Sisteme Distribuite

Mihai Zaharia

Cursul 14

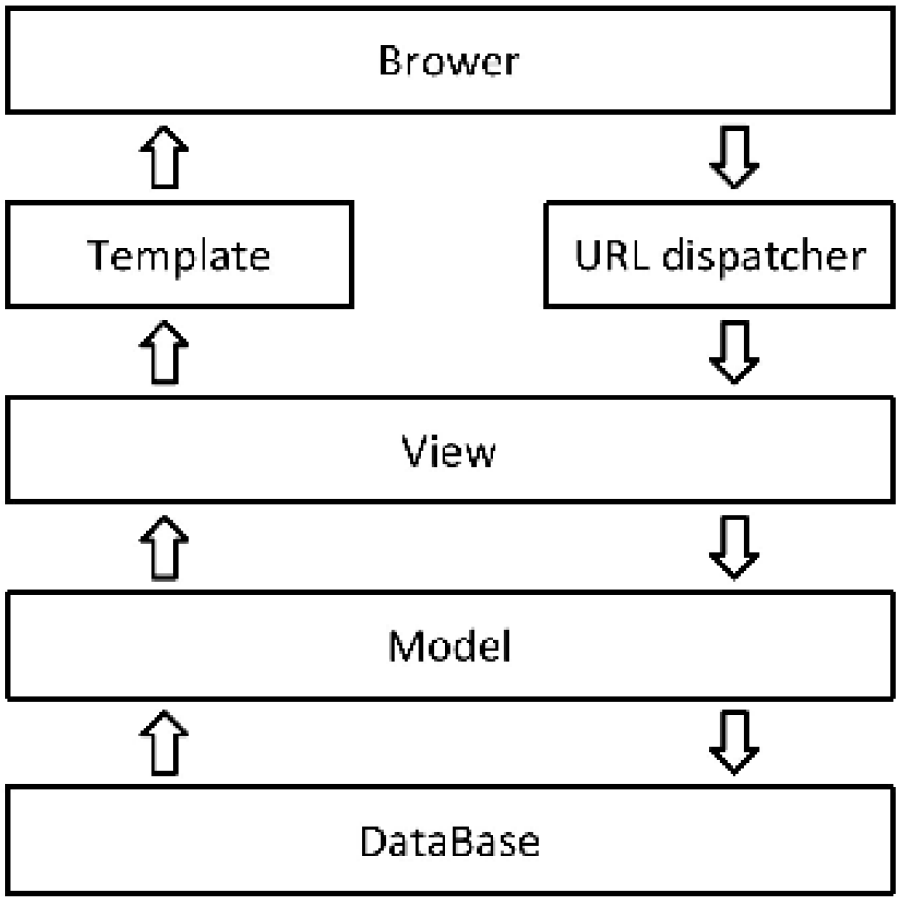
Caracteristici

- Respectă principiile paradigmei modulare
- utilizează modelul MCV(MVT)
- Furnizează interfe'e de administare
- Furnizează o multitudine de API-uri
- Furnizează un sistem de gestiune a scheletelor
- Respectă principiul DRY (Don't repeat yourself)
- Respectă principiul cuplarii scazute
- Respectă principiul separarii activitatilor transversale (concerns)
- consistența
- flexibilitate
- securitate

Cine il foloseste?

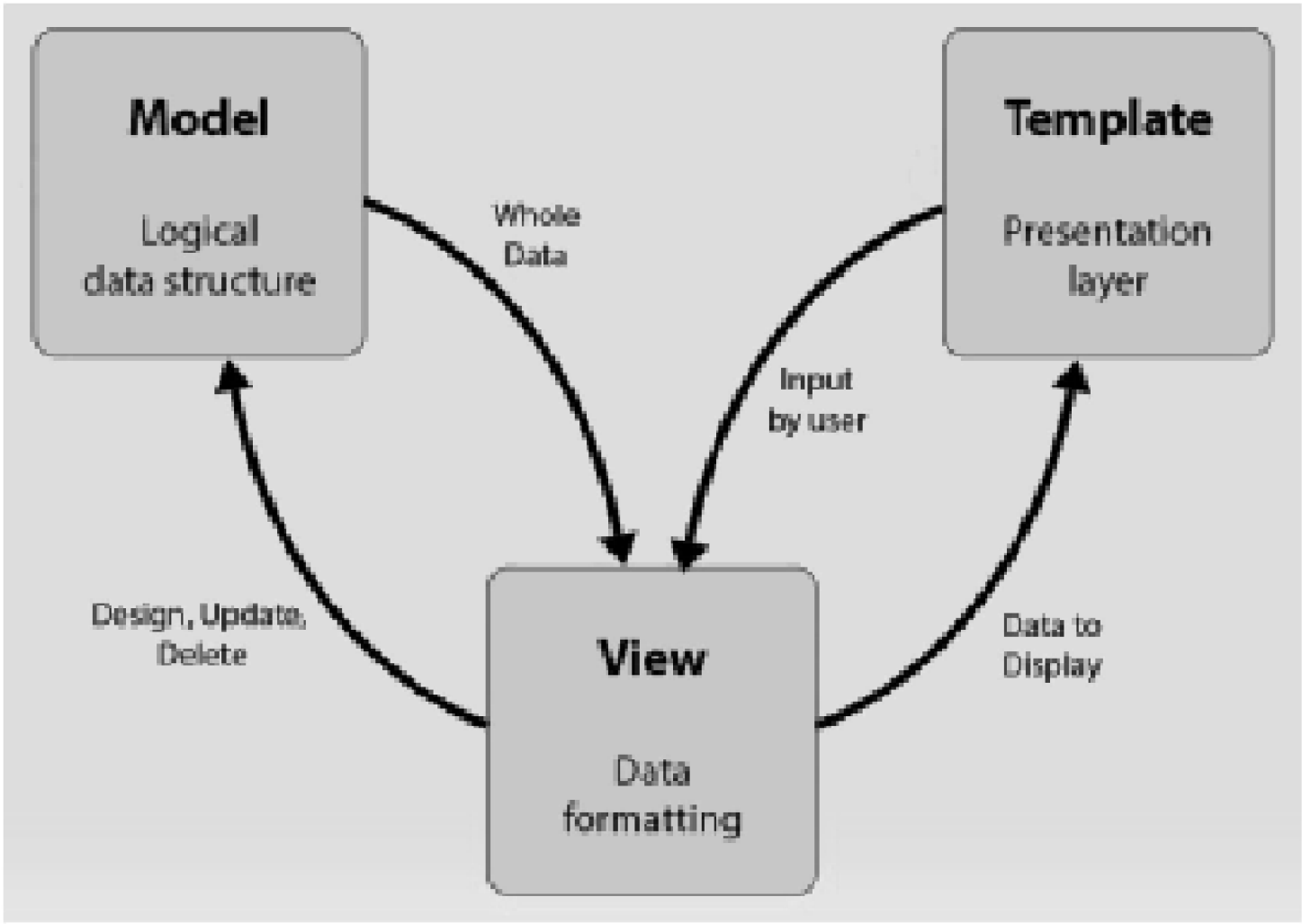
- Instagram
- Disqus
- Spotify
- YouTube
- NASA
- etc
- pt detalii vezi la Django Sites database

Model - Vizualizare - Schelet (MVT)



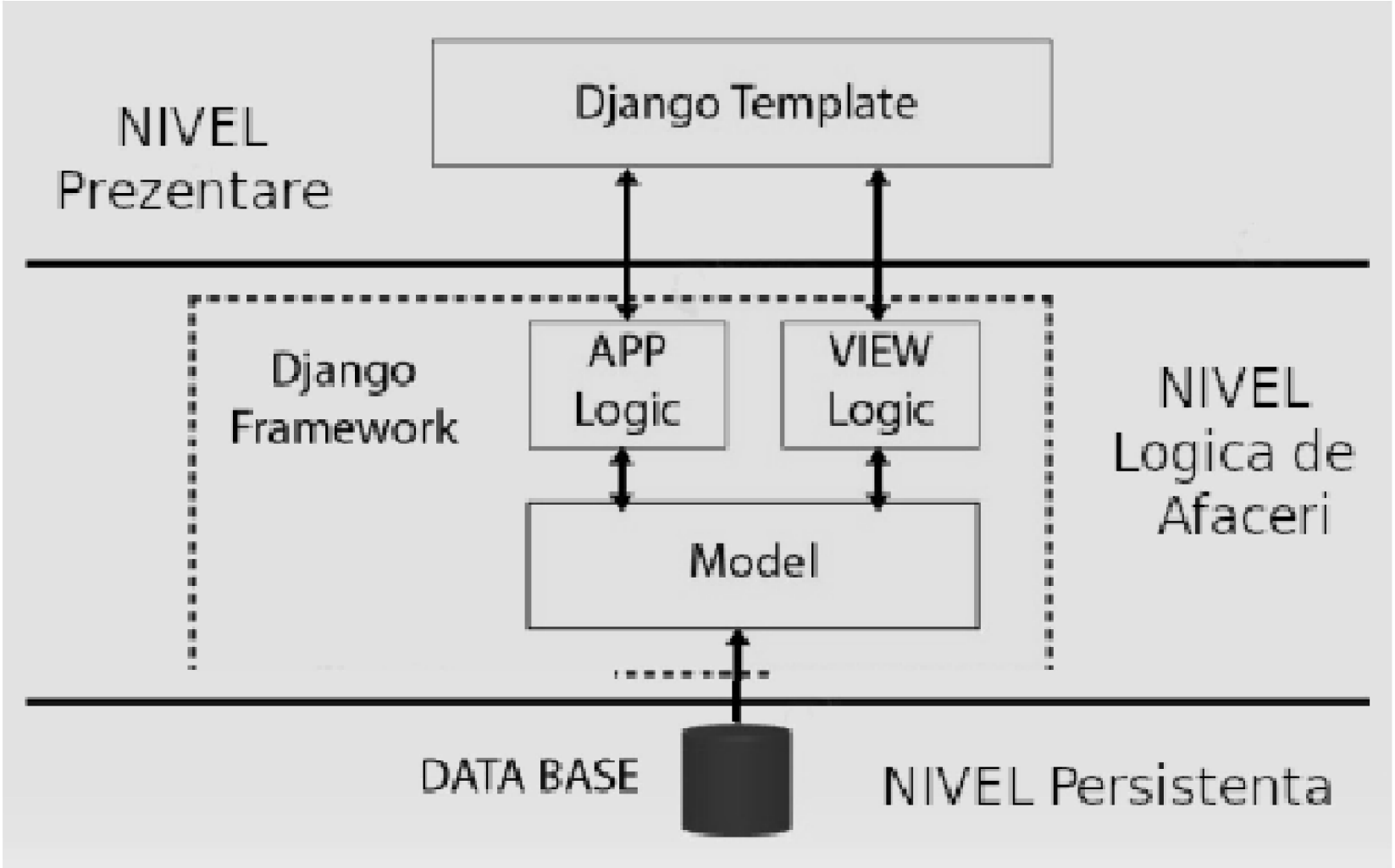
- X

MVT - detalii



- x

MVT - detalii



- X

Flask vs Django?

	django	Flask
Type of Framework	Full Stack Web Framework	WSGI framework
Flexibility	Feature-packed	Full flexibility
ORM Usage	Built-in ORM	SQLAlchemy is used
Design	Batteries-included	Minimalistic design
Working Style	Monolithic	Diversified

Modificări settings.py

- TEMPLATES = [
  {
  'BACKEND':
  'django.template.backends.django.DjangoTemplates',
  'DIRS': [os.path.join(BASE_DIR, '_templates')],
  'APP_DIRS': True,
  'OPTIONS': {
  'context_processors': [
  'django.template.context_processors.debug',
  'django.template.context_processors.request',
  'django.contrib.auth.context_processors.auth',
  'django.contrib.messages.context_processors.messages',
  ],
  },
  },
  ]

Modificări în settings.py

- Dacă doresc un director static:

```
# necesar când va fi mutat în producție
# STATIC_ROOT = os.path.join(BASE_DIR, '_static')

STATICFILES_DIRS = (
   os.path.join(BASE_DIR, '_static'),
)
```

- Modificați urmatoarele linii:
  - DEBUG = False
    ALLOWED_HOSTS = ['localhost','127.0.0.1']
- Adaugați noile aplicații în INSTALLED_APPS:
  - 'signup',

Modificări în views.py

- from django.shortcuts import render,
  render_to_response, RequestContext
  from django.contrib import messages

  def home(request):
  # Render page
  return render_to_response("index.html", locals(),

  context_instance=RequestContext(request))
  return render(request, "index.html", context)

models.py

```
import uuid

class signupModel(models.Model):
    first_name = models.CharField(max_length=120, null=False, blank=False)
    last_name = models.CharField(max_length=120, null=False, blank=False)
    email_address = models.EmailField(max_length=120, null=False, blank=False)
    tech256_username = models.CharField(max_length=120, null=True, blank=True)
    webste_url = models.URLField(max_length=200, null=True, blank=True)
    # Defineste variabilele pentru optiuni
    newsletter_nu = 'NU'
    newsletter_da = 'DA'
    newsletter_options = (
        (newsletter_nu, 'Nu - nu-mi plac gunoaielel'),
        (newsletter_da, 'Da - trimite-o catre contul de gunoaie'),
 )
    # Coloane implicite ale bazei de date si utilizarea variabilelor definite anterior
newsletter_preference = models.CharField(max_length=4, choices=newsletter_options,
default=newsletter_da)
    talk_description = models.TextField(null=False, blank=False)
    active = models.BooleanField(default=True)
    sid = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False

    def __str__(self):
        return self.sid
```

forms.py

- from django import forms
  from .models import signupModel

  class SignupForm(forms.ModelForm):
      class Meta:
          model = signupModel
          fields =
  ['first_name','last_name','email_address','tech256_use
  rname','webste_url','newsletter_preference','talk_desc
  ription','active']

Modicare views.py

- ```
  # Creati aici vizualizarile (views).
  from django.shortcuts import render, render_to_response, RequestContext
  from django.contrib import messages

  from .forms import SignupForm

  def home(request):
      form = SignupForm(request.POST or None)
      if request.method=="POST":
          if form.is_valid():
              human = True
              save_it = form.save(commit=False)
              save_it.save()
              messages.success(request,"Bine că v-ati inregistrat!")
          else:
              form = SignupForm(request.POST or None)
              messages.error(request,"OOPS o eroare regretabila.")
      else:
          form = SignupForm(request.POST or None)

      # Renderul paginii
      return render_to_response("signupform.html", locals(),
  context_instance=RequestContext(request))
      return render(request, "signupform.html", context)
  ```
  ```

Crearea scheletelor

- index.html
  - {% if messages %}
    {% for message in messages %}
    <ul {% if message.tags %} class="{{ message.tags }}"{% endif %}>
    {{ message }}

    {% endfor %}
    {% endif %}
    {% block content %}{% endblock %}
- signuform.html
  - {% extends 'index.html' %}
    {% block content %}
    <form method="POST" action='#'> {% csrf_token %}
    {{ form.as_p }}
    <input type='submit' value='Sign Up!'>
    </form>
    {% endblock %}

admin.py

```
from .models import signupModel

class signupAdmin(admin.ModelAdmin):
    list_display =
["sid","first_name","last_name","email_address","tech2
56_username”]
    class Meta:
        model = signupModel

admin.site.register(signupModel, signupAdmin)
```

Applicarea unui schelet

- Copie peste elementele statice din directorul _static.
- Copie codul HTML peste index.html și plasează o formă.
- Modică calea către URL-ul directorului_static
- Modifică CSS-ul pentru a corecta elementele.
- Vizualizează în formatele 0Desktop și Mobile.