

A top-down view of a workspace. On the left, a silver laptop is partially visible. In the upper center, a white coffee cup with a red handle sits on a light-colored mouse. Below the mouse, a pair of tortoiseshell glasses lies on a dark grey surface. The background is a dark grey color with a light green circular shape in the top right corner and a light pink circular shape in the bottom left corner. The text is centered in a white, italicized serif font.

*MODELOVANJE*  
&  
*DOKUMENTOVANJE*  
*DJANGO APLIKACIJA*

---

# *Modelovanje Django aplikacija*

---

- UML podsetnik
- Arhitektura veb aplikacija
- Dijagrami klasa za prikazivanje Django sistema
- Dijagrami interakcije za prikazivanje veb aplikacija

# *UML*

---

UML (engl. Unified Modeling Language) - Objedinjeni jezik za modelovanje

Grafički jezik za vizualizovanje, specifikovanje, konstruisanje i dokumentovanje u nekom softverskom sistemu

Standard za šematsko prikazivanje sistema, koji obuhvata i koncepcijske aspekte (poslovni procesi i funkcije sistema) i konkretne aspekte (klase u nekom prog.jeziku, šeme baza podataka, softverske komponente koje se mogu lako ponovo koristiti)

# *Dijagram*

---

Dijagram je grafička reprezentacija skupa povezanih elemenata. Dijagram se najčešće pojavljuje u obliku grafa čvorova (stvari) povezanih granama (relacijama).

Dijagrami se crtaju da bi se sistem vizualizovao iz različitih perspektiva.

# *Vrste dijagrama*

---

Postoje:

- statički dijagrami
- dinamički dijagrami

# *Statički dijagrami*

---

Dijagram klasa (class diagram) prikazuje logičku strukturu apstrakcija: skup klasa, interfejsa, kolaboracija i njihovih relacija.

Dijagram objekata (object diagram) prikazuje logičku strukturu instanci: skup objekata (instanci klasa) i njihovih veza.

Dijagram komponenata (component diagram) prikazuje fizičku organizaciju i zavisnosti između skupa komponenata.

Dijagram raspoređivanja (deployment diagram) prikazuje konfiguraciju čvorova obrade i komponenata koje žive na njima.

Dijagram paketa (package diagram) prikazuje statičku strukturu grupisanja elemenata modela u pakete.

Dijagram složene strukture (composite structure diagram) prikazuje hijerarhijsko razlaganje složene klase (objekta) na delove.

# *Dinamički dijagrami*

---

Dijagram slučajeve korišćenja (use case diagram) prikazuje skup slučajeve korišćenja, aktera (specijalne vrste klasa) i njihovih relacija

Dijagram interakcije (interaction diagram) prikazuje jednu interakciju koju čine skup objekata i njihovih veza sa porukama koje razmenjuju

Dijagram aktivnosti (activity diagram) prikazuje tok od jedne do druge aktivnosti u sistemu (nije specijalna vrsta dijagrama stanja u UML2.0)

Dijagram stanja (statechart diagram) prikazuje konačni automat koji obuhvata stanja, tranzicije, događaje i aktivnosti

# *Arhitektura veb sajta*

---

Web sajt sadrži tri glavne komponente:

- web server,
- mrežnu konekciju,
- jedan ili više klijentskih browser-a

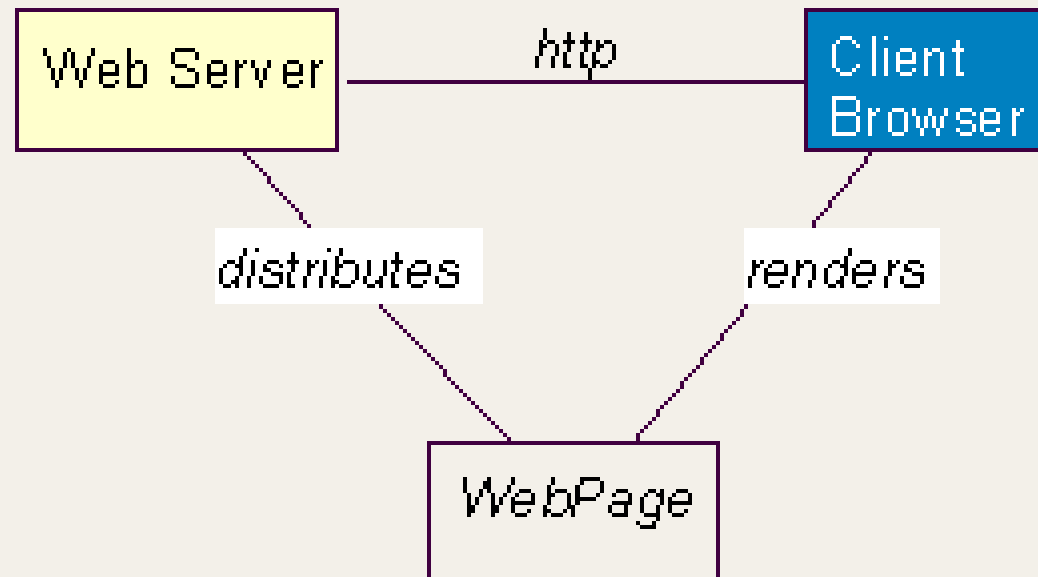
Web server distribuira (web) strane formatiranih informacija klijentima koji ih zahtevaju.

Zahtev se postavlja preko mrežne konekcije i upotrebljava HTTP protokol.



# *Arhitektura Veb sajta*

---



# *Osnovna arhitektura Web aplikacije*

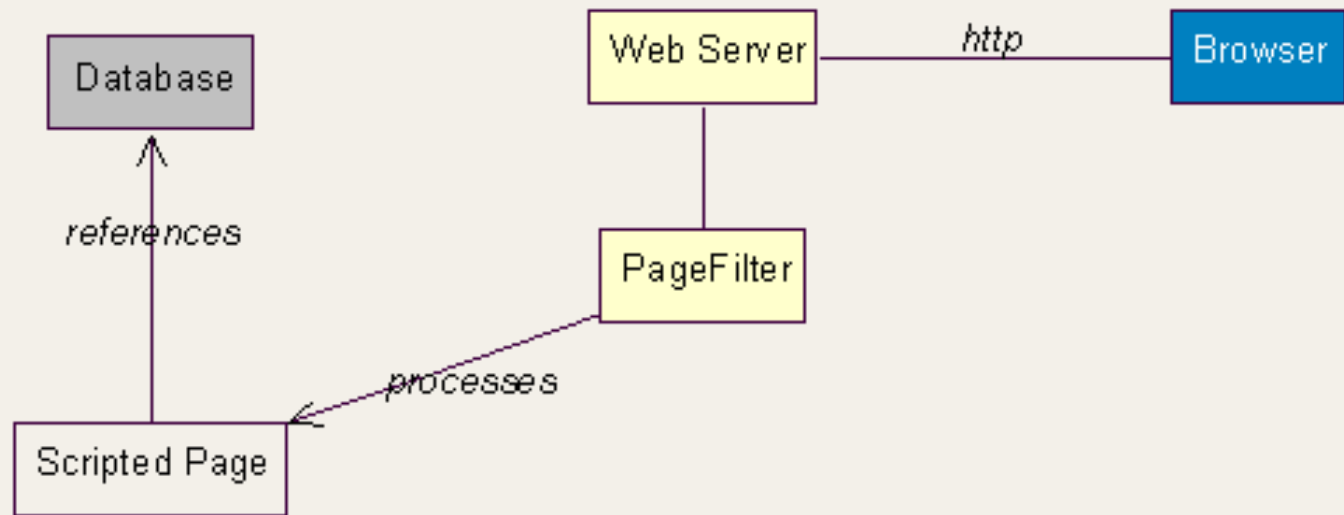
---

Informacije koje pruža Web sajt tipično su zapamćene u formatiranom obliku, u fajlovima

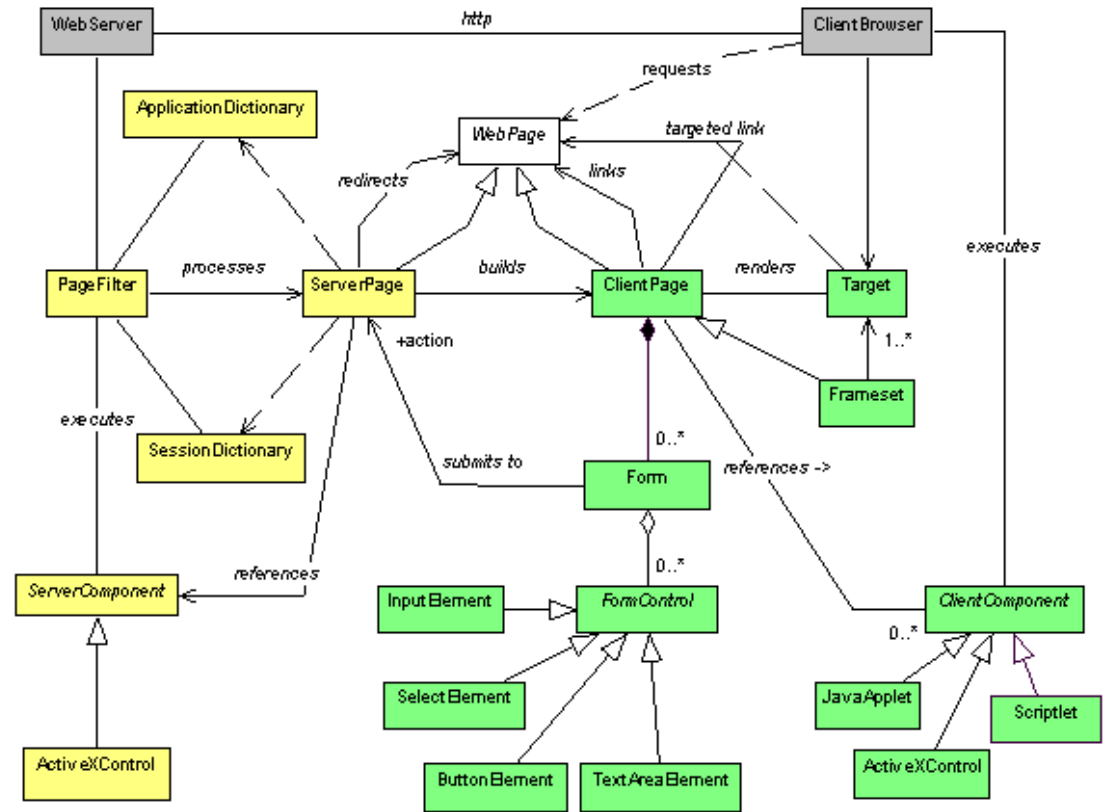
Klijent zahteva fajl po imenu i kada je neophodno pruža i informaciju o njegovoj celokupnoj putanji (adresi). Ovi fajlovi se nazivaju stranicama i reprezentuju sadržaj Web sajta.

# *Arhitektura dinamične veb aplikacije*

---



# Model generalizovane arhitekture



# *Kako modelovati Django aplikaciju?*

---

Korišćenjem UML-a.

Klasni dijagram za Django aplikacije postoji kao i za svaki drugi sistem, međutim...

Django aplikacije često nemaju klase (sva poslovna logika je implementirana putem funkcija view-ova, koji najčešće nisu u okviru klasa)

Iz ovog razloga klasni dijagram za Django aplikaciju je sledeći...

# *Dijagram modela klasa Django aplikacije*

---

Praksa nalaže da se najčešće pravi UML model za klase modela u okviru Django aplikacija

Postoji alat koji nam automatizuje ceo proces

# *Alat za generisanje dijagrama klasa modela*

---

Potrebno je instalirati paket Django extensions:

```
pip install django-extensions
```

... I dodati u settings.py fajlu u niz instaliranih aplikacija

```
INSTALLED_APPS = [  
    ...  
    'django_extensions'  
]
```

# *Korišćenje alata*

Nakon instalacije paketa django-extensions, manage.py nudi više opcija (konkretno se mogu videti u podsekciji izlistanih opcija django\_extensions)

Nama je od interesa opcija graph\_models

```
[django_extensions]
  admin_generator
  clean_pyc
  clear_cache
  compile_pyc
  create_command
  create_jobs
  create_template_tags
  delete_squashed_migrations
  describe_form
  drop_test_database
  dumpscript
  export_emails
  find_template
  generate_password
  generate_secret_key
  graph_models
  list_model_info
  list_signals
  mail_debug
  managestate
  merge_model_instances
  notes
  pipchecker
  print_settings
  print_user_for_session
  raise_test_exception
  reset_db
  reset_schema
  runjob
  runjobs
  runprofilesver
  runscript
  runserver_plus
  set_default_site
  set_fake_emails
  set_fake_passwords
  shell_plus
  show_template_tags
  show_urls
  sqlcreate
  sqldiff
  sqldsn
  sync_s3
  syncdata
  unreferenced_files
  update_permissions
  validate_templates
```



# *Rad komande graph\_models*

---

Sa opcijom --help se mogu videti sve opcije ove komande

Nama je trenutno od interesa sledeće:

Opcija -a: služi za generisanje modela iz svih aplikacija u okviru projekta (odnosno u nizu INSTALLED\_APPS)

Opcija -o: specificira naziv izlaznog fajla (u našoj situaciji će to biti bilo kako nazvan .png fajl)

Samim tim dobijamo komandu:

```
py.exe .\manage.py graph_models -a -o output.png
```

# *Šta radi ova komanda u pozadini?*

---

Generiše tekstualni fajl u predefinisanom formatu

Zatim se iz tog tekstualnog fajla generiše slika korišćenjem alata koji ume da tumači to

Alat koji služi za tumačenje ovih fajlova je GraphViz (<https://graphviz.org/>)

# *Omogućavanje rada komande*

---

Međutim da bi komanda radila na zamišljen način nedostaje par stvari:

1. Omogućavanje generisanja tekstualnog sadržaja korišćenjem komande
2. Instaliranje GraphViz-a

# *Rešenje problema 1*

---

Podrazumevano se pokretanjem komande na već opisani način javlja sledeća greška:

```
CommandError: Neither pygraphviz nor pydotplus could be found to  
generate the image. To generate text output,  
use the --json or --dot options.
```

Da bi se rešio ovaj problem postoji nekoliko rešenja, ali je najlakše instalacija nekog od zahtevanih paketa, koji će moći da generiše fajlove u zahtevanom formatu.

Instaliraćemo paket pydot, koji služi za generisanje dot fajlova, koje alat GraphViz ume da tumači.

# ***DOT***

---

Predstavlja jezik za opisivanje grafova, pa samim tim dot fajl predstavlja fajl koji sadrži informacije o nekom grafu

Više o ovom jeziku na

[https://en.wikipedia.org/wiki/DOT\\_\(graph\\_description\\_language\)](https://en.wikipedia.org/wiki/DOT_(graph_description_language))

# *Ishod problema 1*

---

Instalacijom alata pydot, graph\_models funkcionalnost će podrazumevano generisati dot fajl (moguće je i eksplicitno navoditi ovo sa opcijom --dot)

Kada ponovo pokrenemo istu komandu, javlja se naredni problem:

```
FileNotFoundError: [WinError 2] "dot" not found in path.
```

Ovo se dešava iz razloga jer komanda "ne vidi" GraphViz.

## *Rešenje problema 2*

---

GraphViz se preuzima sa sledećeg linka: <https://graphviz.org/download/>  
(dovoljno je skinuti zip arhivu)

Ali, iako smo preuzeli alat, komanda će javiti isti problem, jer još uvek ne vidi GraphViz

Ovo je iz razloga što GraphViz nije dodat u PATH environment varijablu, koja služi za direktan pristup nekoj aplikaciji/komandi

## *Rešenje problema 2*

---

Da bismo ovo dodali, postoji više rešenja, gde je najlakše sledeće:

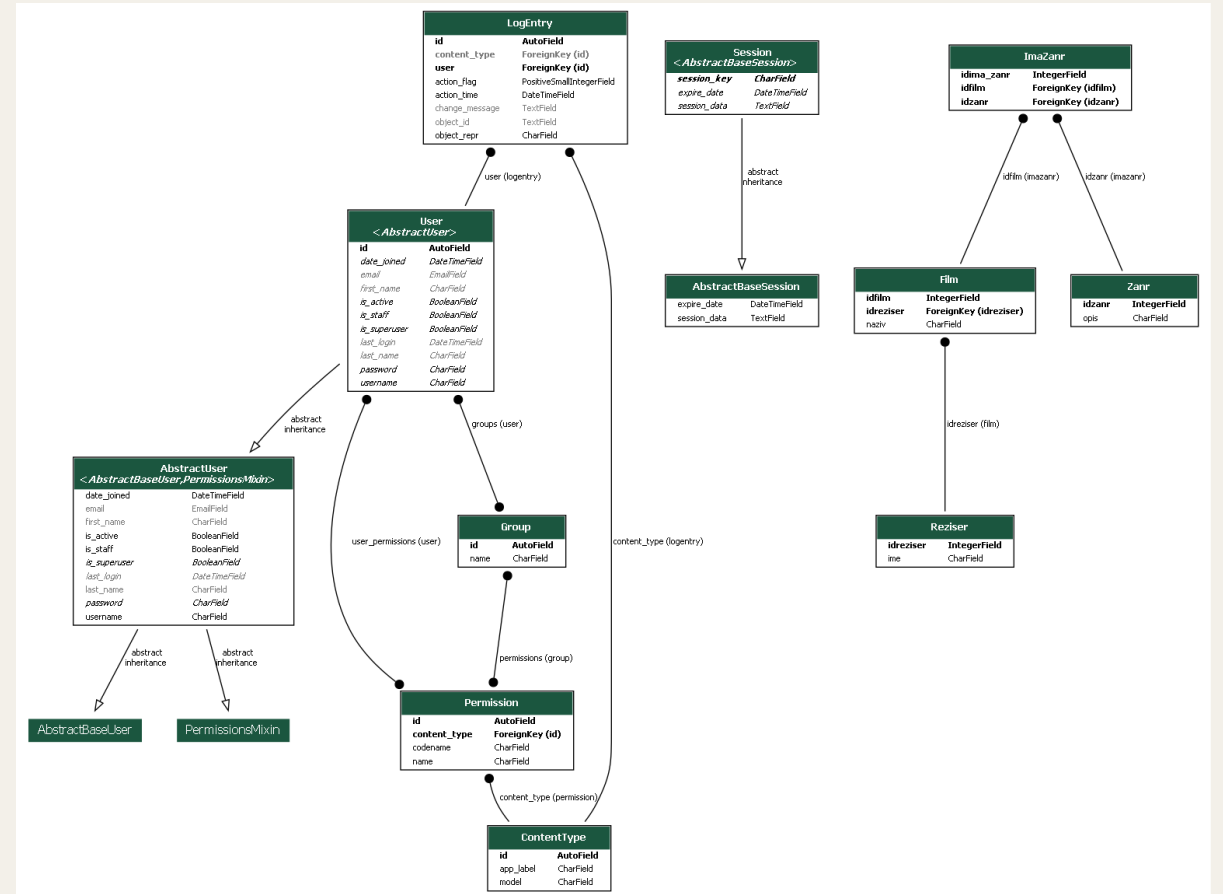
Dodati liniju koda u settings.py fajlu, da bi se sigurno uvek izvršavala pri pokretanju projekta, koja dodaje putanju ka GraphViz-u u PATH promenljivu

```
os.environ["PATH"] += os.pathsep +  
"C:\\...\\Graphviz-11.0.0-win64\\bin"
```



# Rezultat komande

Primer rezultata komande:



# *Dijagrami sekvenci*

---

Predstavljaju vrstu dijagrama interakcije

Primarni fokus im je da prikažu vremensku liniju izvršavanja programa

Daju fokus na komunikaciji među ulogama prilikom izvršavanja nekog dela programa

To se radi putem poruka, koje se šalju među ulogama (najčešće klase)

Pogodne su za prikaz rada veb aplikacija, pogotovo sa MVC/MVT arhitekturom

# Primer dijagrama sekvence:

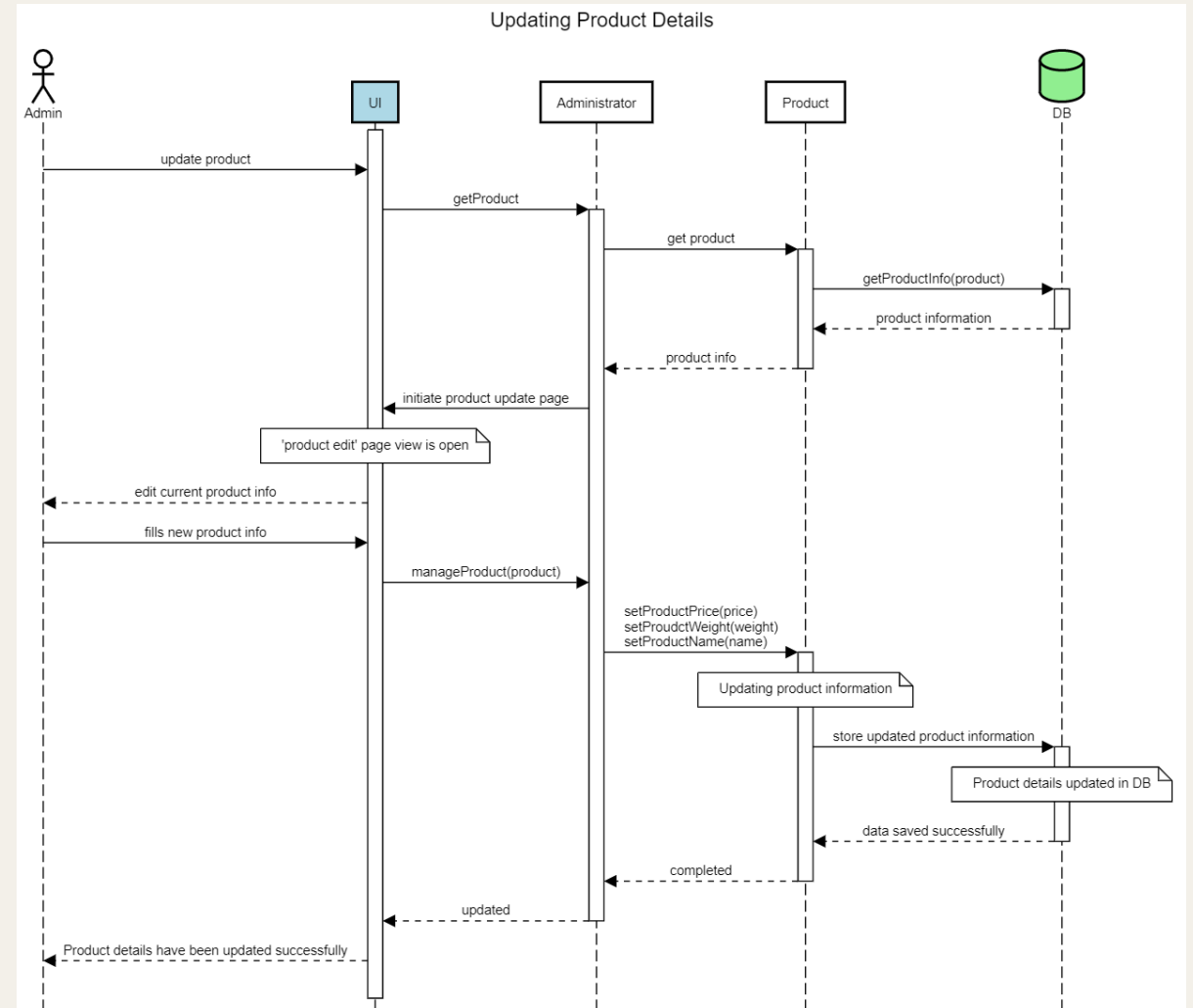
Mapiranje na MVC arhitekturu:

UI - front end, tj. View

Administrator - biznis logika, tj. Controller

Product - logika komunikacije sa bazom, tj.

Model



# *Dokumentovanje Django aplikacija*

---

- Sintaksa dokumentacije Django aplikacija
- Generisanje dokumentacije za Django admin

# *Pisanje dokumentacije*

---

Pišu se u stilu komentara ispod potpisa funkcije/klase, gde se blok komentara označava sa `"""`

Predstavlja tzv. Google Style Docstrings konvenciju

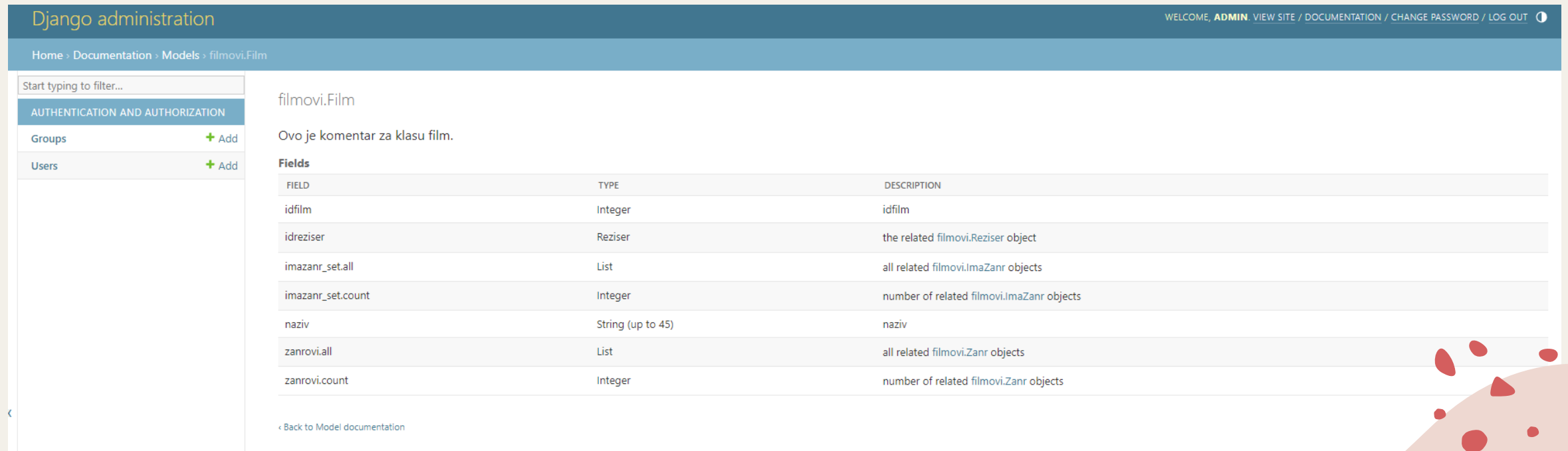
Primer dokumentacije za model:

```
class Film(models.Model):  
    """  
    Ovo je komentar za klasu film.  
    """
```

# Prikaz dokumentacije

Django radni okvir ima već implementiran izgled i sistem za prikazivanje dokumentacije aplikacije pisane u na već opisan način, u okviru administracionog panela

Primer prikaza dokumentacije za model:



The screenshot shows the Django administration interface. The top navigation bar includes the title 'Django administration' and user information: 'WELCOME, ADMIN. VIEW SITE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT'. The breadcrumb trail is 'Home > Documentation > Models > filmovi.Film'. On the left, a sidebar menu shows 'AUTHENTICATION AND AUTHORIZATION' with sub-items 'Groups + Add' and 'Users + Add'. The main content area displays the model documentation for 'filmovi.Film', including a description 'Ovo je komentar za klasu film.' and a table of fields.

FIELD	TYPE	DESCRIPTION
idfilm	Integer	idfilm
idreziser	Reziser	the related filmovi.Reziser object
imazanr_set.all	List	all related filmovi.ImaZanr objects
imazanr_set.count	Integer	number of related filmovi.ImaZanr objects
naziv	String (up to 45)	naziv
zanrovi.all	List	all related filmovi.Zanr objects
zanrovi.count	Integer	number of related filmovi.Zanr objects

[Back to Model documentation](#)

# *Omogućavanje prikaza dokumentacije*

---

Da bi se omogućilo ovo potrebno je uraditi sledeće:

1. Dodati u instalirane aplikacije `django.contrib.admindocs`
2. Dodati putanju `path('admin/doc/', include('django.contrib.admindocs.urls'))` među rute (bitno je da se doda iznad rute `'admin/'`) u niz `urlpatterns`
3. Instalirati paket `docutils`

Isto uputstvo i ostale informacije o generatoru dokumentacije je moguće videti na:  
<https://docs.djangoproject.com/en/5.0/ref/contrib/admin/admindocs/>

# *Pristup dokumentaciji*

Pristup se radi preko putanje:  
<http://localhost:8000/admin/doc/>

Izgled glavne stranice za dokumentaciju:

Home > Documentation

Start typing to filter...

## AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

## Documentation

### Tags

List of all the template tags and their functions.

### Filters

Filters are actions which can be applied to variables in a template to alter the output.

### Models

Models are descriptions of all the objects in the system and their associated fields. Each model has a list of fields which can be accessed as template variables.

### Views

Each page on the public site is generated by a view. The view defines which template is used to generate the page and which objects are available to that template.

### Bookmarklets

Tools for your browser to quickly access admin functionality.



# *Sintaksa dokumentacije*

---

Da bi stranice za različite modele i view-ove mogle da se uvežu linkovima, postoje različite notacije u dokumentaciji koje služe za uvezivanje celina

Notacija se radi putem tzv. Documentation helper-a

The following special markup can be used in your docstrings to easily create hyperlinks to other components:

Django Component	reStructuredText roles
Models	<code>:model: `app_label.ModelName`</code>
Views	<code>:view: `app_label.view_name`</code>
Template tags	<code>:tag: `tagname`</code>
Template filters	<code>:filter: `filtername`</code>
Templates	<code>:template: `path/to/template.html`</code>

# Primer dokumentacije i rezultata

Home > Documentation > Views > filmovi.views.index

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

filmovi.views.index

**Primer:**

Prikazuje naslovnu stranicu sa filmovi.Film

Display an individual myapp.MyModel.

**mymodel**

An instance of filmovi.Film.

**Template:**

index.html

[← Back to View documentation](#)

```
def index(request):
    """
    Primer:

    **Opis:**

    Prikazuje naslovnu stranicu sa :model:`filmovi.Film`

    Display an individual :model:`myapp.MyModel`.

    ``mymodel``
    An instance of :model:`filmovi.Film`.

    **Template:**

    :template:`index.html`
    """
```