind Python Klassen. Django erzeugt dann aus der Klasse eine Tabelle in der Datenbank. Wir müssen uns daher nicht selber im einzelnen mit der Datenbank beschäftigen, dies übernimmt Django für uns. Wir brauchen unser Projekt nur Objekt Orientiert Umsetzen. Ein gewissen Verständnis einer Datenbank Struktur ist aber in jedem Fall Hilfreich um effizient zu Programmieren.

Anlegen des Projektes



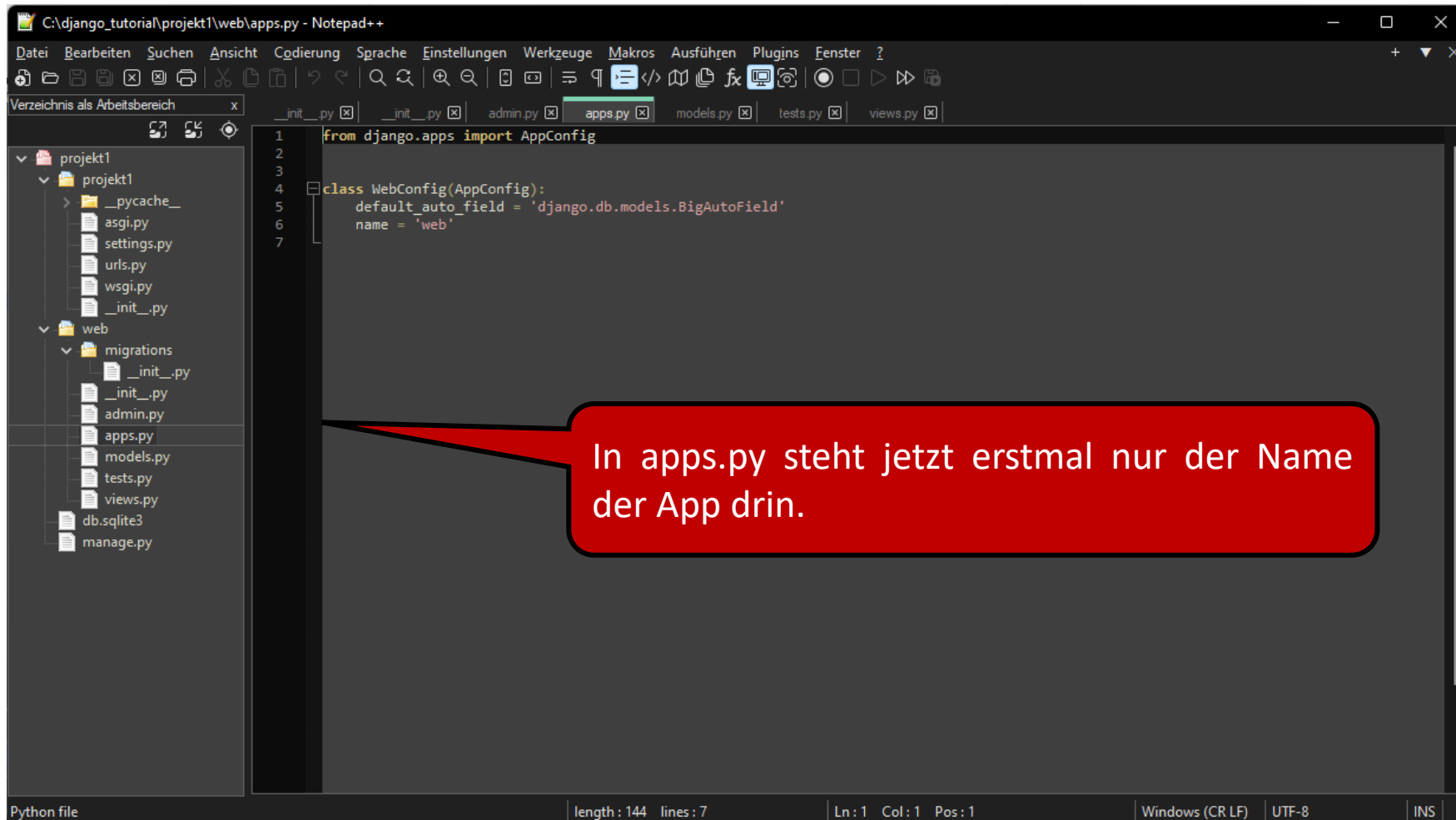
The screenshot shows a Notepad++ window titled "C:\django_tutorial\projekt1\web\admin.py - Notepad++". The left sidebar displays a file explorer for the "projekt1" directory, showing subdirectories like "migrations" and "web", and files like "admin.py", "apps.py", "models.py", "tests.py", and "views.py". The main editor area shows the content of "admin.py":

```
1 from django.contrib import admin
2
3 # Register your models here.
4
```

A red callout box points to the "admin.py" file in the file explorer and contains the text: "In der admin.py können wir direkte Admin befehle hinterlegen, die das Modell betreffen."

The status bar at the bottom indicates "Python file", "length: 66 lines: 4", "Ln: 1 Col: 1 Pos: 1", "Windows (CR LF)", "UTF-8", and "INS".

Anlegen des Projektes



The screenshot shows a Notepad++ window with the file path `C:\django_tutorial\projekt1\web\apps.py`. The left sidebar displays the project structure:

- projekt1
 - __pycache__
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - __init__.py
 - web
 - migrations
 - __init__.py
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - views.py
 - db.sqlite3
 - manage.py

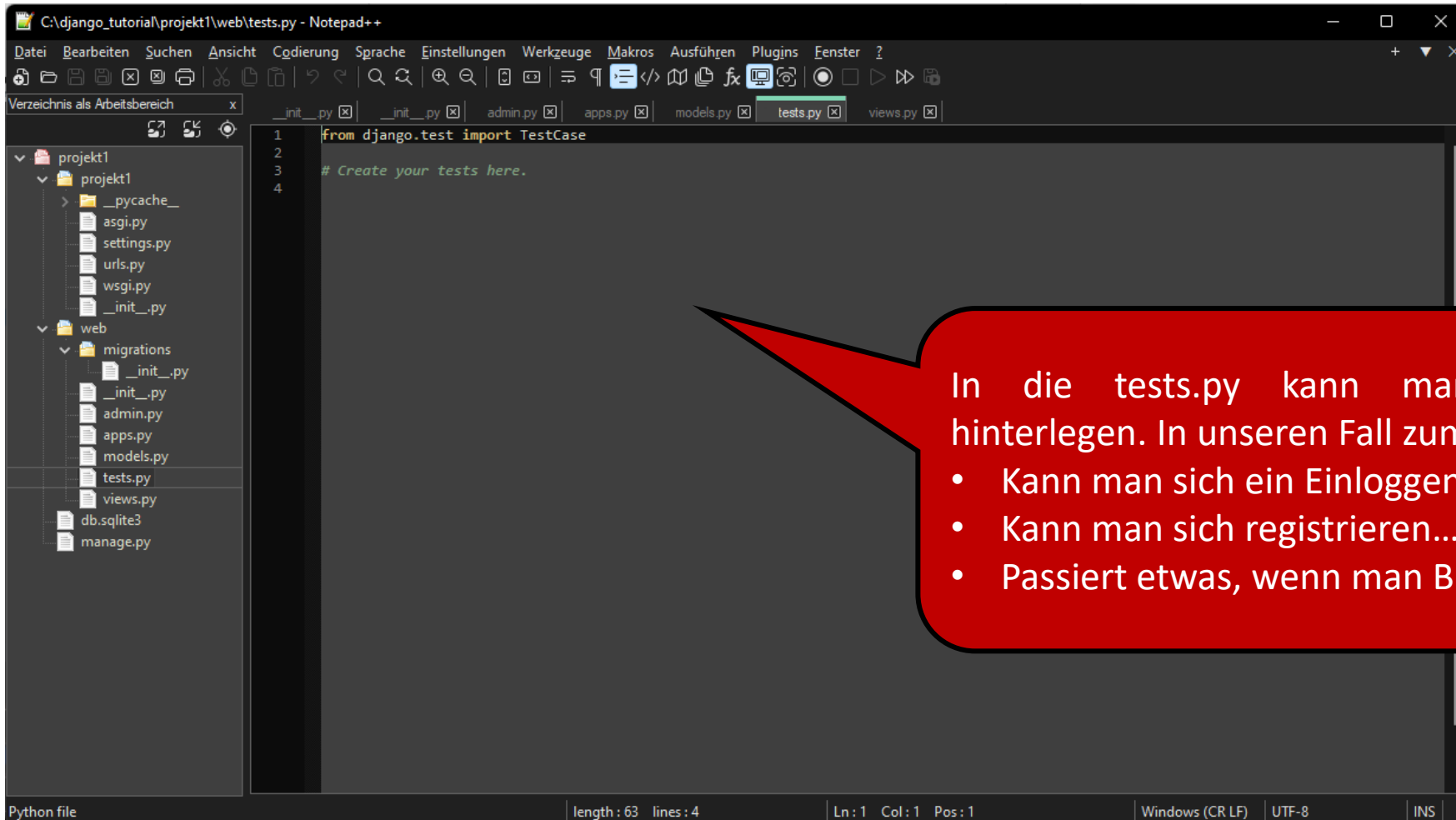
The main editor area shows the following Python code in `apps.py`:

```
1 from django.apps import AppConfig
2
3
4 class WebConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'web'
7
```

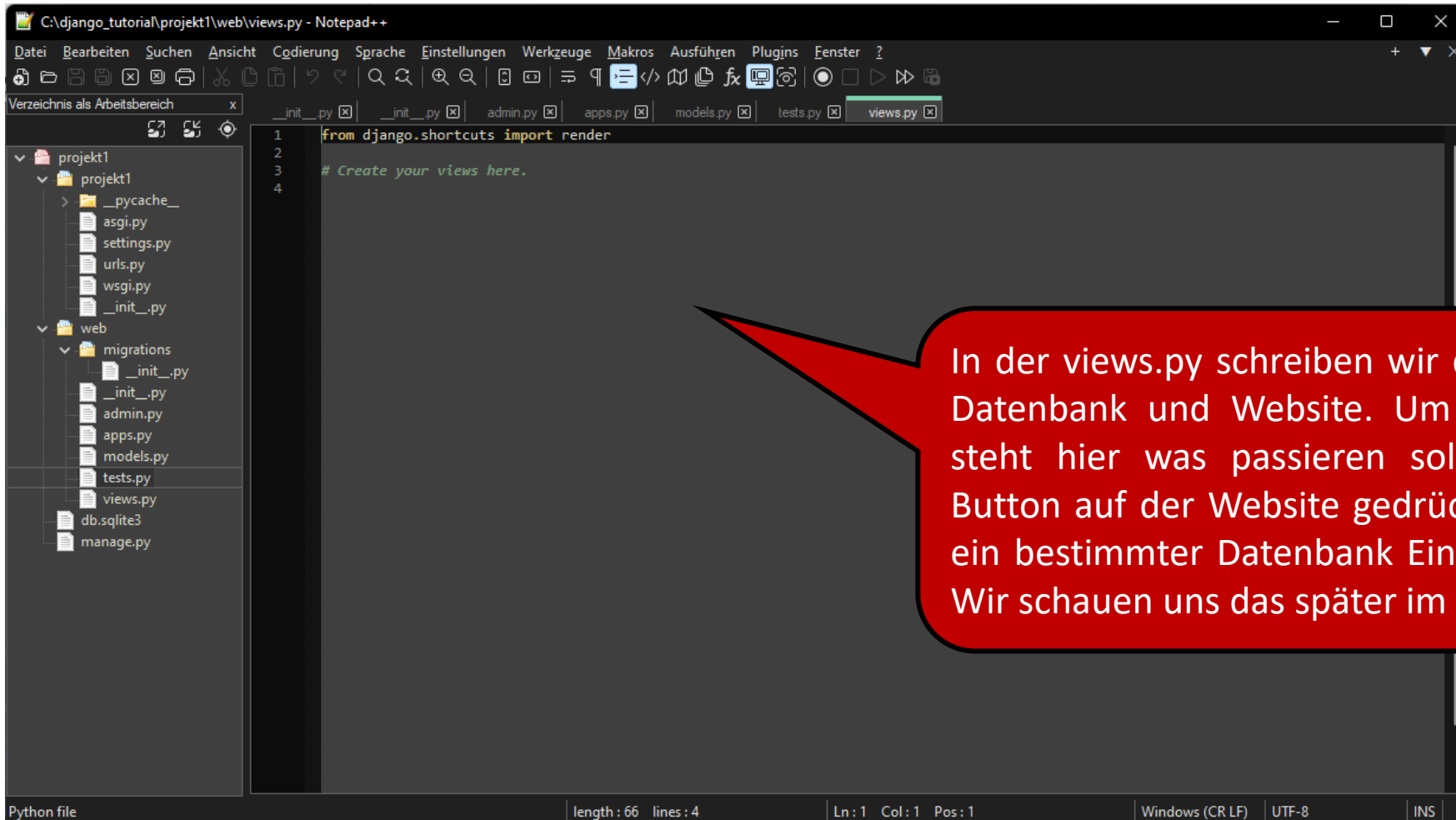
A red callout box points to the `name = 'web'` line with the text: "In apps.py steht jetzt erstmal nur der Name der App drin."

The status bar at the bottom indicates: Python file | length: 144 lines: 7 | Ln: 1 Col: 1 Pos: 1 | Windows (CR LF) | UTF-8 | INS

Anlegen des Projektes



Anlegen des Projektes



The screenshot shows the Notepad++ editor with the Django project structure on the left and the `views.py` file open in the main editor. The project structure includes a `projekt1` directory with subdirectories `__pycache__`, `web`, and `migrations`. The `web` directory contains `asgi.py`, `settings.py`, `urls.py`, `wsgi.py`, and `__init__.py`. The `migrations` directory contains `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `views.py`, `db.sqlite3`, and `manage.py`. The `views.py` file contains the following code:

```
1 from django.shortcuts import render
2
3 # Create your views here.
4
```

A red speech bubble points to the `views.py` file in the project structure, containing the following text:

In der `views.py` schreiben wir die Schnittstelle zwischen Datenbank und Website. Um es allgemein zu halten, steht hier was passieren soll, wenn ein bestimmter Button auf der Website gedrückt werden soll. Oder wie ein bestimmter Datenbank Eintrag gesucht werden soll. Wir schauen uns das später im Detail genauer an.

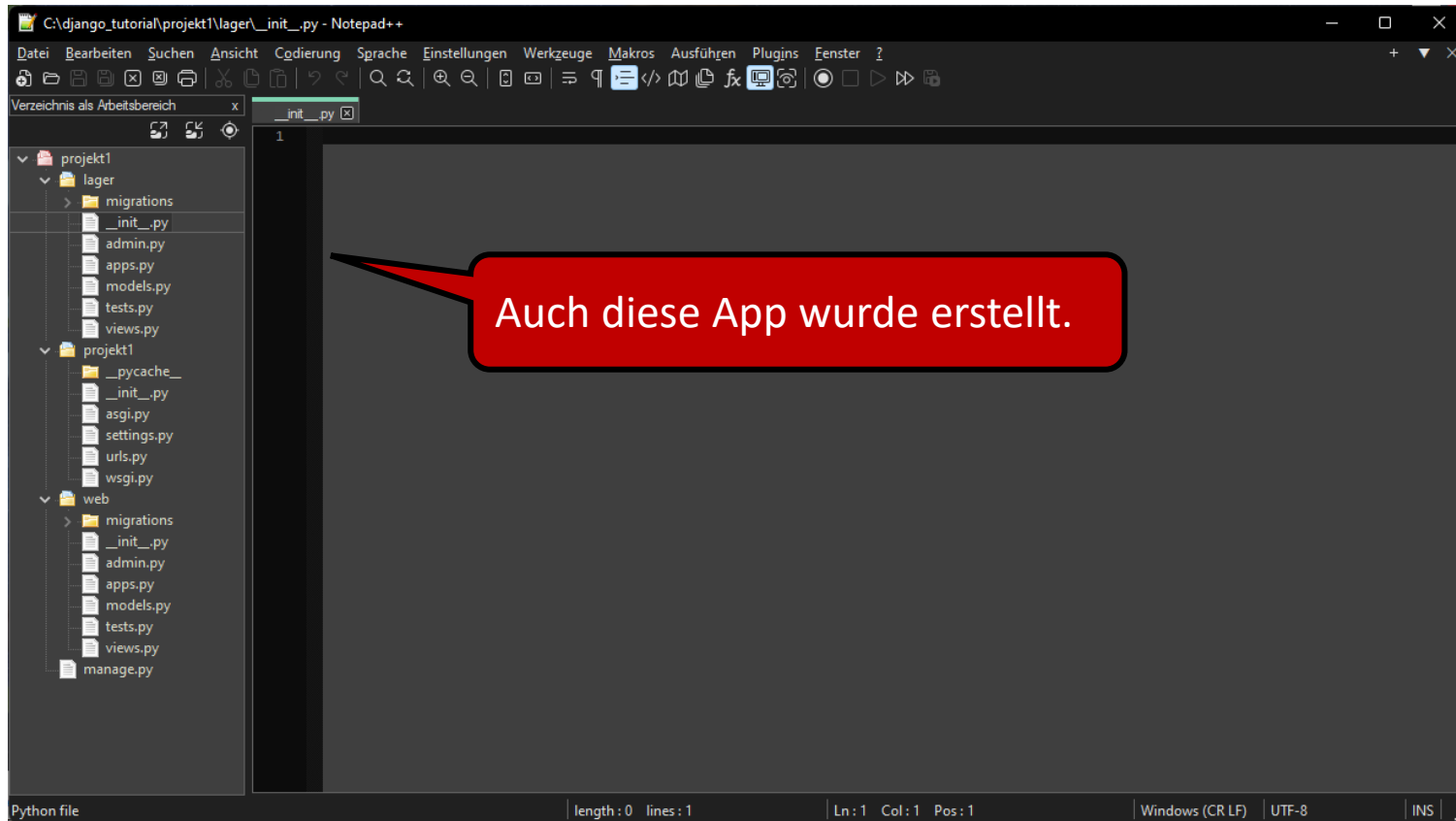
Anlegen des Projektes



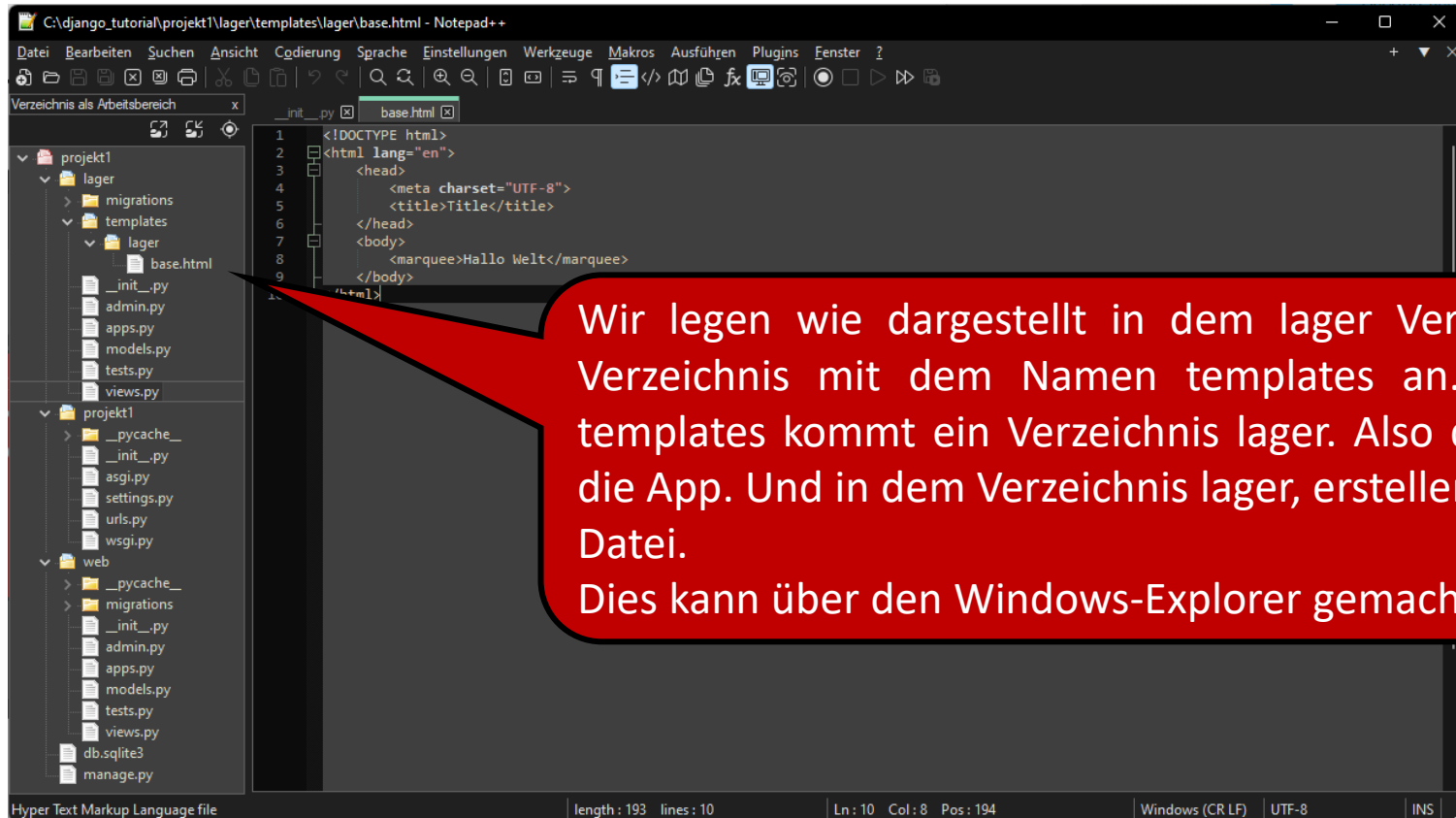
```
C:\django_tutorial>tutorial-env\Scripts\activate.bat
(tutorial-env) C:\django_tutorial>cd projekt1
(tutorial-env) C:\django_tutorial\projekt1>python manage.py startapp web
(tutorial-env) C:\django_tutorial\projekt1>python manage.py startapp lager
(tutorial-env) C:\django_tutorial\projekt1>
```

Wir wechseln wieder in die Konsole und erstellen mit der `manage.py` eine App mit dem Namen *lager*. Hier soll alles rein, das mit dem eigentlichen Lager Einträgen zusammen hängt.

Anlegen des Projektes



Anlegen des Projektes

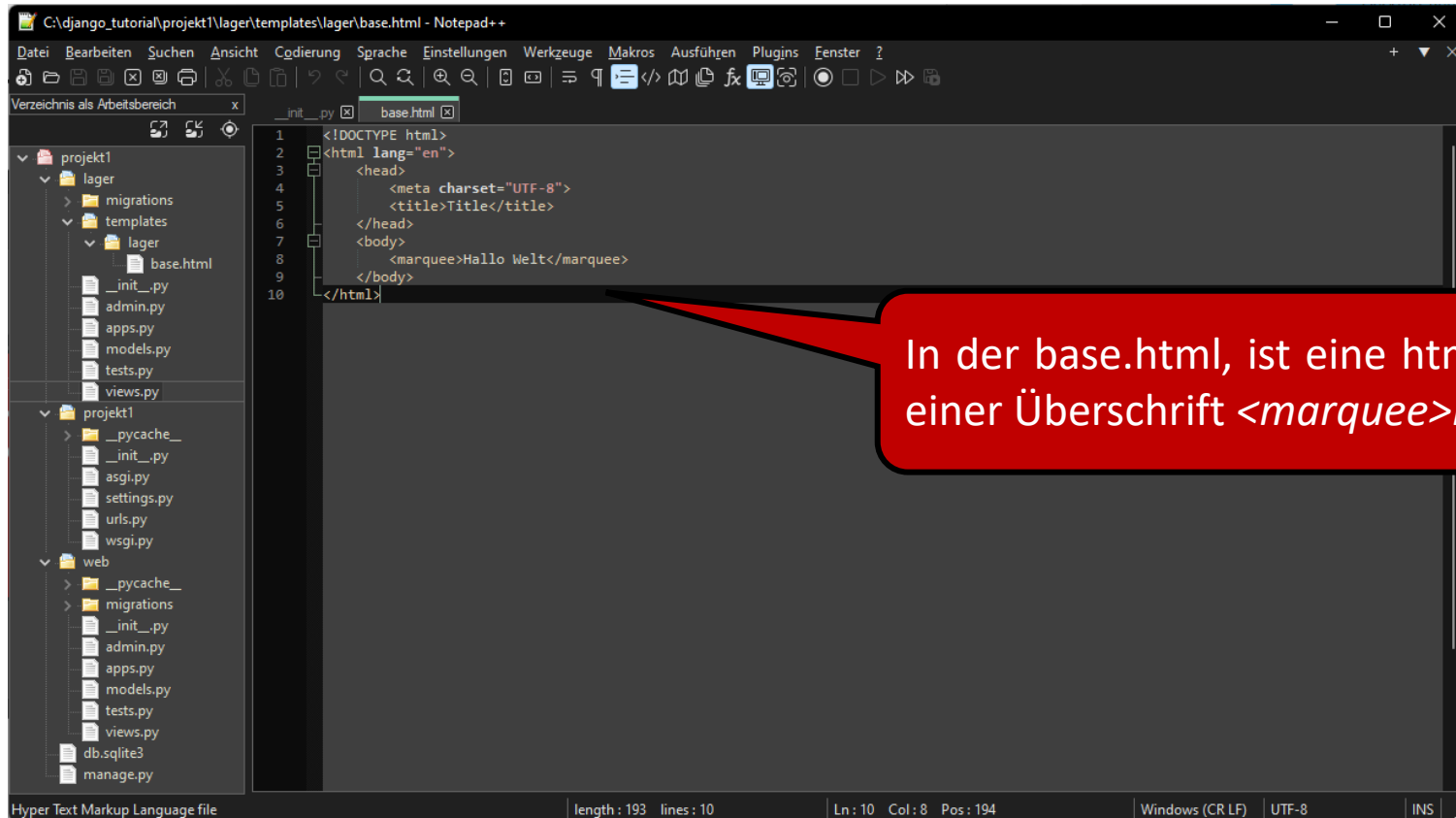


```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6   </head>
7   <body>
8     <marquee>Hallo Welt</marquee>
9   </body>
10 </html>
```

Wir legen wie dargestellt in dem lager Verzeichnis ein weiteres Verzeichnis mit dem Namen templates an. In dem Verzeichnis templates kommt ein Verzeichnis lager. Also der gleiche Name wie die App. Und in dem Verzeichnis lager, erstellen wir eine „base.html“ Datei.

Dies kann über den Windows-Explorer gemacht werden.

Anlegen des Projektes

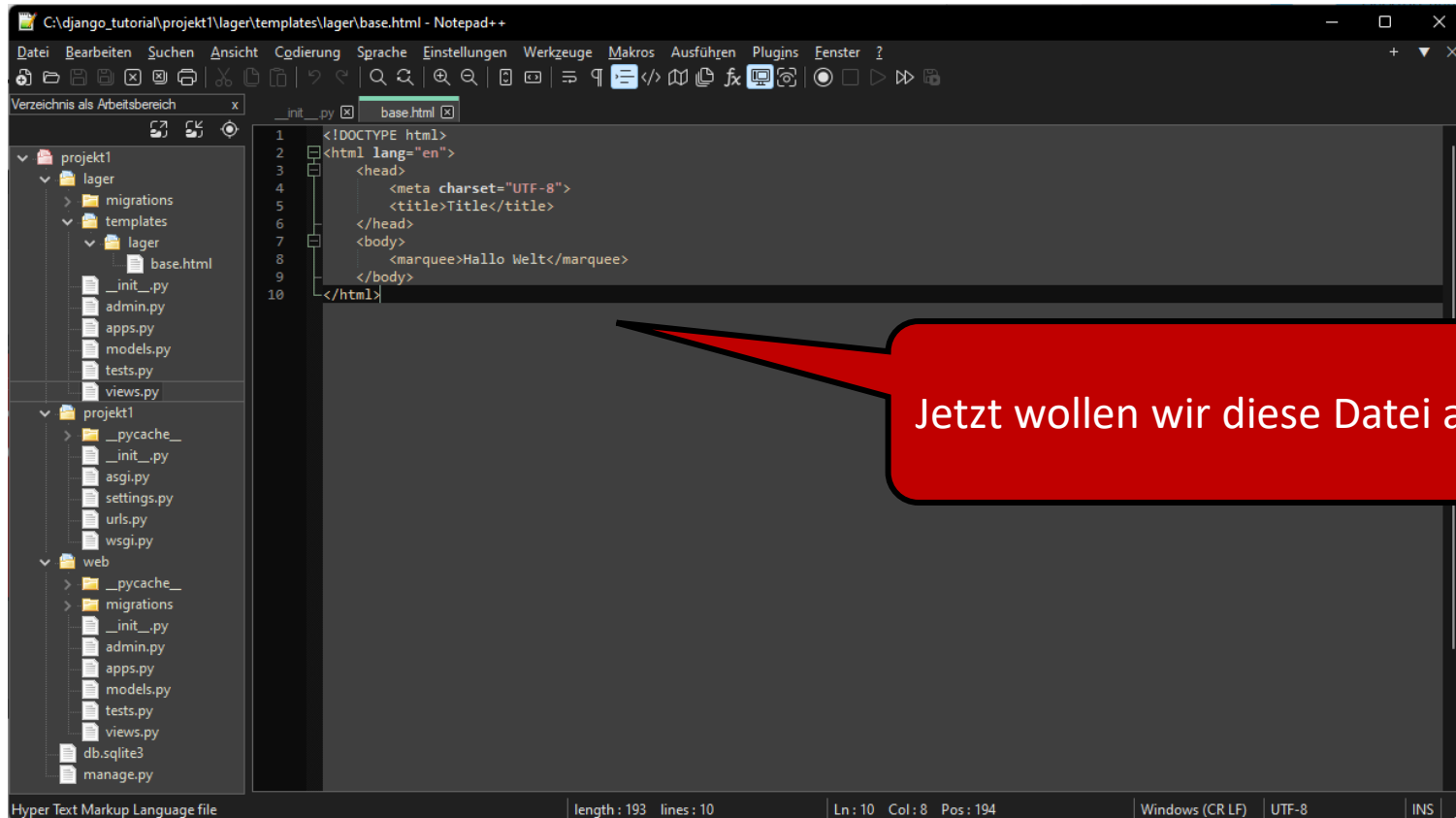


The screenshot shows a Notepad++ window with the file path `C:\django_tutorial\projekt1\lager\templates\lager\base.html`. The left sidebar displays a file explorer with the project structure. The main editor area shows the content of `base.html`:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6   </head>
7   <body>
8     <marquee>Hallo Welt</marquee>
9   </body>
10 </html>
```

A red callout box points to the `<marquee>Hallo Welt</marquee>` line, containing the text: "In der base.html, ist eine html Seite hinterlegt. Erstmal nur mit einer Überschrift `<marquee>Hallo Welt</marquee>`".

Anlegen des Projektes



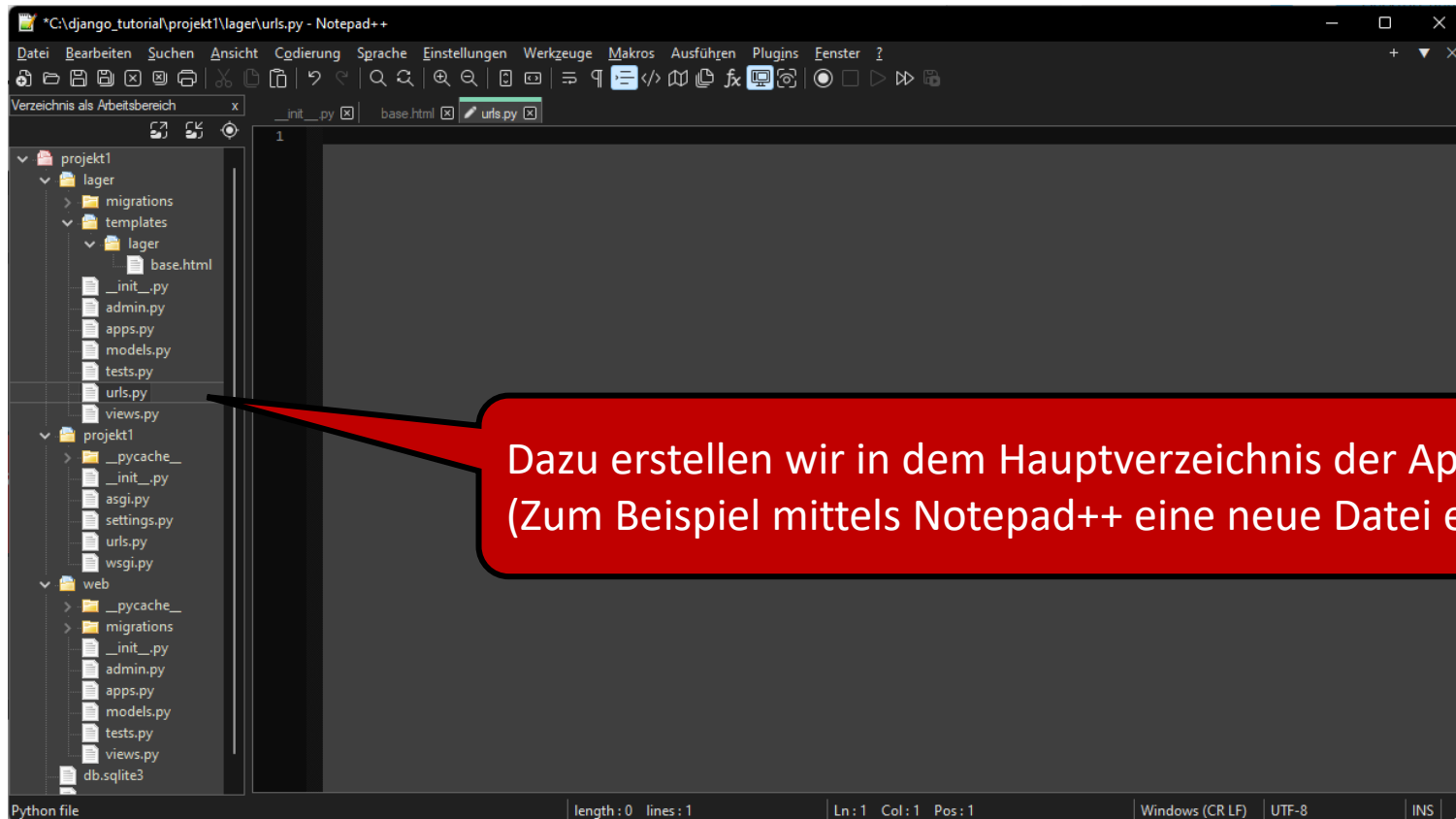
The screenshot shows the Notepad++ editor with the file `base.html` open. The file content is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6   </head>
7   <body>
8     <marquee>Hallo Welt</marquee>
9   </body>
10 </html>
```

A red speech bubble points to the `base.html` file in the project explorer on the left. The text inside the bubble is:

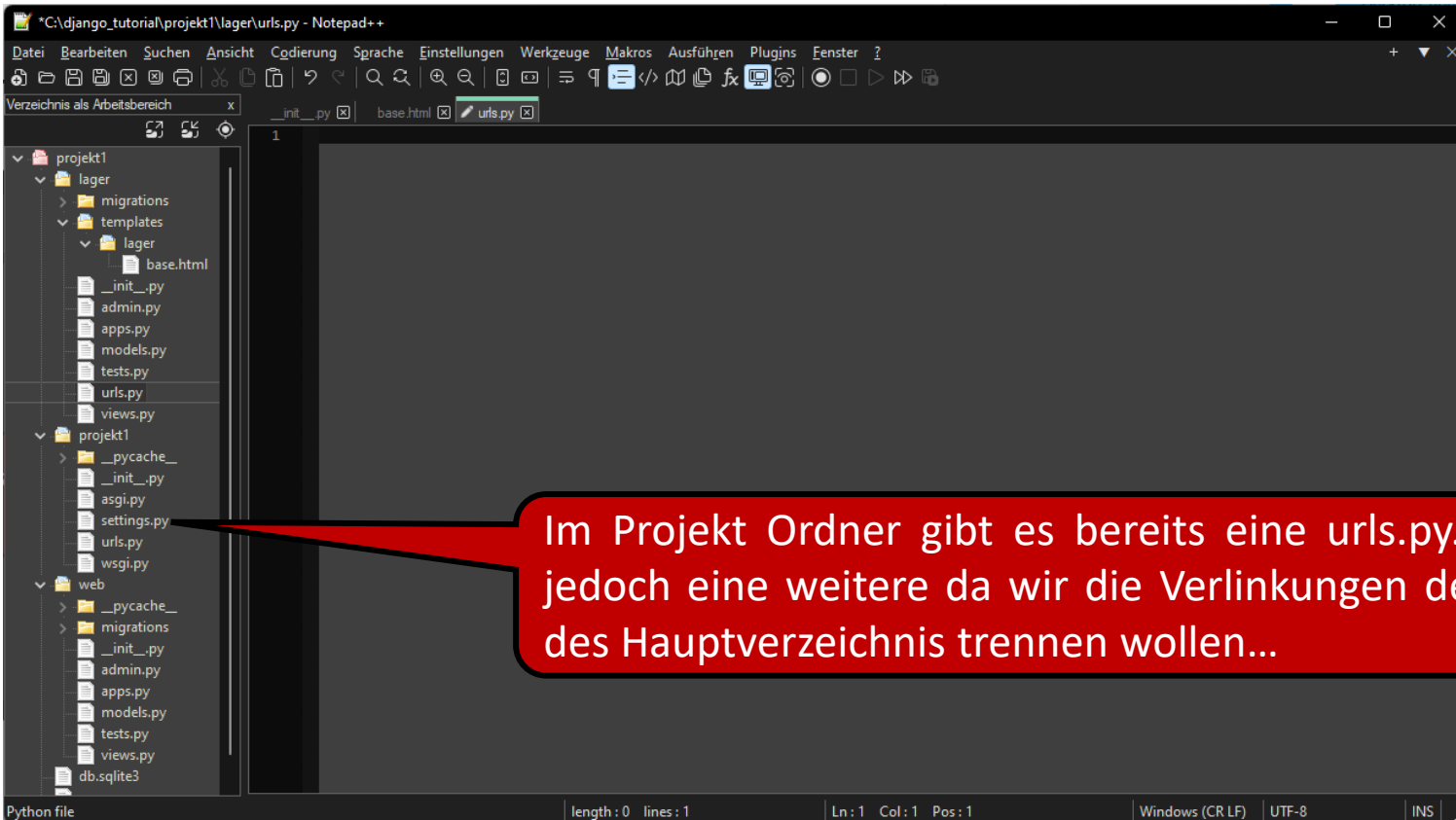
Jetzt wollen wir diese Datei auch aufrufen...

Anlegen des Projektes

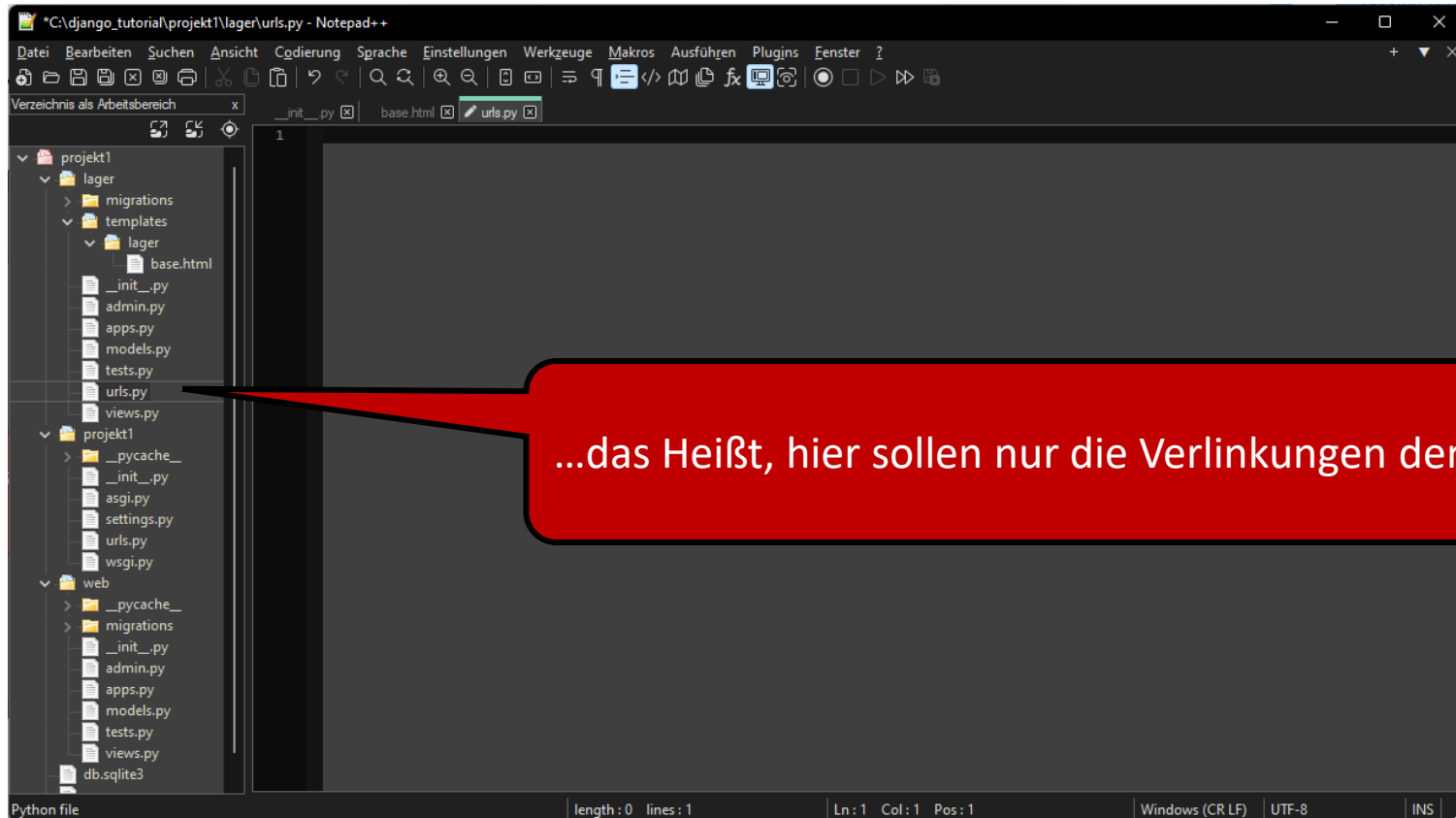


Dazu erstellen wir in dem Hauptverzeichnis der App eine *urls.py*.
(Zum Beispiel mittels Notepad++ eine neue Datei erstellen).

Anlegen des Projektes

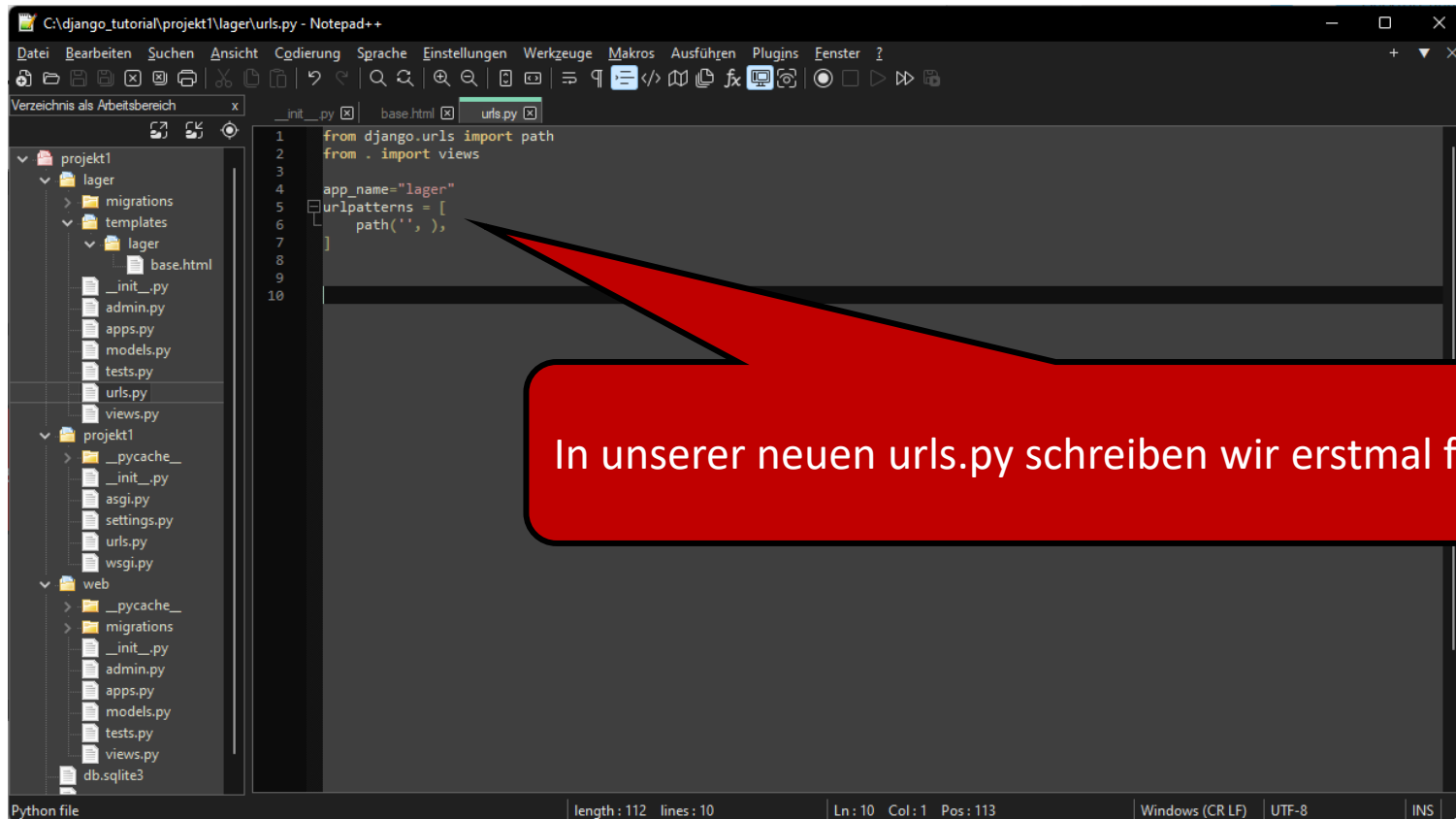


Anlegen des Projektes



...das heißt, hier sollen nur die Verlinkungen der App lager rein.

Anlegen des Projektes

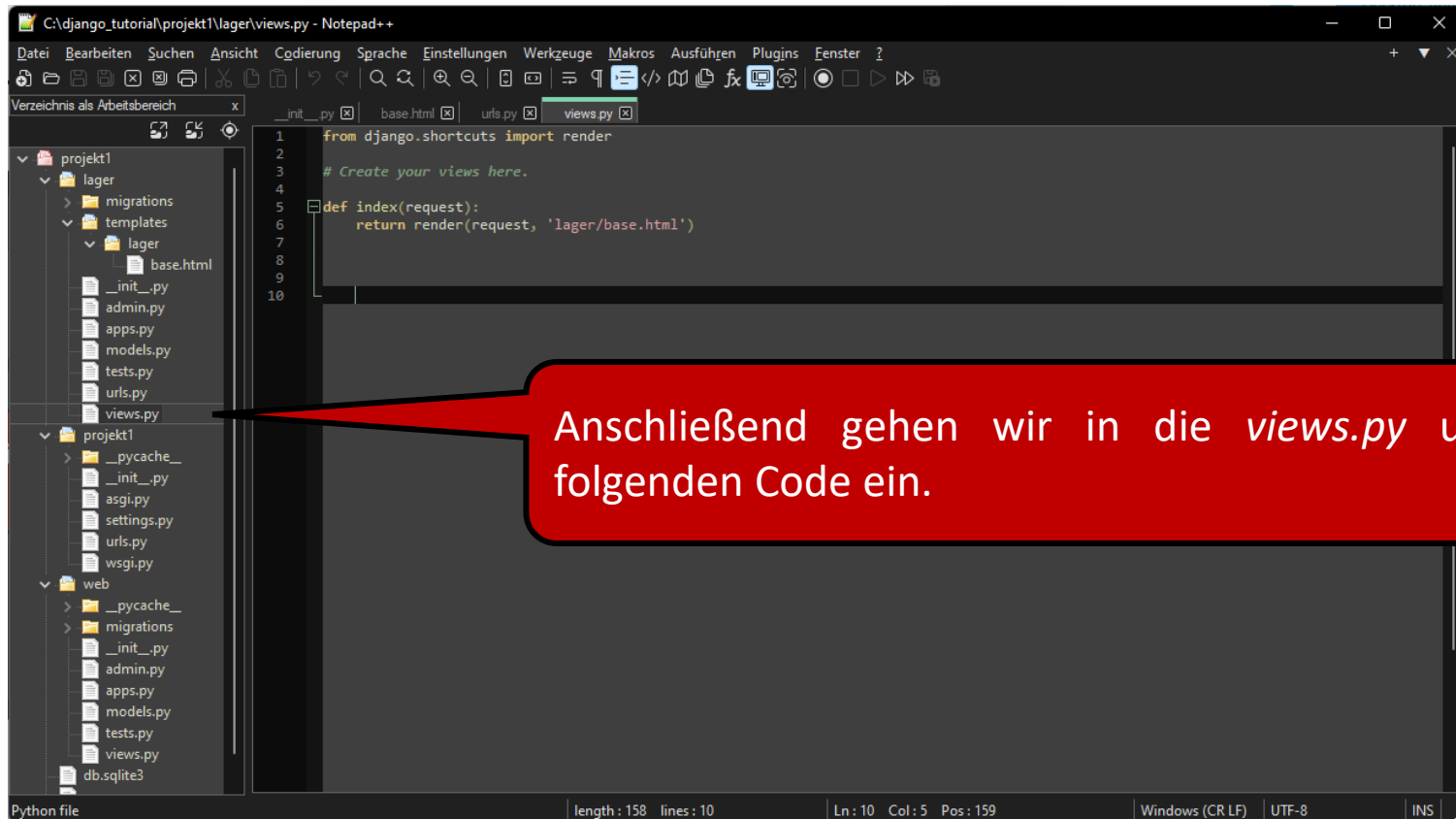


```
1 from django.urls import path
2 from . import views
3
4 app_name="lager"
5 urlpatterns = [
6     path('', ),
7 ]
8
9
10
```

In unserer neuen urls.py schreiben wir erstmal folgendes rein.

Python file | length: 112 | lines: 10 | Ln: 10 | Col: 1 | Pos: 113 | Windows (CR LF) | UTF-8 | INS

Anlegen des Projektes



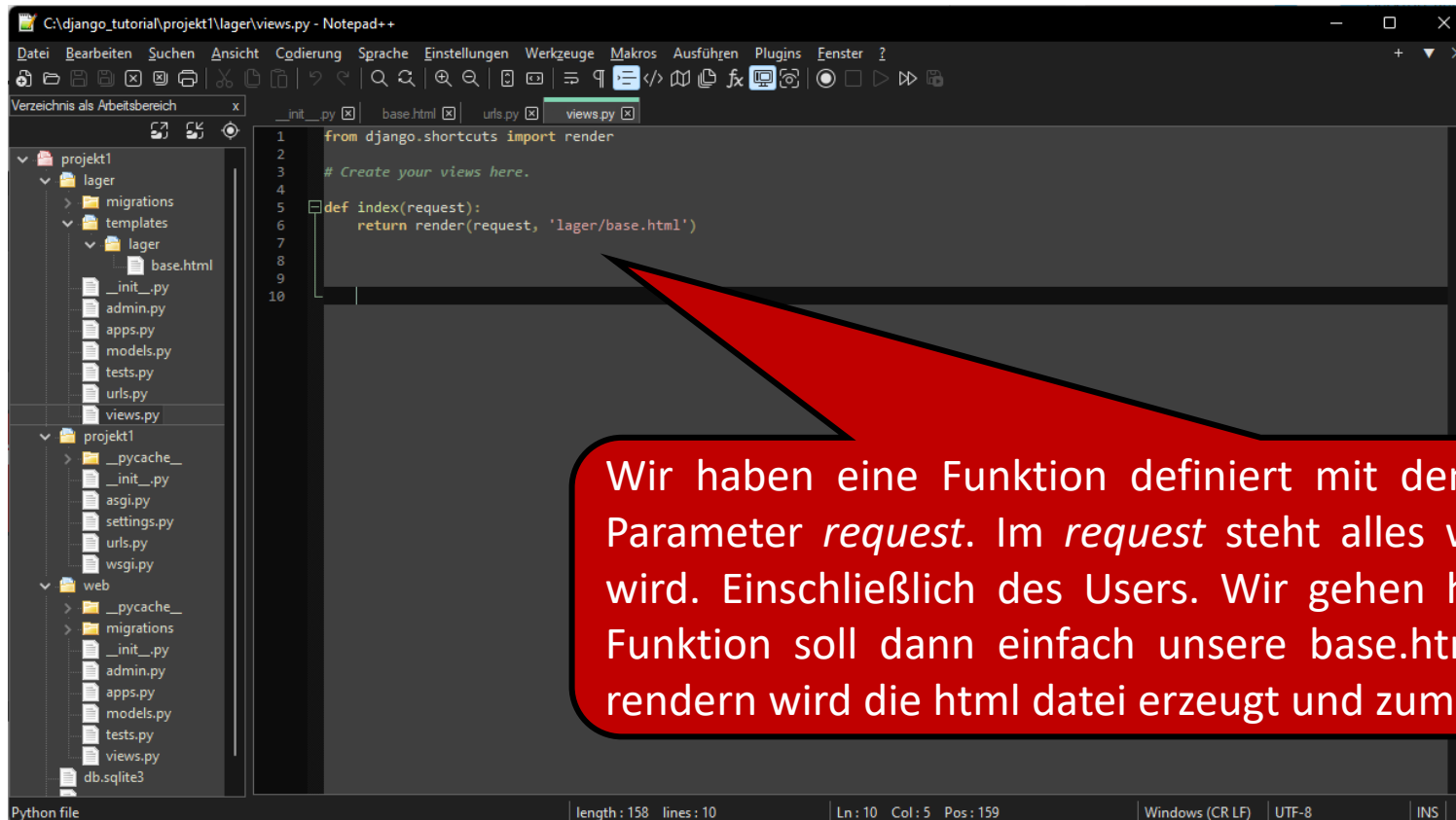
The screenshot shows a Notepad++ window titled 'C:\django_tutorial\projekt1\lager\views.py - Notepad++'. The left sidebar displays a file explorer for the 'projekt1' directory, showing subdirectories like 'lager' and 'projekt1', and files like 'views.py'. The main editor area shows the content of 'views.py' with the following code:

```
1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def index(request):
6     return render(request, 'lager/base.html')
7
8
9
10
```

A red callout box with a black border points to the 'views.py' file in the file explorer and contains the text: 'Anschließend gehen wir in die *views.py* und geben den folgenden Code ein.'

The status bar at the bottom indicates 'Python file', 'length: 158 lines: 10', 'Ln: 10 Col: 5 Pos: 159', 'Windows (CR LF)', 'UTF-8', and 'INS'.

Anlegen des Projektes



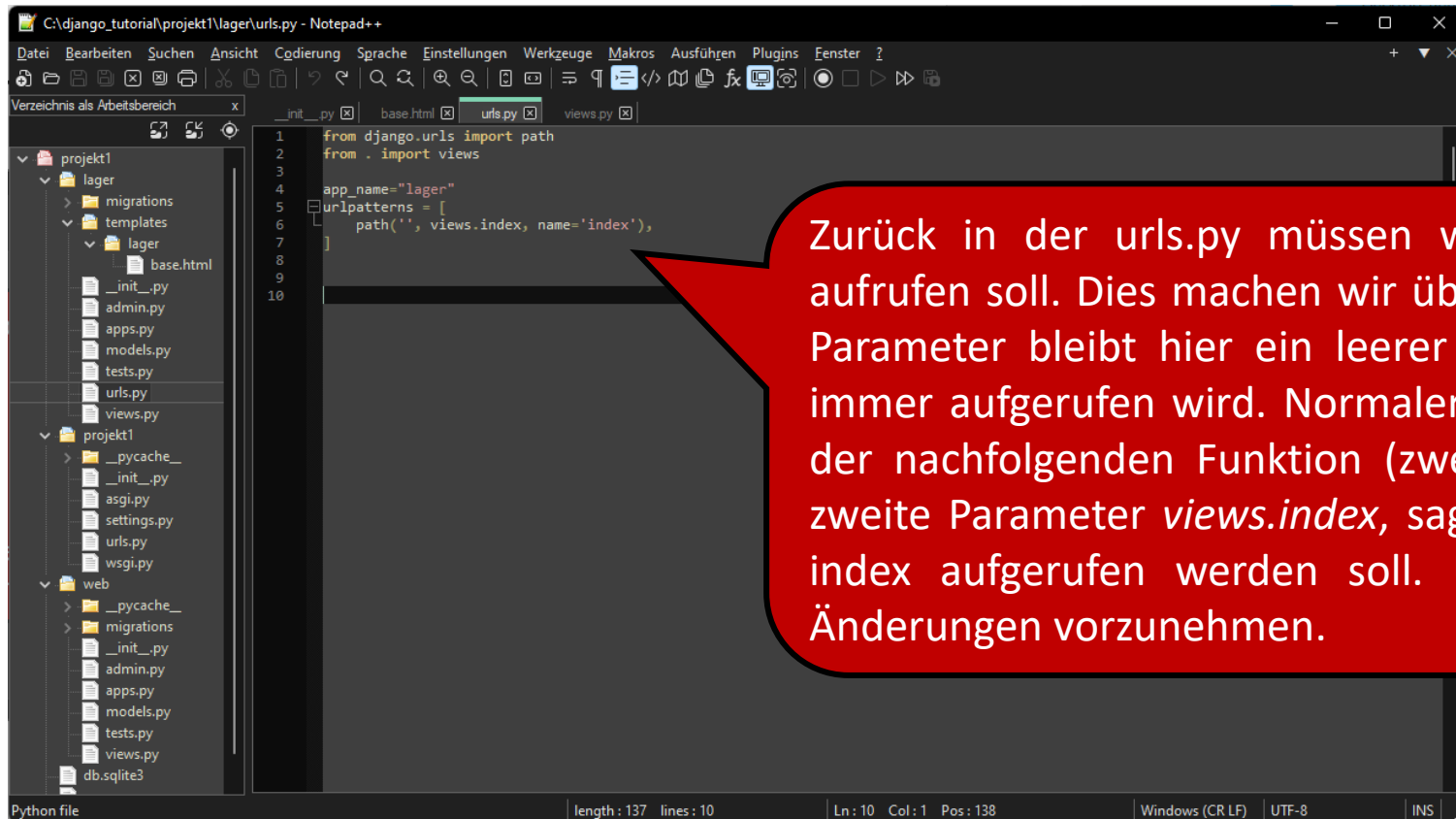
The screenshot shows a Notepad++ window with the file path `C:\django_tutorial\projekt1\lager\views.py`. The left sidebar displays a project tree with folders `projekt1`, `lager`, and `web`. The `lager` folder contains `migrations`, `templates`, and `lager` subfolders, along with files `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py`, and `views.py`. The `views.py` file is open in the editor, showing the following code:

```
1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def index(request):
6     return render(request, 'lager/base.html')
7
8
9
10
```

A red callout bubble points to the `index` function definition.

Wir haben eine Funktion definiert mit dem Namen `index`. Sie bekommt einen Parameter *request*. Im *request* steht alles was von der Website zurück gegeben wird. Einschließlich des Users. Wir gehen hier später noch tiefer drauf ein. Die Funktion soll dann einfach unsere `base.html` zurück geben (rendern). Mit dem rendern wird die html datei erzeugt und zum Browsers des Anwenders gesendet.

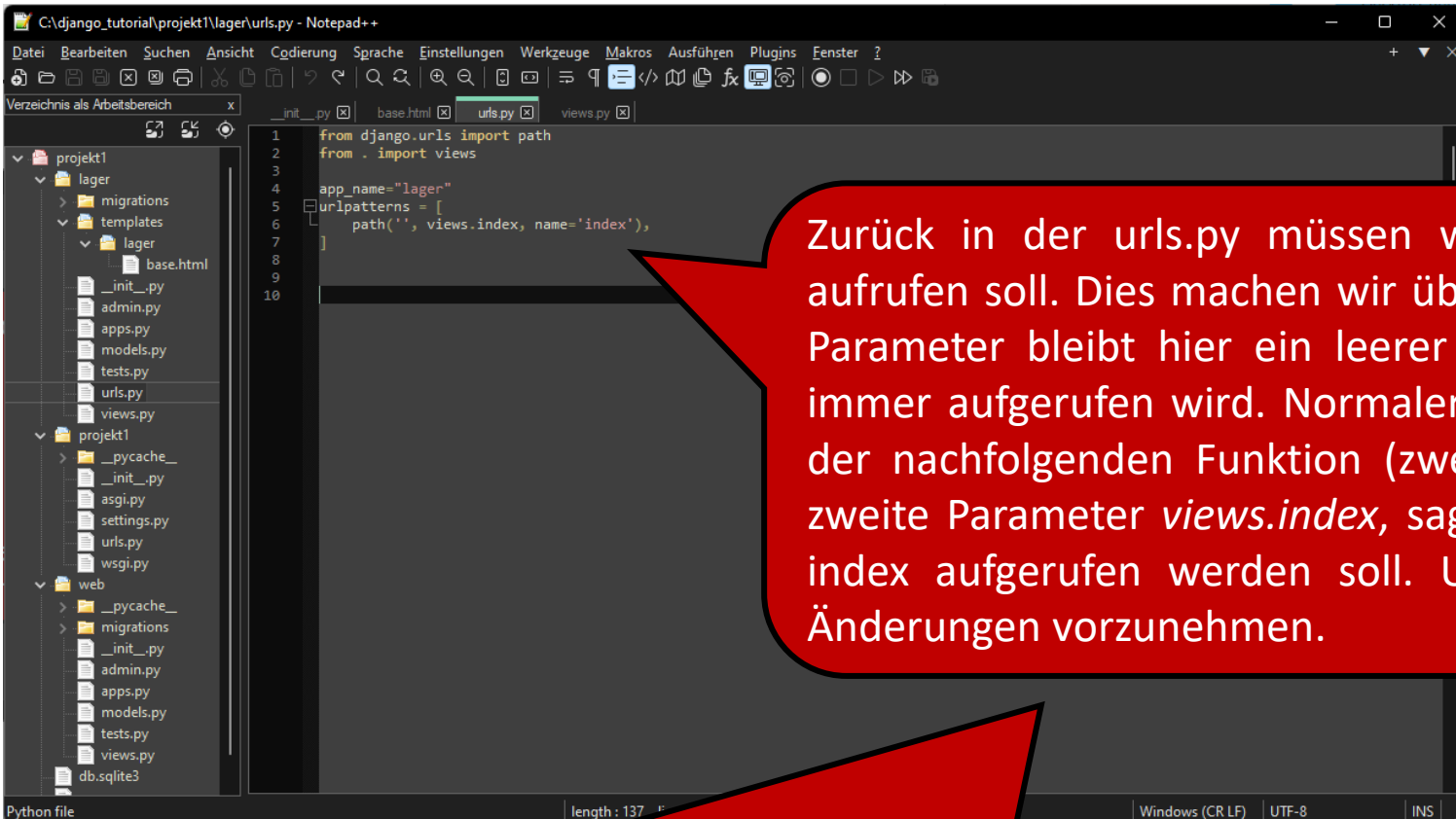
Anlegen des Projektes



```
1 from django.urls import path
2 from . import views
3
4 app_name="lager"
5 urlpatterns = [
6     path('', views.index, name='index'),
7 ]
8
9
10
```

Zurück in der urls.py müssen wir jetzt angeben welche Funktion er wann aufrufen soll. Dies machen wir über die Parameter der *path* Funktion. Der erste Parameter bleibt hier ein leerer String und sagt, soviel wie, das dieser path immer aufgerufen wird. Normalerweise steht hier eine bestimmte URL, die mit der nachfolgenden Funktion (zweiter Parameter) verbunden werden soll. Der zweite Parameter *views.index*, sagt aus das die in der *views* Datei die Funktion *index* aufgerufen werden soll. Und der Parametername ermöglicht django Änderungen vorzunehmen.

Anlegen des Projektes

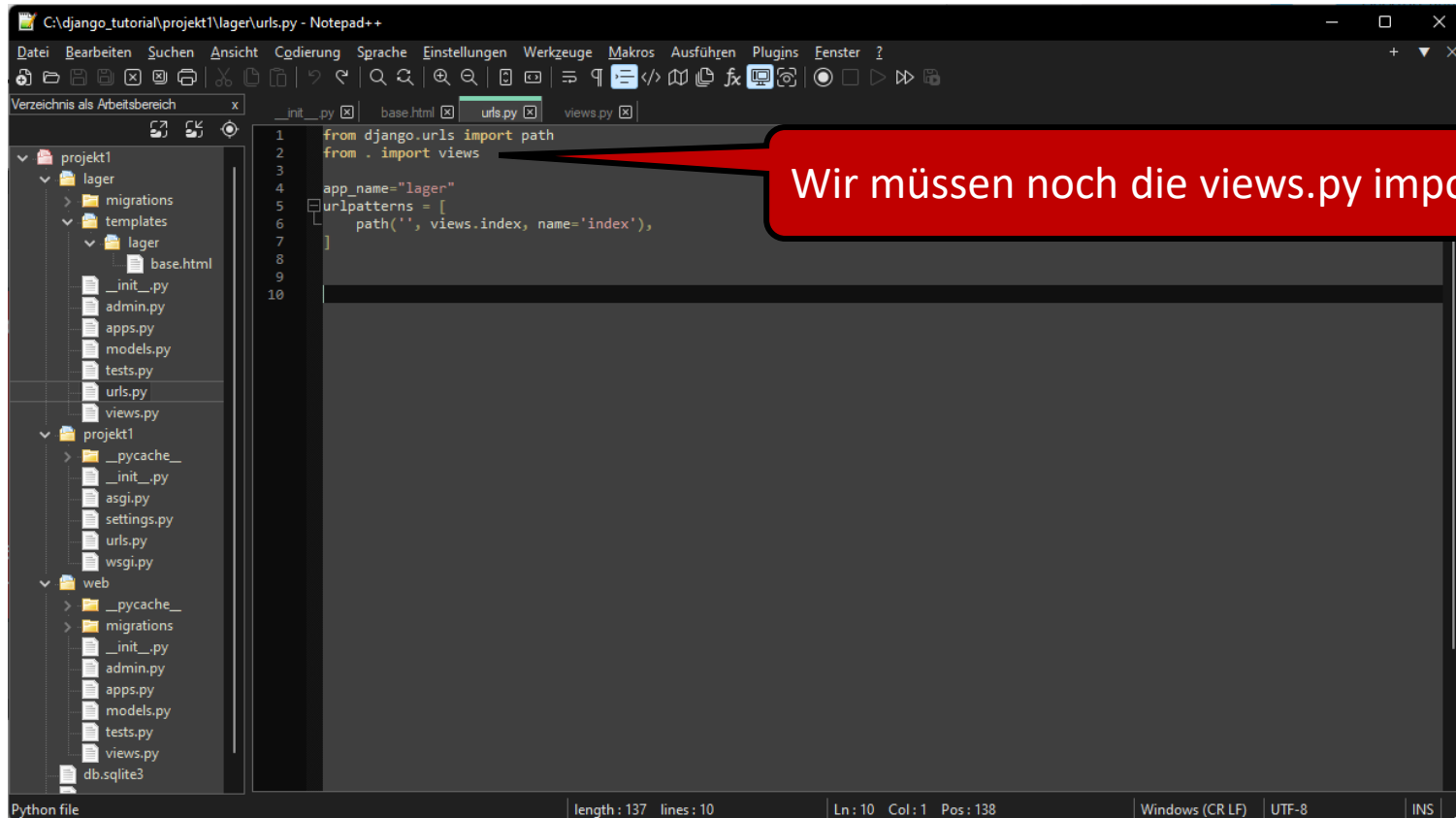


```
1 from django.urls import path
2 from . import views
3
4 app_name="lager"
5 urlpatterns = [
6     path('', views.index, name='index'),
7 ]
8
9
10
```

Zurück in der `urls.py` müssen wir jetzt angeben welche Funktion er wann aufrufen soll. Dies machen wir über die Parameter der `path` Funktion. Der erste Parameter bleibt hier ein leerer String und sagt, soviel wie, das dieser `path` immer aufgerufen wird. Normalerweise steht hier eine bestimmte URL, die mit der nachfolgenden Funktion (zweiter Parameter) verbunden werden soll. Der zweite Parameter `views.index`, sagt aus das die in der `views` Datei die Funktion `index` aufgerufen werden soll. Und der Parameter `name` ermöglicht django Änderungen vorzunehmen.

Also wenn der User in der Adresszeile des Browser eine URL (1. Parameter) eingibt, wird die Funktion die als 2. Parameter steht aufgerufen.

Anlegen des Projektes

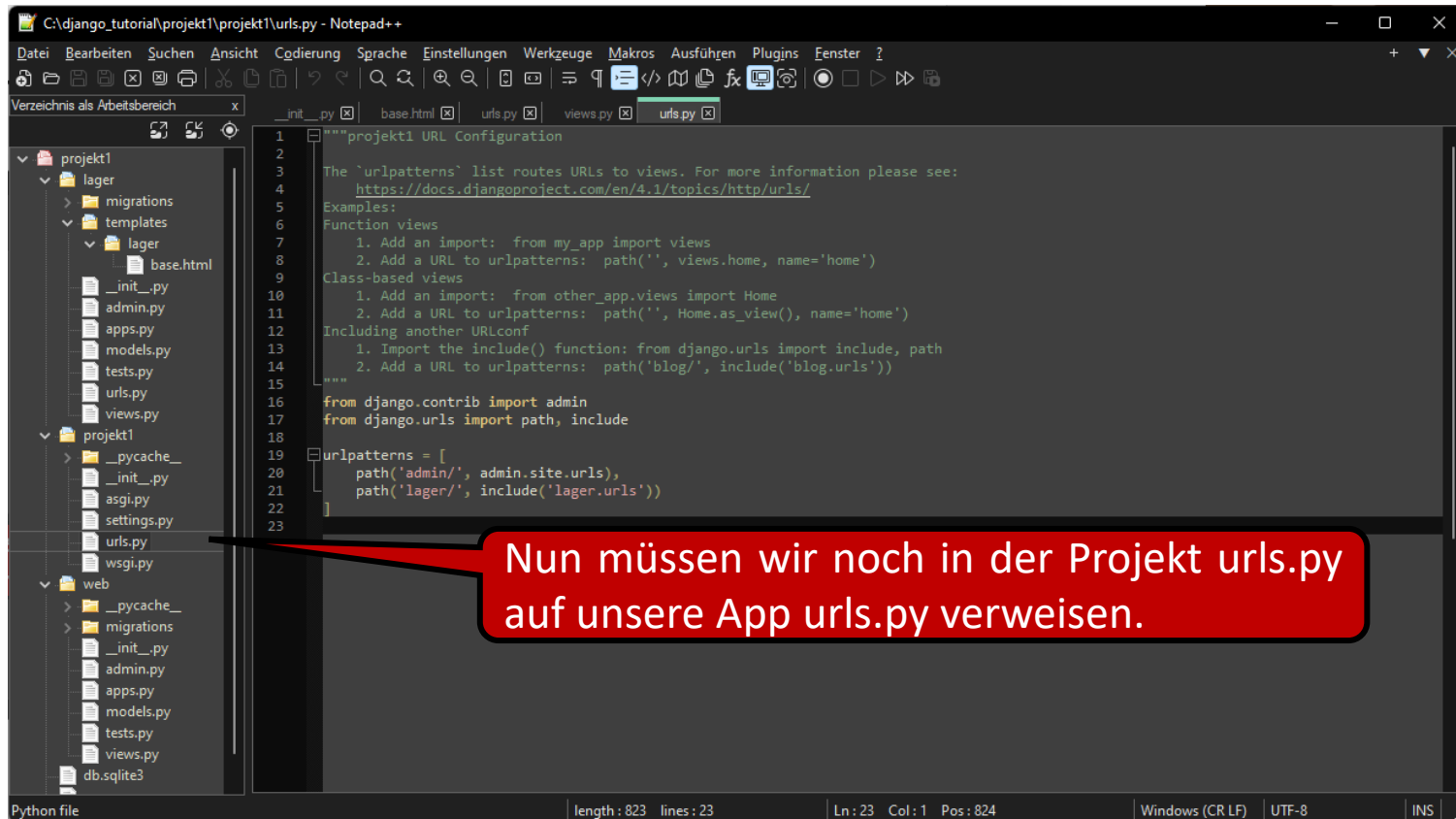


```
1 from django.urls import path
2 from . import views
3
4 app_name="lager"
5 urlpatterns = [
6     path('', views.index, name='index'),
7 ]
8
9
10
```

Wir müssen noch die views.py importieren.

Python file | length: 137 lines: 10 | Ln: 10 Col: 1 Pos: 138 | Windows (CR LF) | UTF-8 | INS

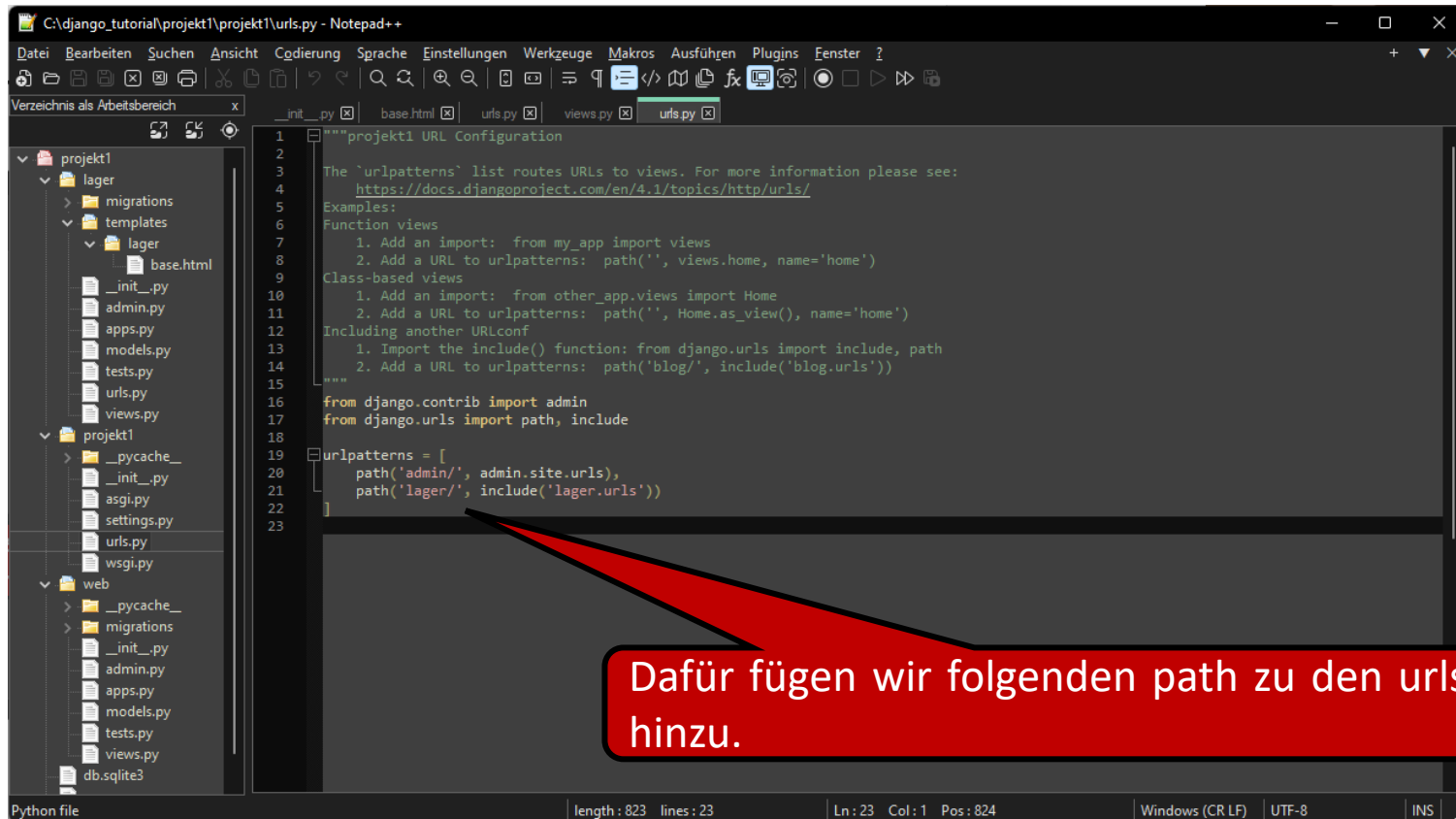
Anlegen des Projektes



```
1 """projekt1 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import:  from my_app import views
8     2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10    1. Add an import:  from other_app.views import Home
11    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('lager/', include('lager.urls'))
22 ]
23
```

Nun müssen wir noch in der Projekt urls.py auf unsere App urls.py verweisen.

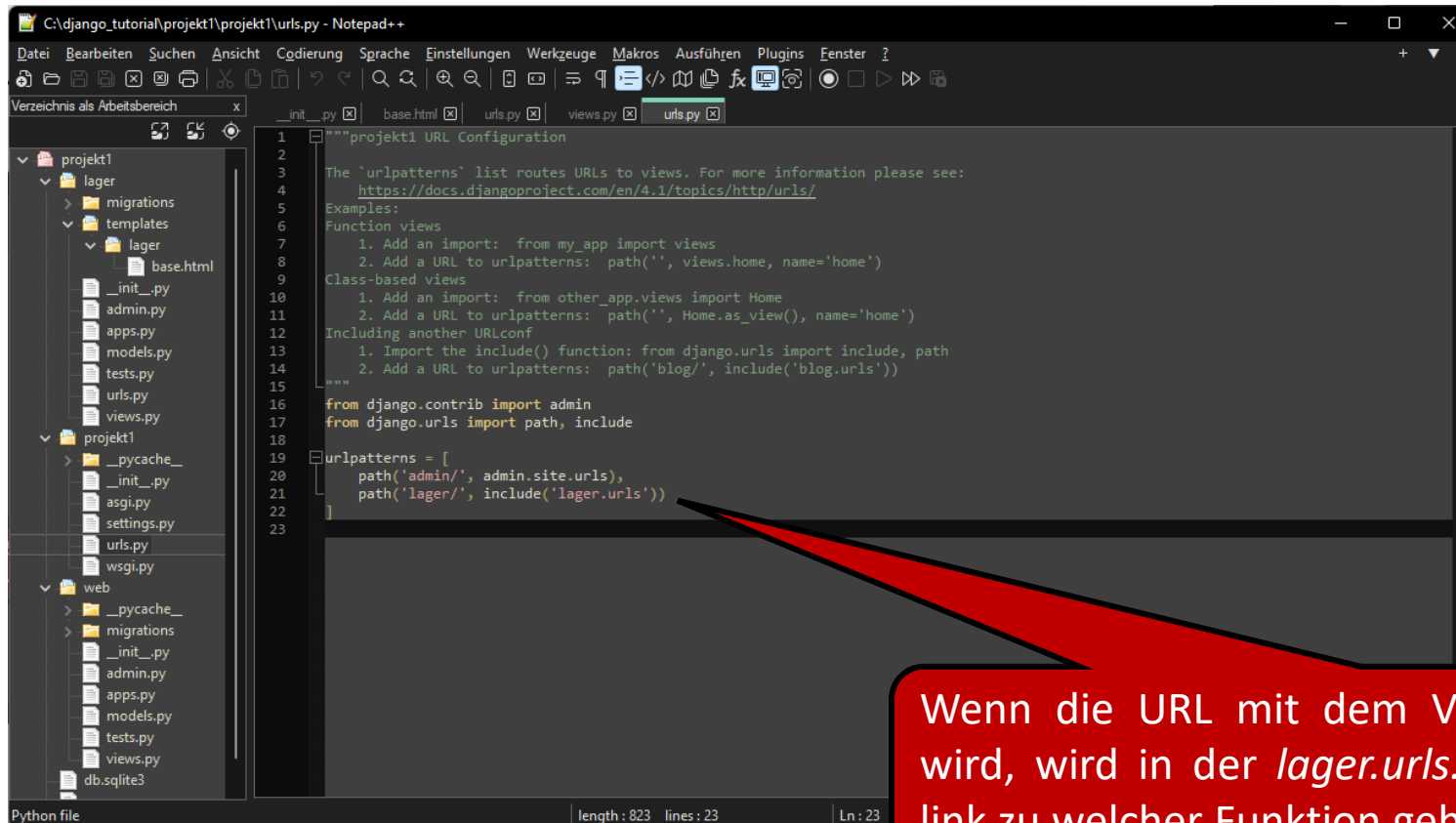
Anlegen des Projektes



```
1 """projekt1 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7 1. Add an import:  from my_app import views
8 2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10 1. Add an import:  from other_app.views import Home
11 2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('lager/', include('lager.urls'))
22 ]
23
```

Dafür fügen wir folgenden path zu den urls hinzu.

Anlegen des Projektes



```
1 """projekt1 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import:  from my_app import views
8     2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10    1. Add an import:  from other_app.views import Home
11    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('lager/', include('lager.urls'))
22 ]
23
```

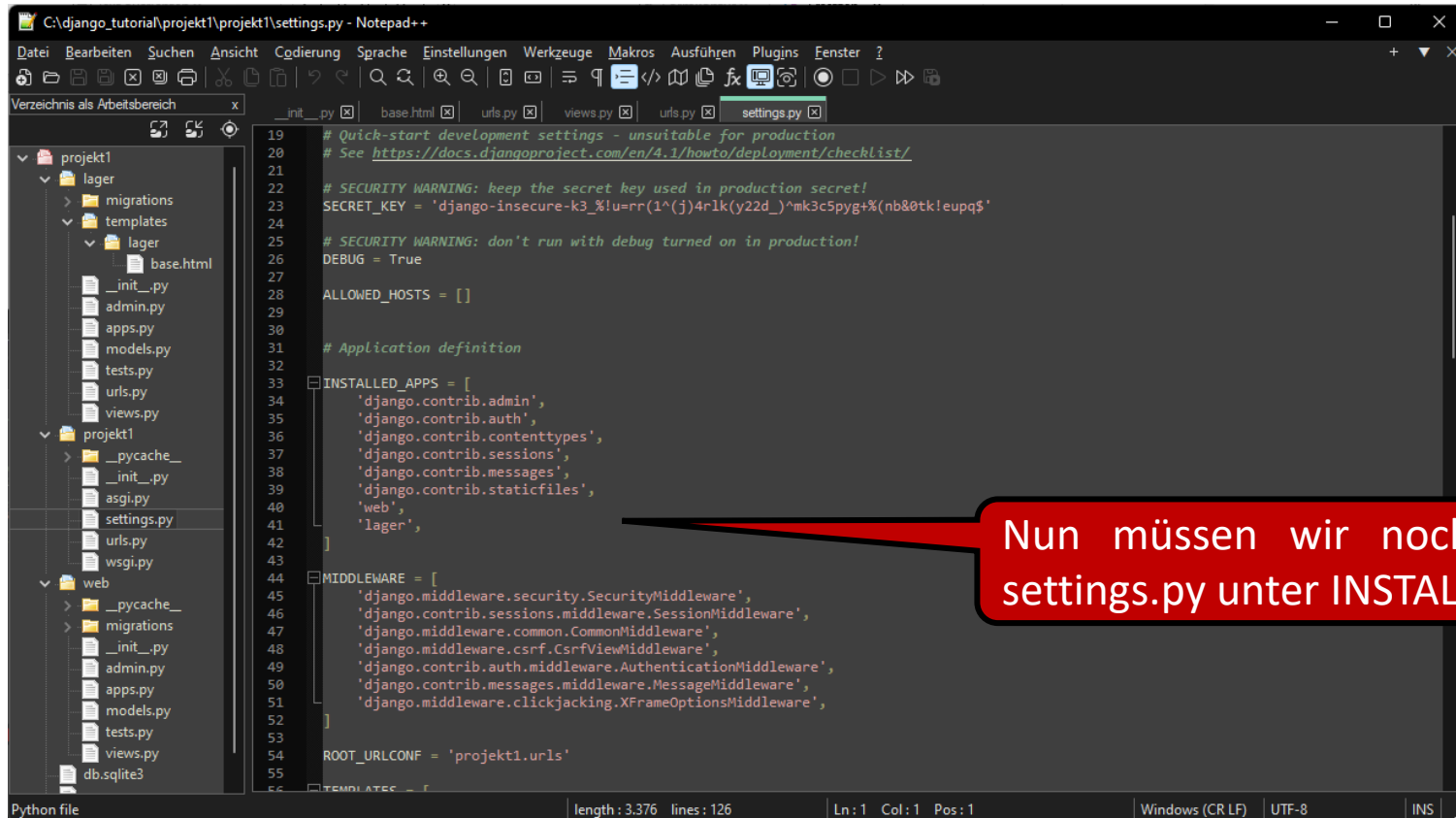
Wenn die URL mit dem Verweis „/lager“ aufgerufen wird, wird in der *lager.urls.py* geprüft welcher direkte link zu welcher Funktion gehört.

Anlegen des Projektes

```
1 """projekt1 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import:  from my_app import views
8     2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10    1. Add an import:  from other_app.views import Home
11    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('lager/', include('lager.urls'))
22 ]
23
```

Außerdem müssen wir noch *include* importieren.

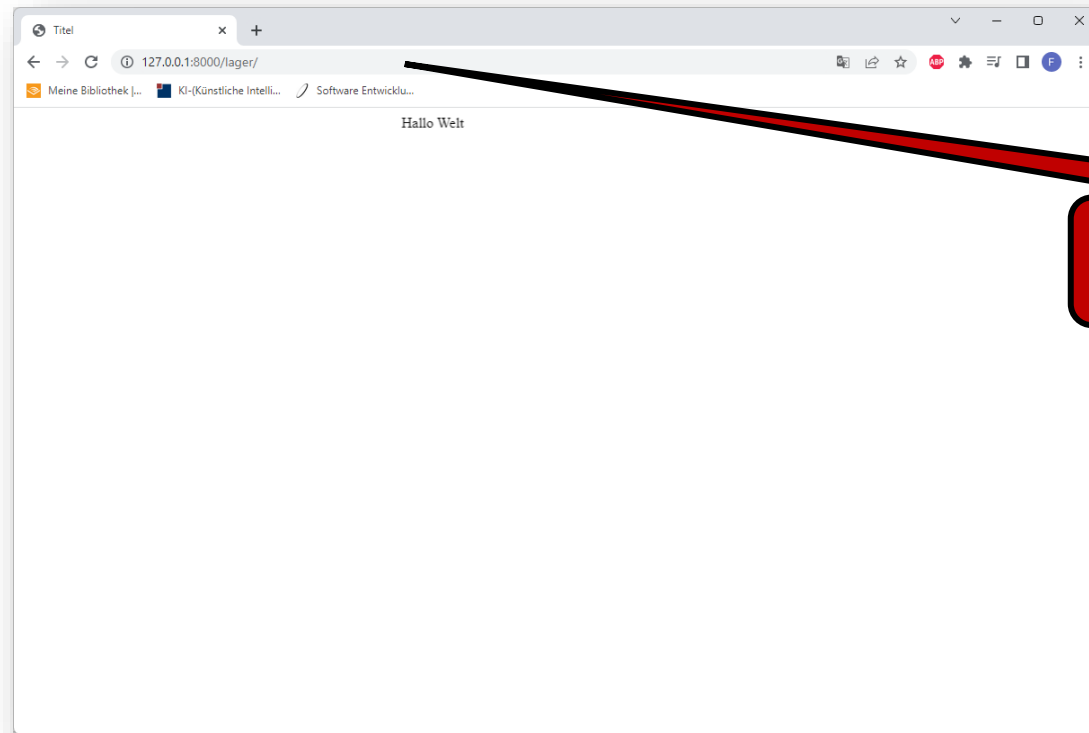
Anlegen des Projektes



```
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-k3!u=rr(1^(j)4rlk(y22d_)^mk3c5pyg+(nb80tk!eupq$'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'web',
41     'lager',
42 ]
43
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52 ]
53
54 ROOT_URLCONF = 'projekt1.urls'
55
56 TEMPLATES = [
```

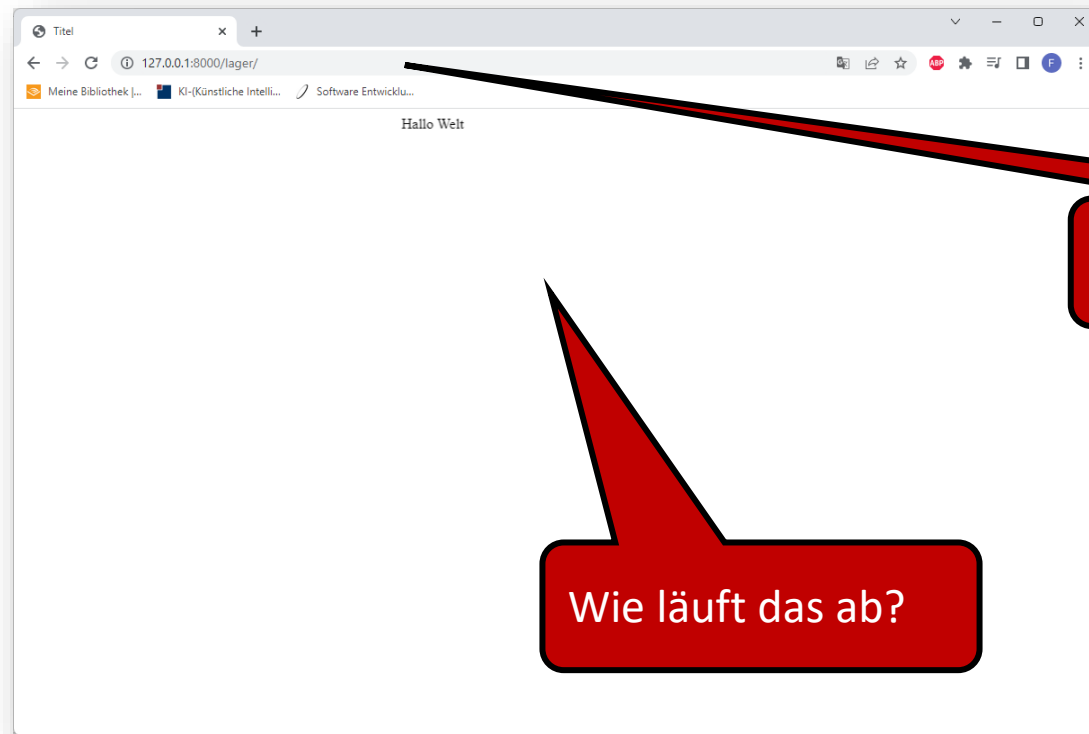
Nun müssen wir noch die beiden Apps, in der settings.py unter INSTALLED_APPS hinzufügen.

Anlegen des Projektes



Dann rufen wir diese Adresse im Browser auf.

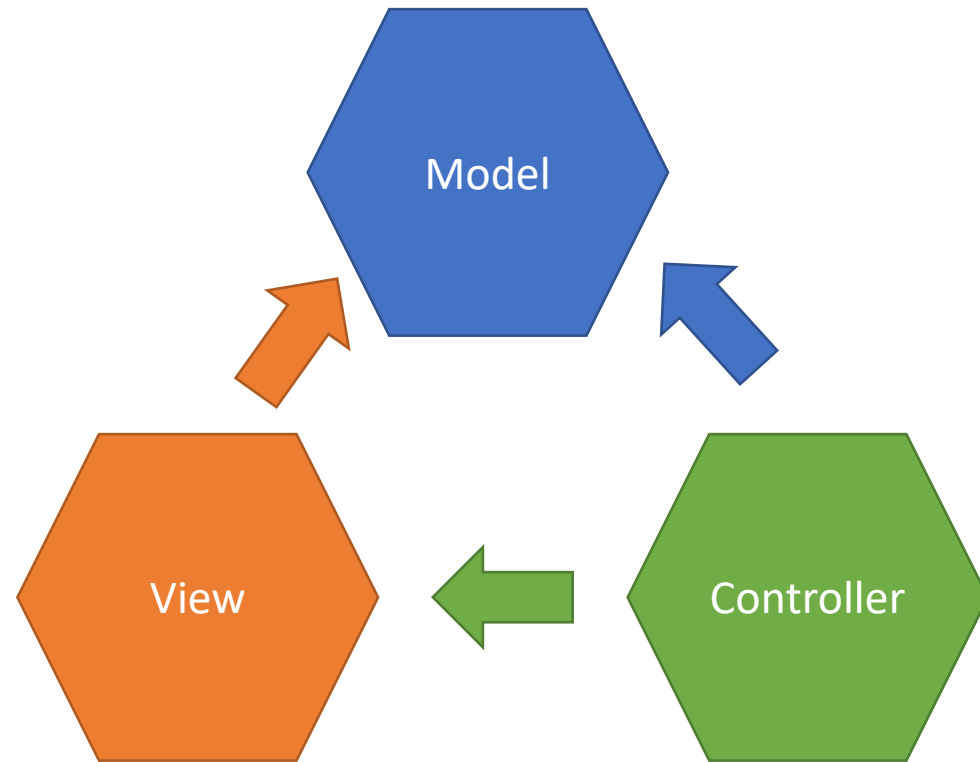
Anlegen des Projektes



Dann rufen wir diese Adresse im Browser auf.

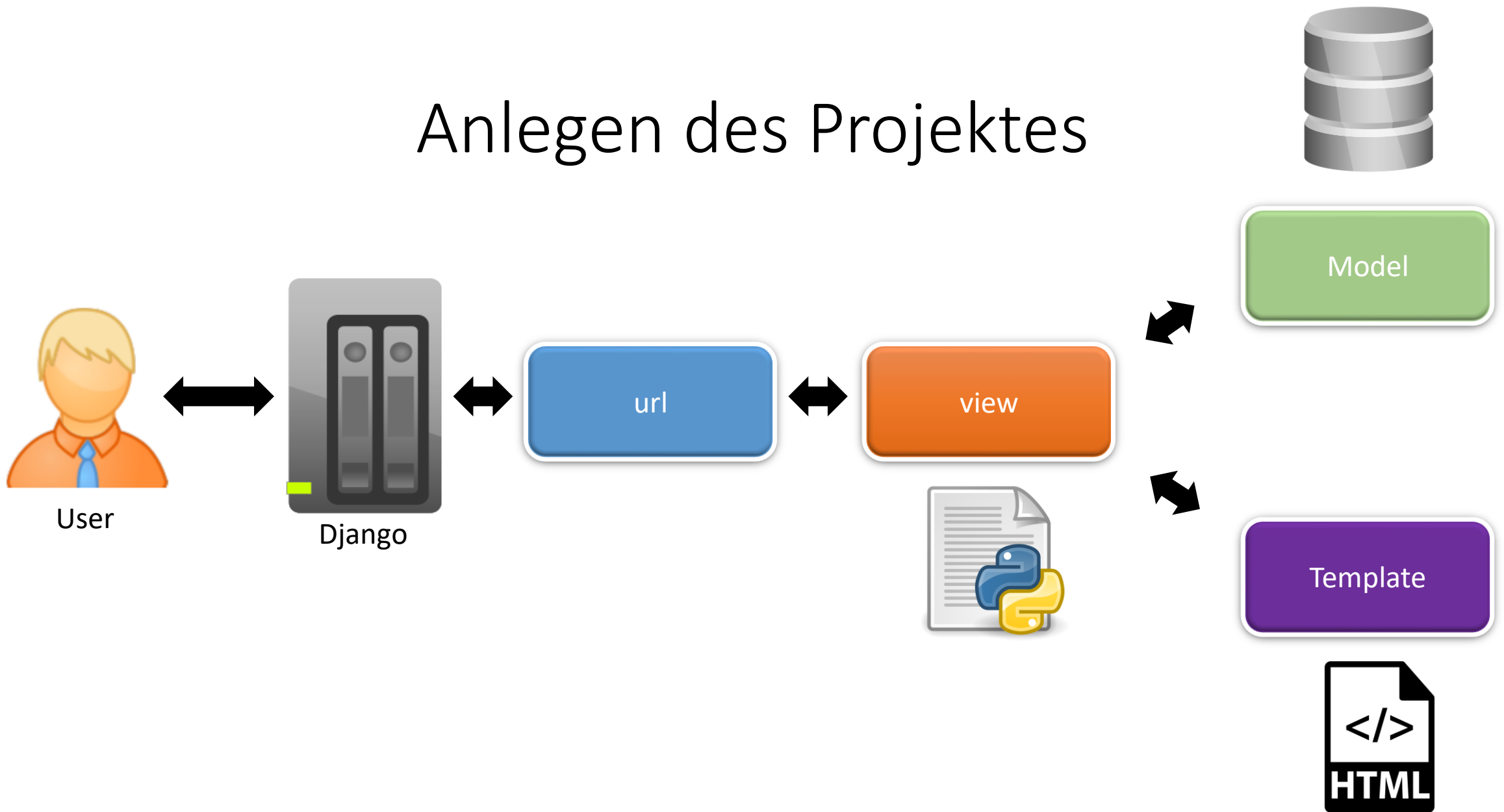
Wie läuft das ab?

Anlegen des Projektes

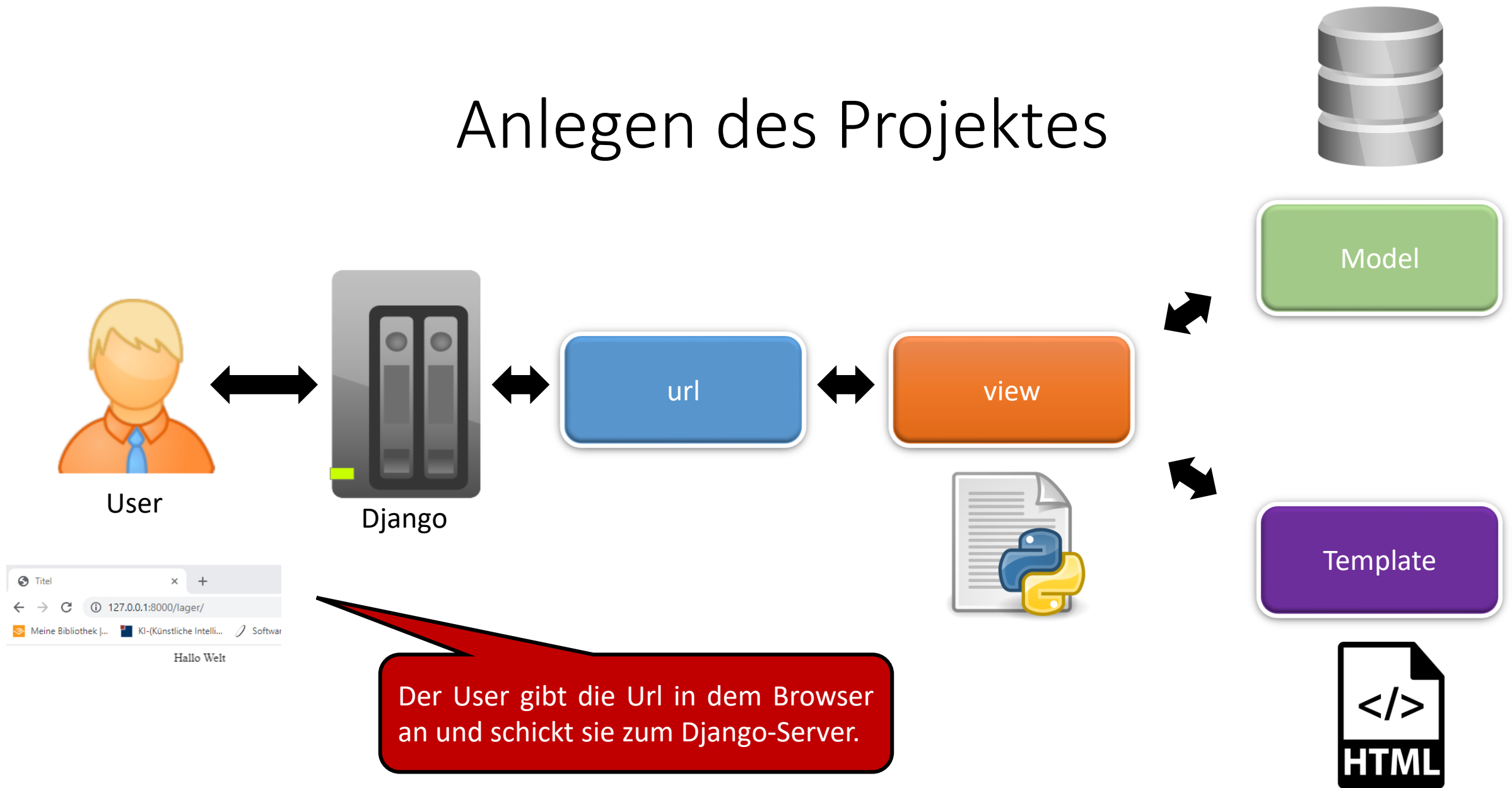


Django benutzt die Software Architecture MVC. Also die Architektur Model-View-Controller. In Django auch MTV genannt, Model-Template-View.

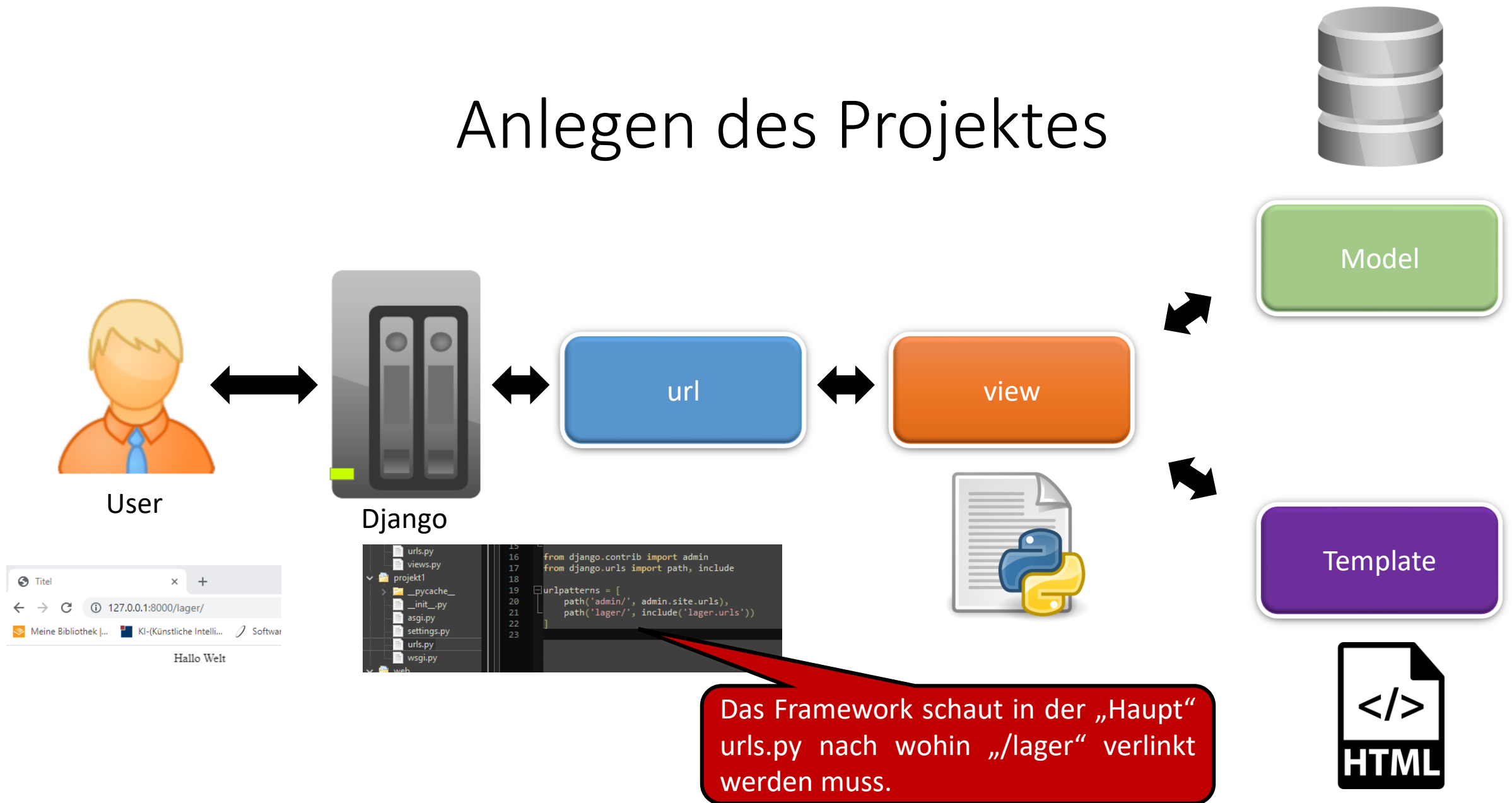
Anlegen des Projektes



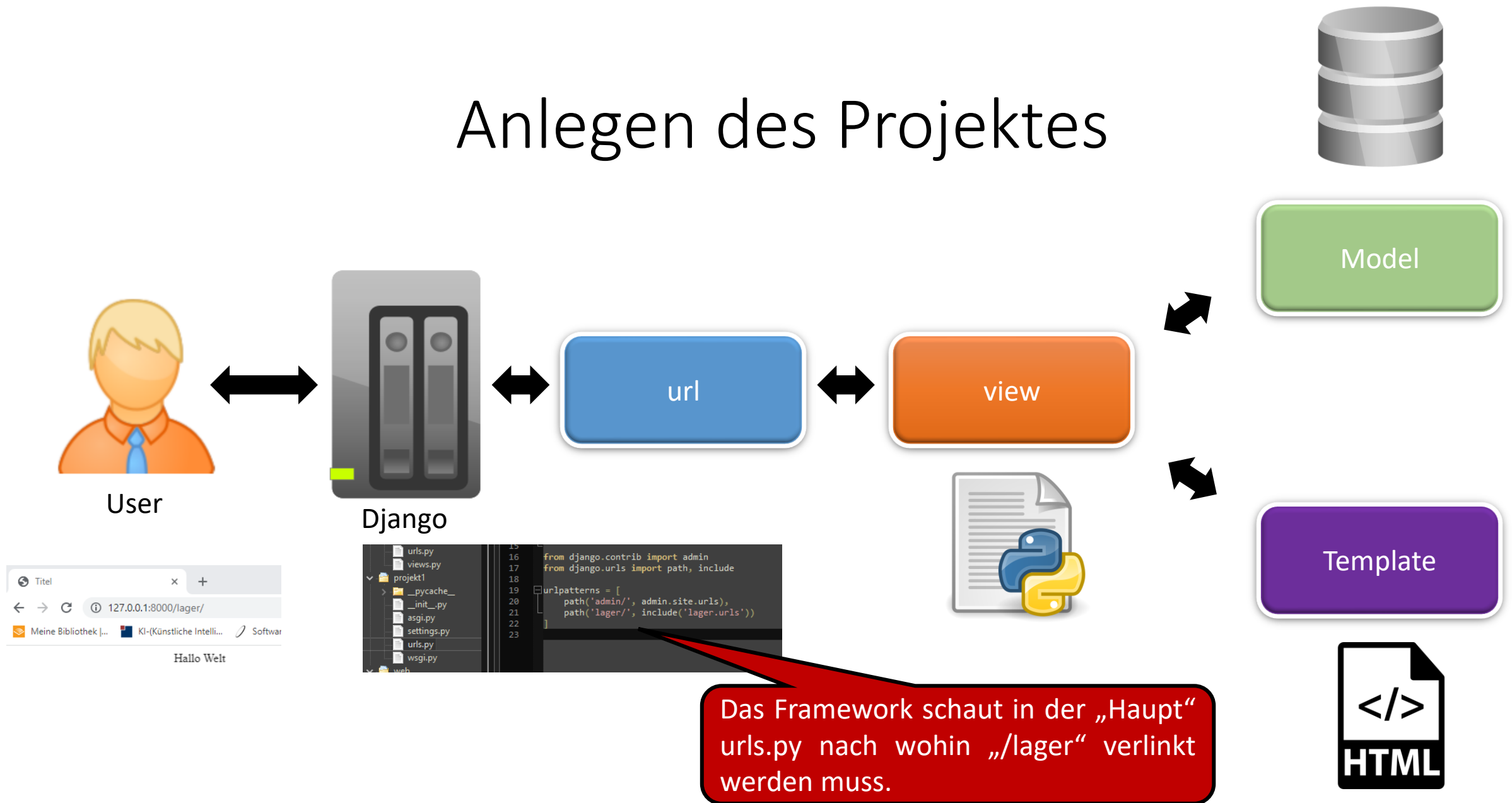
Anlegen des Projektes



Anlegen des Projektes



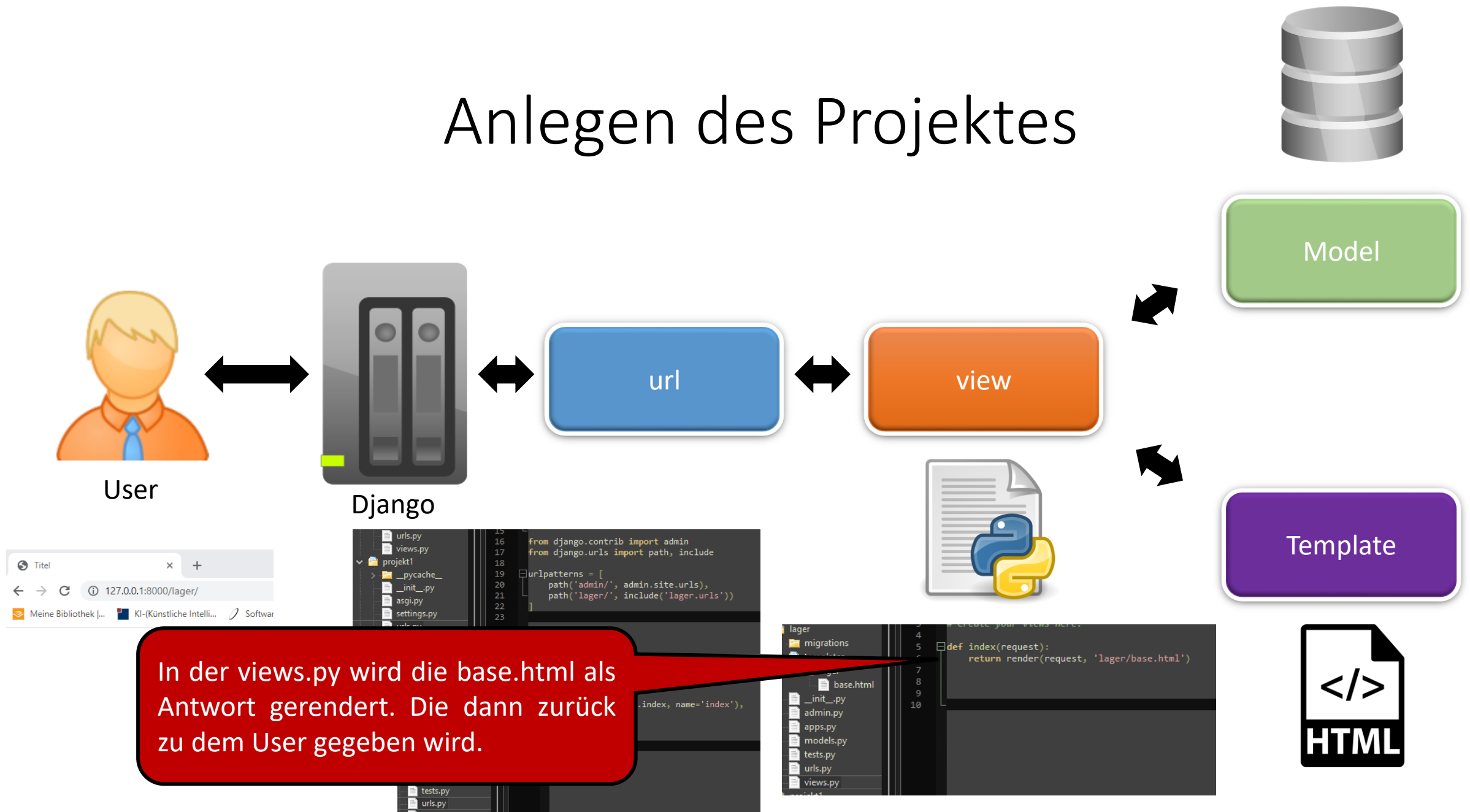
Anlegen des Projektes



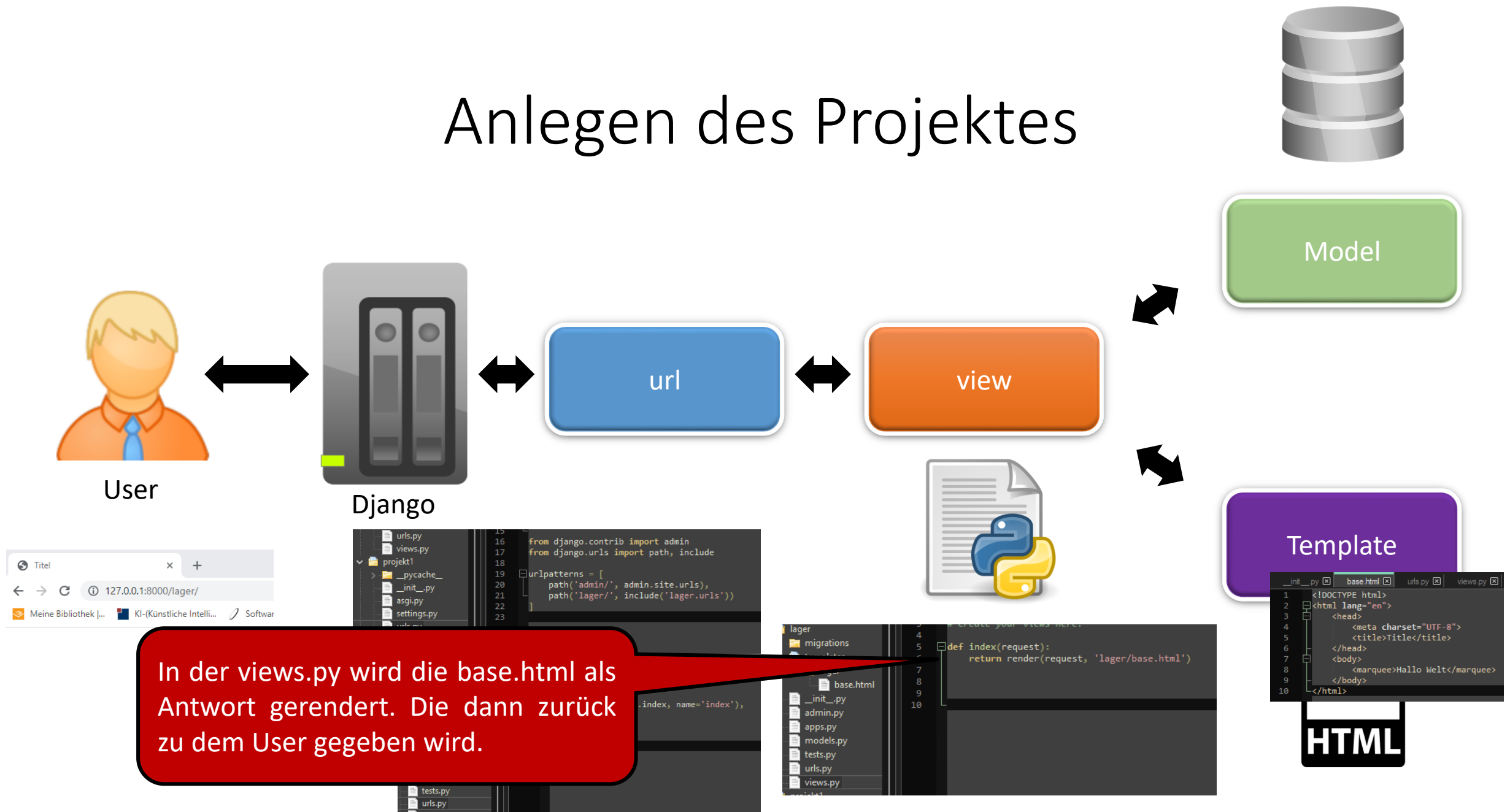
Anlegen des Projektes



Anlegen des Projektes

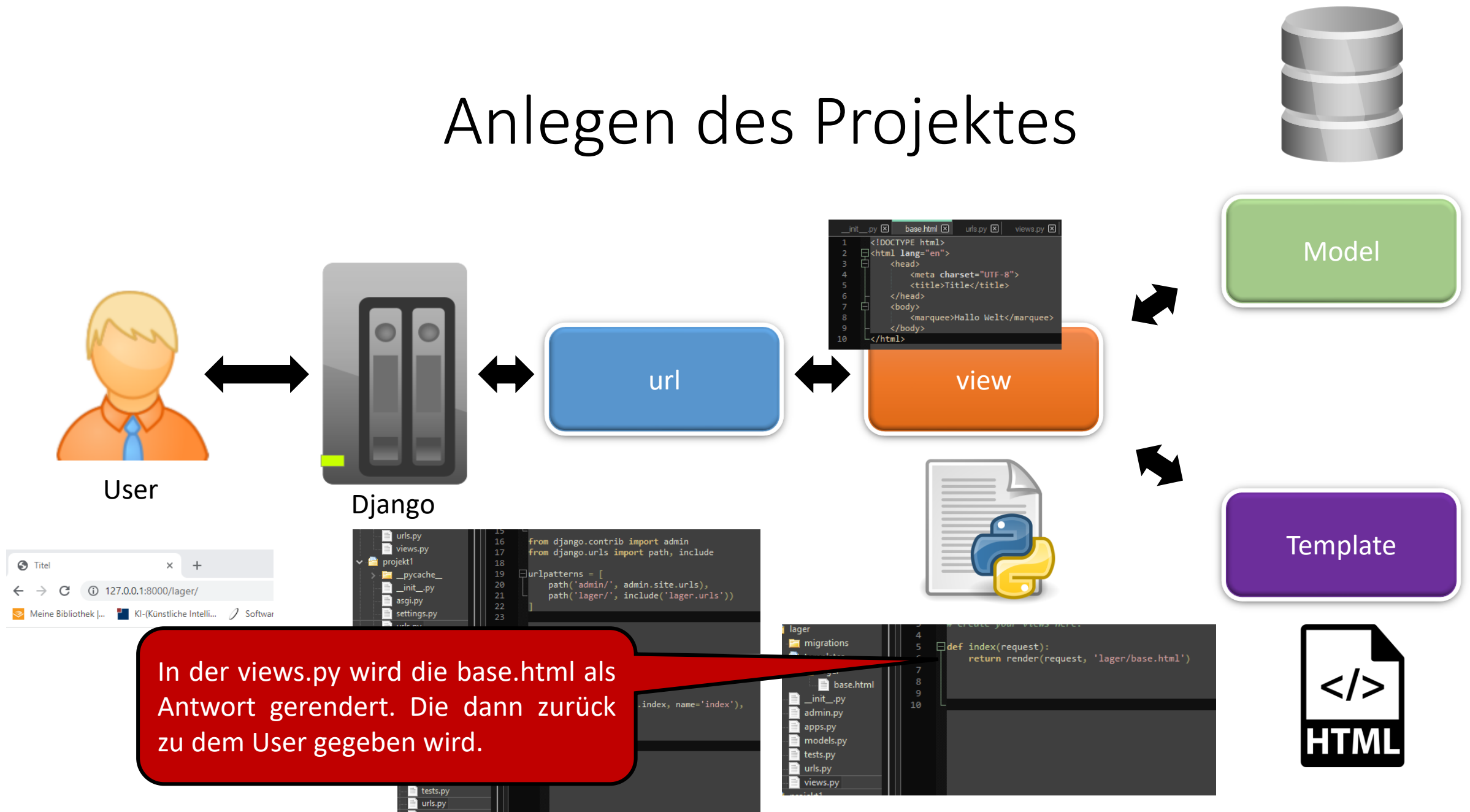


Anlegen des Projektes

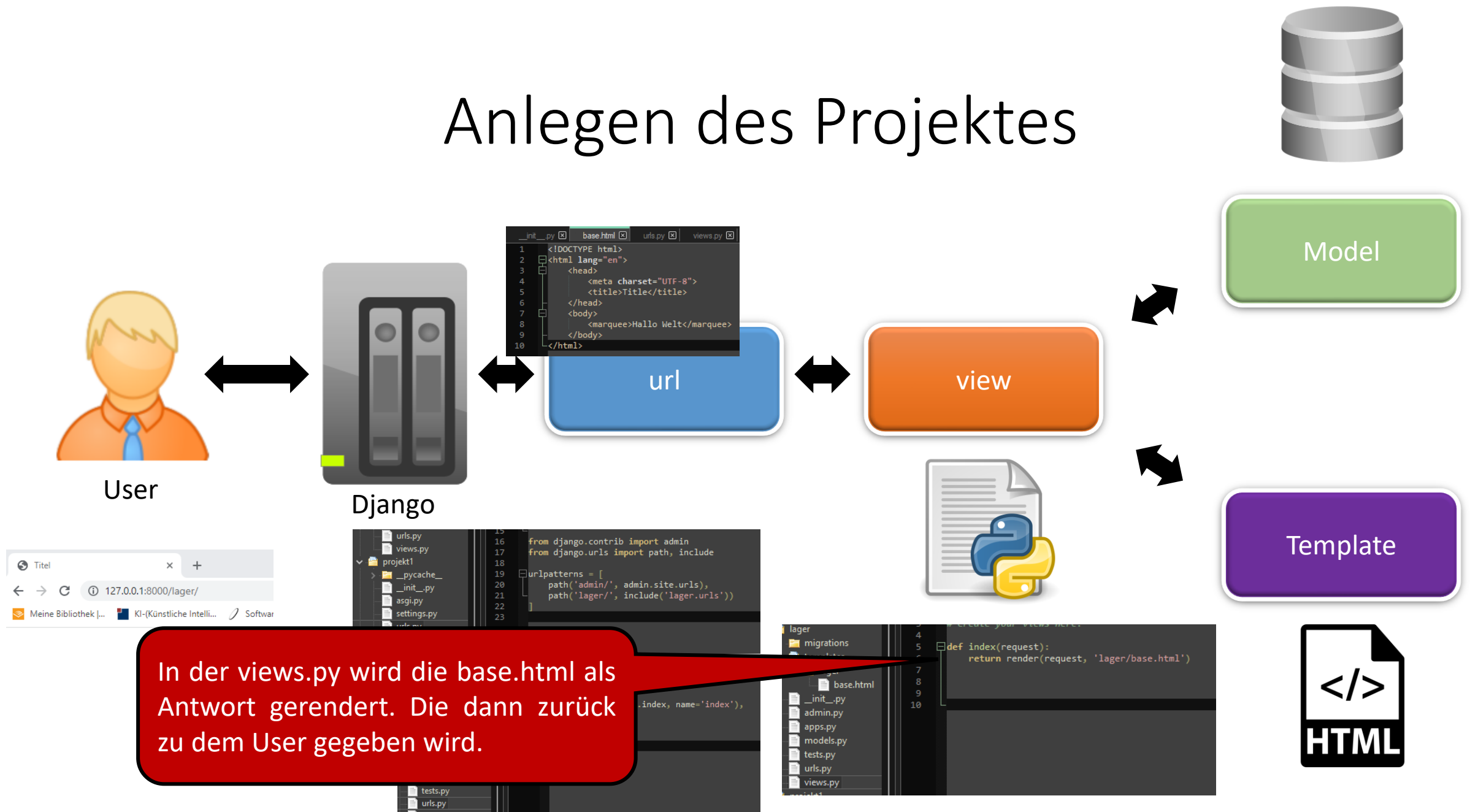


In der views.py wird die base.html als Antwort gerendert. Die dann zurück zu dem User gegeben wird.

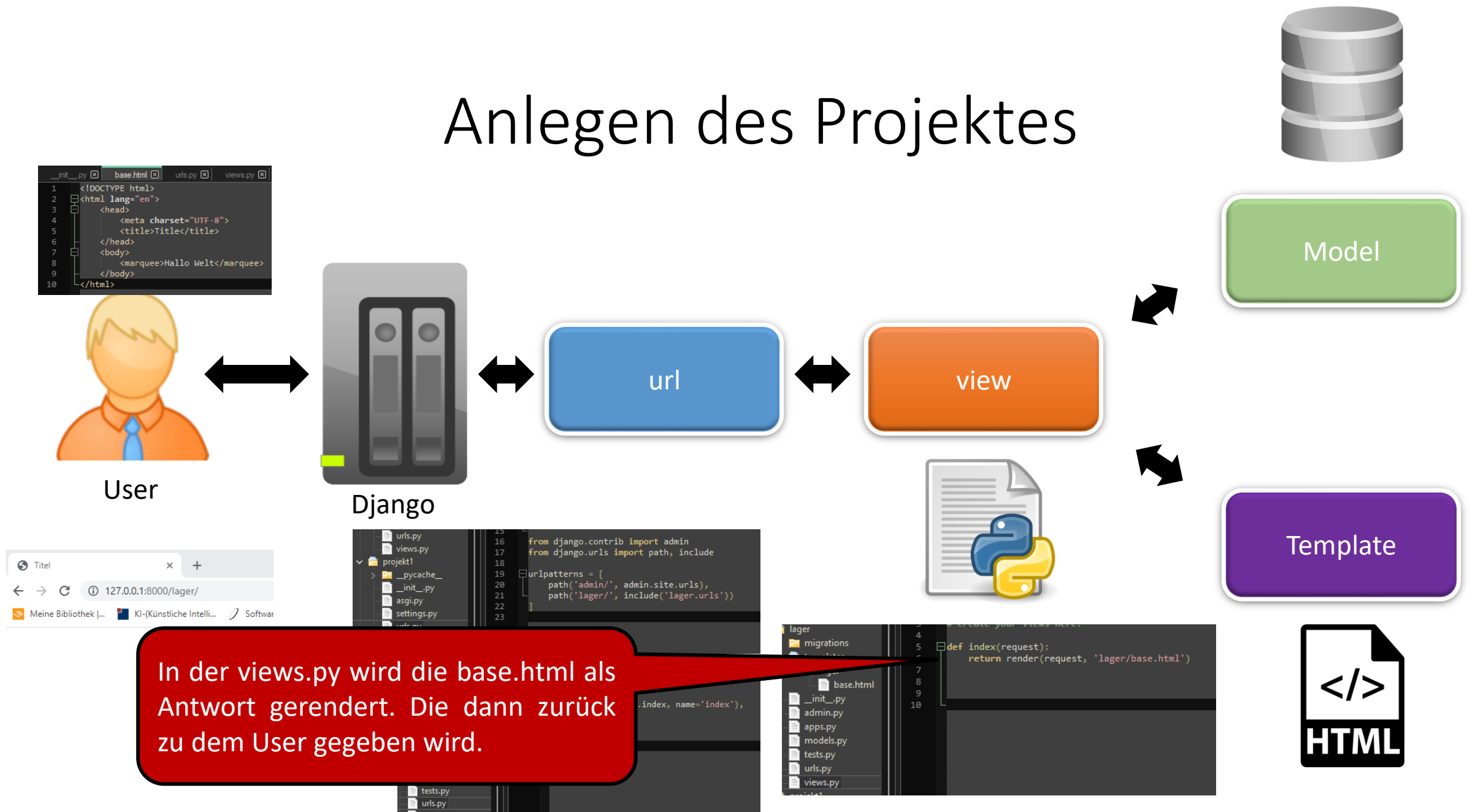
Anlegen des Projektes



Anlegen des Projektes

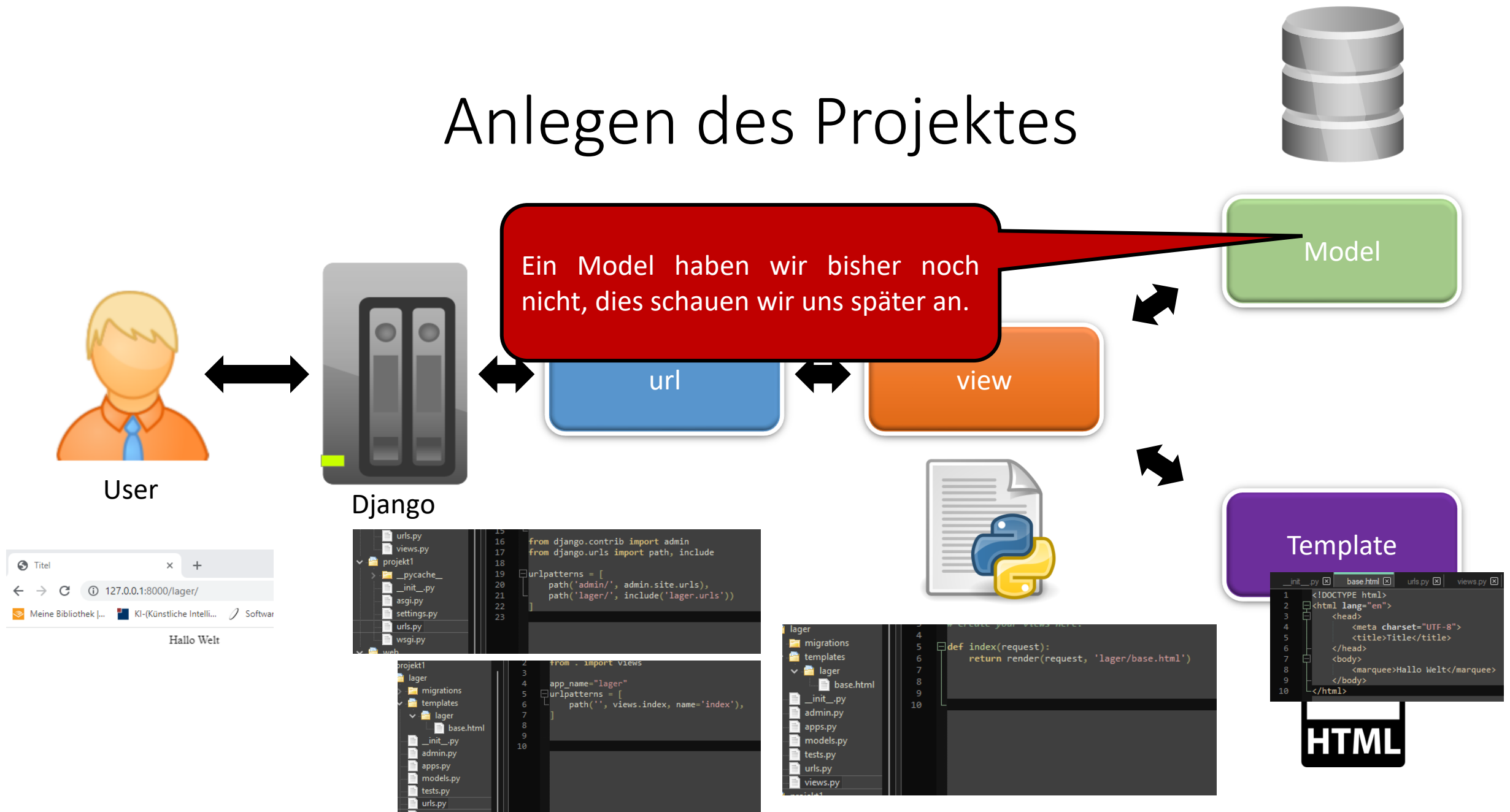


Anlegen des Projektes



In der `views.py` wird die `base.html` als Antwort gerendert. Die dann zurück zu dem User gegeben wird.

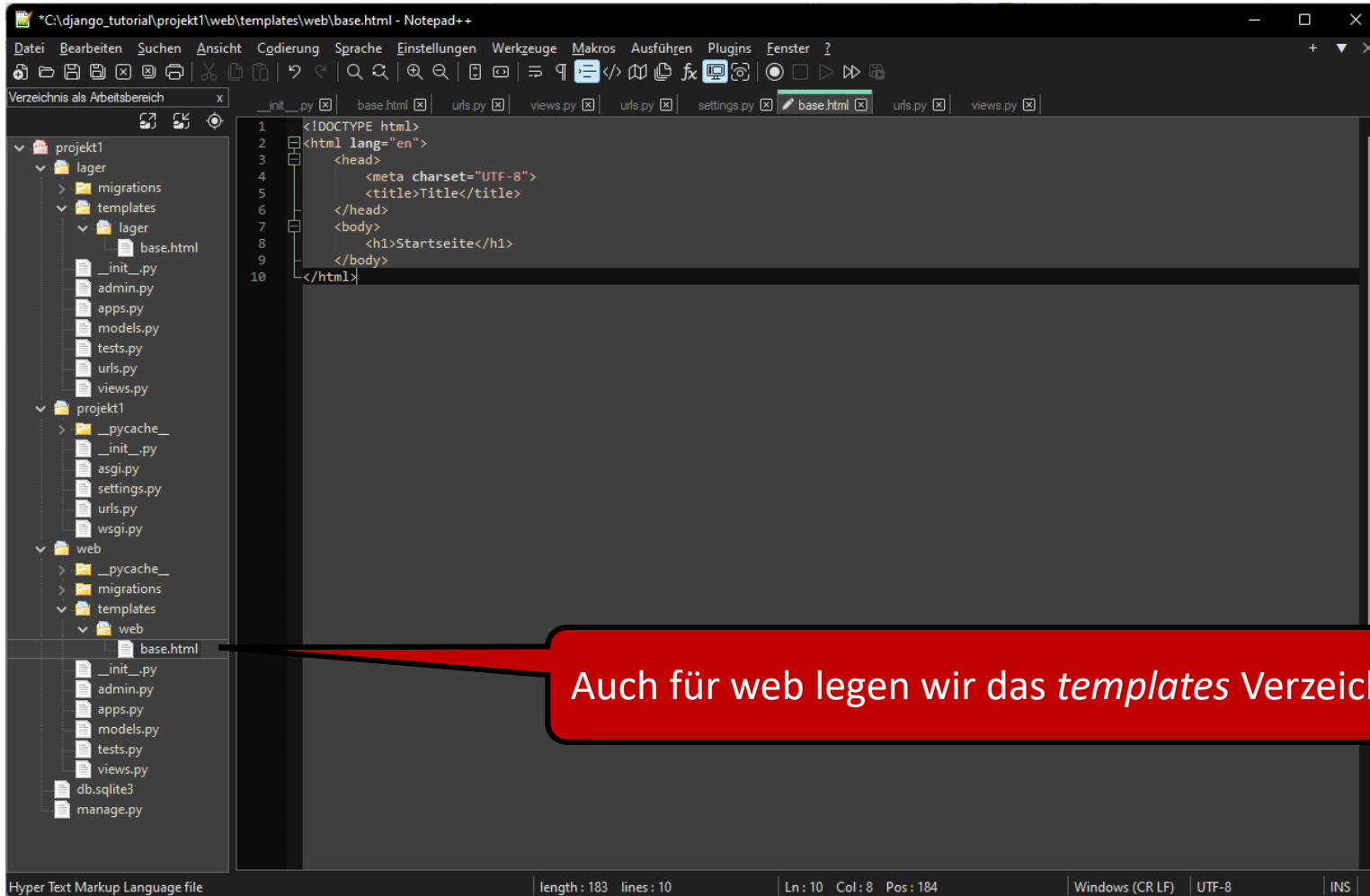
Anlegen des Projektes



Anlegen des Projektes



Anlegen des Projektes

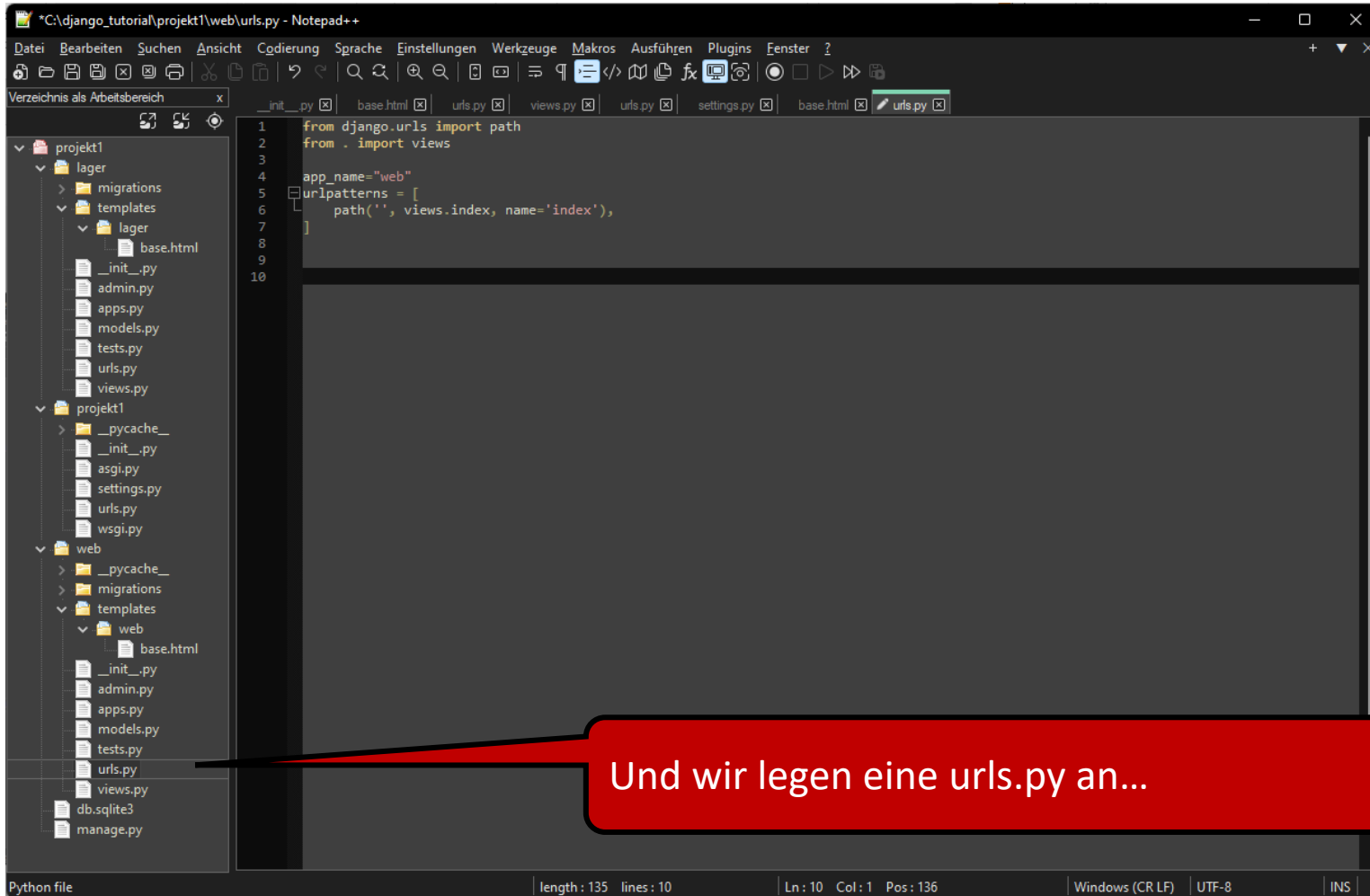


The screenshot shows a Notepad++ window editing a file at `*C:\django_tutorial\projekt1\web\templates\web\base.html`. The file explorer on the left displays the project structure, including folders like `migrations`, `templates`, and `web`. The main editor shows the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6 </head>
7 <body>
8   <h1>Startseite</h1>
9 </body>
10</html>
```

A red callout box points to the `base.html` file in the `web` folder, containing the text: "Auch für web legen wir das *templates* Verzeichnis an."

Anlegen des Projektes



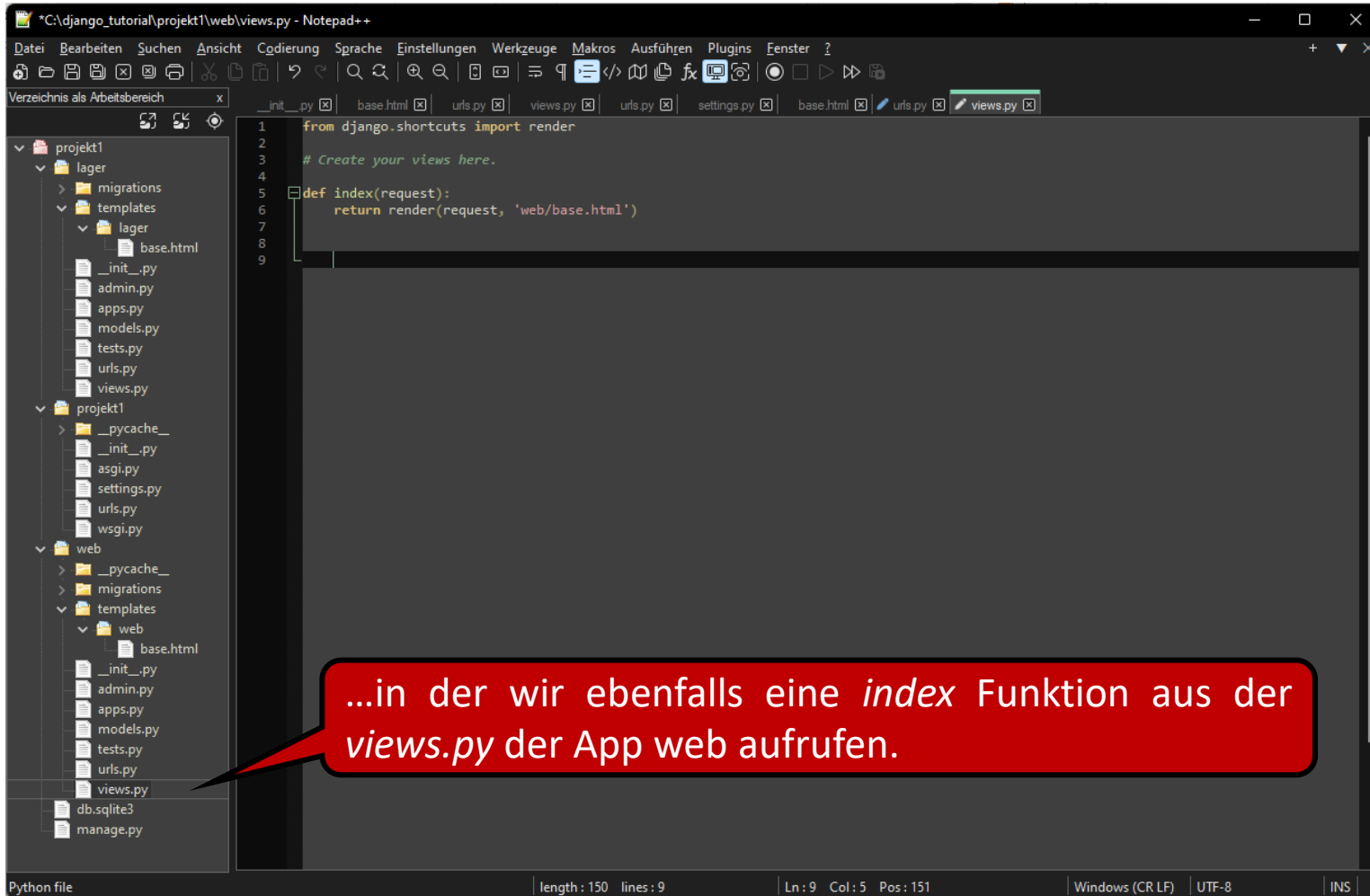
The screenshot shows a Notepad++ window titled "*C:\django_tutorial\projekt1\web\urls.py - Notepad++". The left sidebar displays a file explorer for the "projekt1" directory, showing subdirectories like "lager", "projekt1", and "web". The main editor area shows the content of "urls.py":

```
1 from django.urls import path
2 from . import views
3
4 app_name="web"
5 urlpatterns = [
6     path('', views.index, name='index'),
7 ]
8
9
10
```

A red callout box with a black border points to the "urls.py" file in the file explorer and contains the text: "Und wir legen eine urls.py an..."

At the bottom of the Notepad++ window, the status bar shows: "Python file", "length : 135 lines : 10", "Ln : 10 Col : 1 Pos : 136", "Windows (CR LF)", "UTF-8", and "INS".

Anlegen des Projektes



The screenshot shows a Notepad++ window with the title bar '*C:\django_tutorial\projekt1\web\views.py - Notepad++'. The menu bar includes Datei, Bearbeiten, Suchen, Ansicht, Codierung, Sprache, Einstellungen, Werkzeuge, Makros, Ausführen, Plugins, Fenster, and ?. The toolbar contains various icons for file operations and editing. The left sidebar shows a file explorer with the following structure:

- projekt1
 - lager
 - migrations
 - templates
 - lager
 - base.html
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - projekt1
 - __pycache__
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - web
 - __pycache__
 - migrations
 - templates
 - web
 - base.html
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - db.sqlite3
 - manage.py

The main editor area shows the following code:

```
1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def index(request):
6     return render(request, 'web/base.html')
7
8
9
```

A red speech bubble points to the 'views.py' file in the file explorer, containing the text: "...in der wir ebenfalls eine *index* Funktion aus der *views.py* der App web aufrufen."

The status bar at the bottom shows: Python file | length : 150 lines : 9 | Ln : 9 Col : 5 Pos : 151 | Windows (CR LF) | UTF-8 | INS

Anlegen des Projektes

The screenshot shows a Notepad++ window with the Django project structure on the left and the `urls.py` file open in the editor. The project structure includes a `lager` app under `projekt1`. The `urls.py` file contains the following code:

```
1 """projekt1 URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7 1. Add an import: from my_app import views
8 2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('lager/', include('lager.urls')),
22     path('', include('web.urls')),
23 ]
```

A red callout box with the text "In der „haupt“ urls.py verlinken wir noch die App urls.py" points to the `include('lager.urls')` line in the `urlpatterns` list.

Anlegen des Projektes

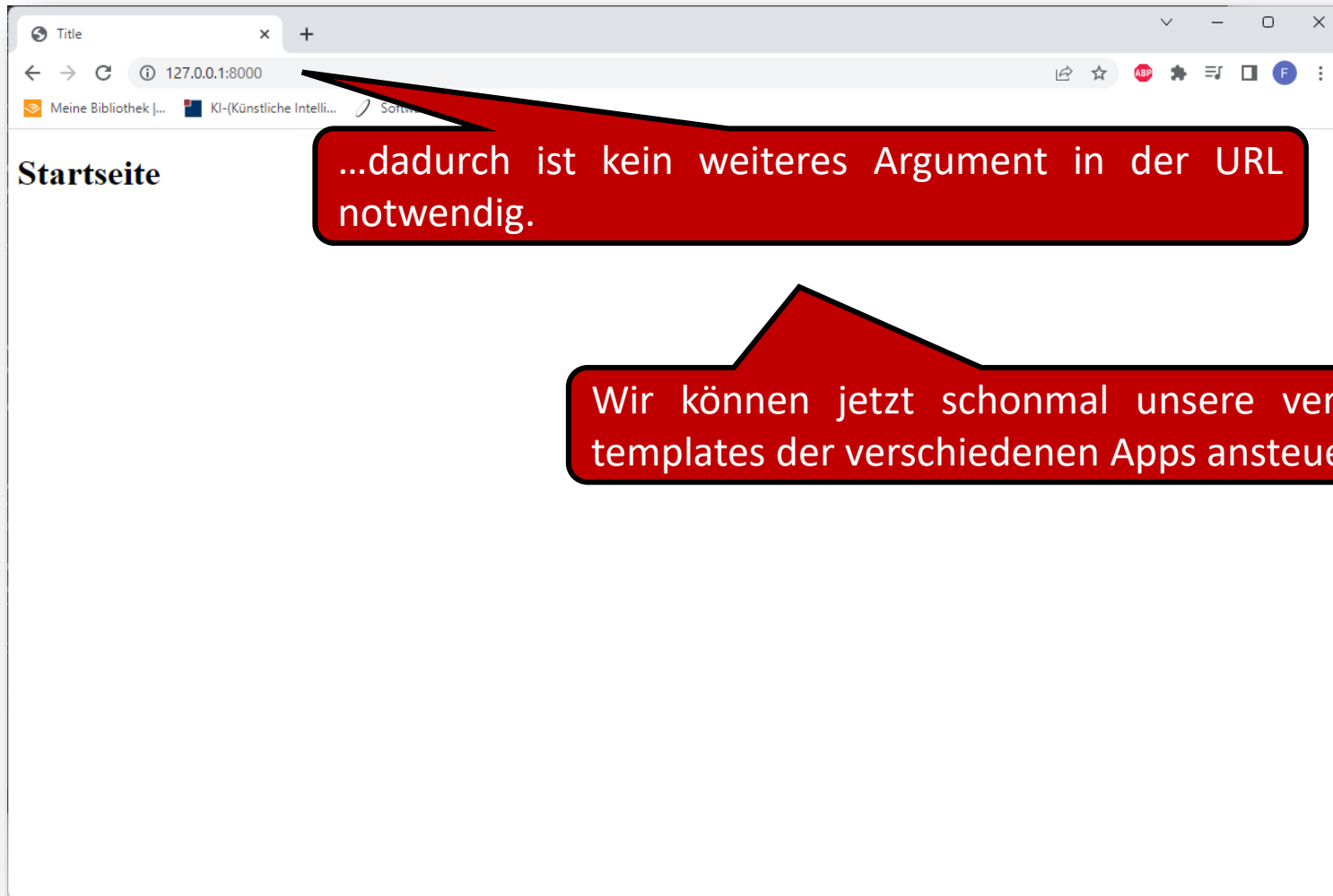
```
1 """projekt1 URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.1/topics/http/urls/
5 Examples:
6 Function views
7 1. Add an import: from my_app import views
8 2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('lager/', include('lager.urls')),
22     path('', include('web.urls')),
23 ]
24
```

Hier geben wir für das erste Argument jedoch ein leeren String an.

Anlegen des Projektes



Anlegen des Projektes



Anlegen des Projektes

```
Eingabeaufforderung - python manage.py runserver

(tutorial-env) C:\django_tutorial\projekt1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 18, 2022 - 20:08:12
Django version 4.1.1, using settings 'projekt1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[18/Sep/2022 20:08:15] "GET /lager/ HTTP/1.1" 200 184
[18/Sep/2022 20:08:23] "GET / HTTP/1.1" 200 174
```

In der Konsole können wir vom laufenden Server die jeweiligen GET Befehle beobachten, wenn eine Seite aufgerufen wird.