ПРИМЕНА РАДНОГ ОКВИРА DJANGO У АГИЛНОМ ВЕБ ПРОГРАМИРАЊУ

Перица Штрбац[1] *Вукман Корађ* [2] *Милош Пејановић* [3]

Резиме: У оквиру предмета Програмирање у интегрисаним технолгијама на АТУСС - Одсек ВИШЕР који је оријентисан ка програмском језику *Python* у делу предмета који се односи на веб програмирање предвиђено је коришћење веб радног оквира *Django*. Овај радни оквир служи за развој великих веб апликација. Идеја је да студенти савладају агилан развој веб апликација коришћењем *Django* радног оквира на прагматичан начин. Студенти на поменутом предмету пре коришћења радног оквира *Django* уче програмски језик *Python* кроз елементарни увод, паралелено програмирање, метапрограмирање, базе података, клијент-сервер архитектуру и математичка израчунавања. Циљ овог рада је да се прикаже један приступ увођења студената у агилно веб програмирање коришћењем радног оквира *Django* са малим ослањањем на знања из *HTML* и *JavaScript* језика. У раду је дат илустративни пример креирања веб апликације за продају аутомобила који служи као практичан пример студентима за разумевање агилног веб програмирања. Коришћена је *MTV* (енг. *Model Template View*) архитектура. У датом примеру приказани су: модели, погледи, шаблони, контролер, корпа за куповину, контекст процесор, сесије те асинхронизам.

Кључне речи: *Django*, веб програмирање, *Python*, агилан развој апликација, *MTV*

USAGE OF DJANGO FRAMEWORK IN AGILE WEB PROGRAMMING

Abstract: Within the course Programming in Integrated Technologies at ATUSS - Department of VISHER oriented towards the Python programming language, the part of the course related to web programming envisages the use of the Django web framework. This framework is used to develop large web applications. The main idea is for students to master the agile development of web applications by pragmatically using the Django framework. Students in the mentioned course learn the Python programming language through an elementary introduction, parallel programming, metaprogramming, databases, client-server architecture, and mathematical calculations before using the Django framework. This paper aims to present an approach to introducing students to agile web programming using the Django framework with a little reliance on knowledge of HTML and JavaScript. The paper gives an illustrative example of creating a web application for car sales that serves as a practical example for students to understand agile web programming. The MTV (Model Template View) architecture is used. This example shows models, views, templates, controller, shopping cart, context processor, sessions, and asynchronism.

Key words: Django, web programming, Python, agile application development, MTV

1. УВОД

Предмет Програмирање у интегрисаним технологијама, који се слуша у трећем семестру мастер струковних студија Академије техничко-уметничких струковних студија на одсеку Висока школа електротехнике и рачунарства у Београду, обухвата изучавање и примену *Python* технологија које се траже на ИТ тржишту. Настава је оријентисана ка практичном приступу где се користе снипети кода или комплетни кодови за дате наставне теме. Теме су: елементарни увод у Python, базе података,

[1] Професор, АТУСС – Одсек Висока школа електротехнике и рачунарства, Војводе Степе 283, Београд, pericas@viser.edu.rs:
[2] Асистент, АТУСС – Одсек Висока школа електротехнике и рачунарства, Војводе Степе 283, Београд, vkorac@viser.edu.rs:
[3] Предавач, АТУСС – Одсек Висока школа електротехнике и рачунарства, Војводе Степе 283, Београд, pejanovicm@viser.edu.rs:

паралелено програмирање, метапрограмирање, клијент-сервер архитектура, математичка израчунавања, процесирање слике те веб програмирање. У оквигу веб програмирања предвиђен је елементарни *CGI* те радни оквири *FLASK* и *Django*.

Радни оквир *Django* је у оквиру предмета предвиђен као радни оквир за агилно веб програмирање у коме се креира комплетна веб апликација. Радни оквир *FLASK* (Aggarwal 2019), који у предмету претходи радном оквиру *Django*, је предвиђен за креирање малих програма на којима студенти стичу практична искуства и креирају ресурсе које ће користити када пређу на *Django* при чему користе *Jinja2*, *HTML5* і *CSS3* (Frain 2020, Meloni 2018).

Основни проблем је организовати да студенти у оквиру једног предавања и једних вежби креирају целу веб апликацију али на такав начин да би на другом пару предавања-вежбе, уз сугестију професора или асистента, могли да креирају целу нову веб апликацију према датим захтевима при чему могу додатно да примене било шта што је потребно из претходних лекција.

У складу са претходним, одабрано је да се креира *Django* пројекат за продају аутомобила који обухвата: инсталацију радног оквира, креирање потребних апликација и на крају имплементацију асинхронизма. Називи у пројекту су на српском језику чиме студенти разликују оно што се генерише аутоматски и оно што је потребно кодовати.

2. ПРОЈЕКАТ „*ProdajaAutomobila*“

Радни оквир *Django* представља *Python* радни оквир за агилно веб програмирање. Неке од карактеристика су: *Full-stack* технологија, отворени код, бесплатан, уграђена ауторизација, подршка за *URL* мапирање, подршка за *Jinja2* шаблоне, подржава базе података, подршка за *OPM* мапирање, базиран на *MTV* парадигми те велика скалабилност. Користи се за развој великих веб апликација као што су *Instagram*, *Pinterest*, *Disqus*, *Spotify* или *Bitbucket*.

2.1. Инсталација радног оквира *Django*

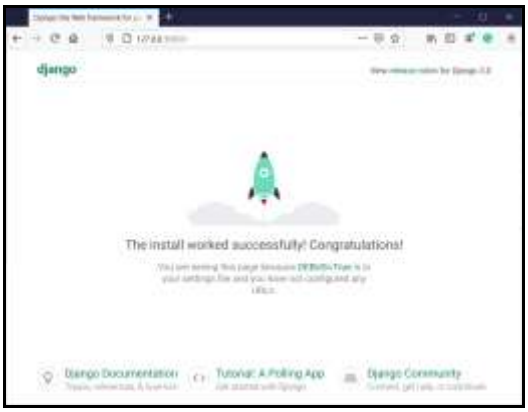
Када је реч о инсталацији, могућа су два начина: стандардна инсталација и инсталација у изолованом виртуелном окружењу. Креирање виртуелног окружења се постиже *cmd* позивом *python -m venv my_venv* где је *my_venv* назив виртуелног окружења и директоријума где ће бити инсталирано то виртуелно окружење. Да бисте активирали креирано виртуелно окружење, користи се *cmd* позив: *source my_env/bin/activate* након чега се добија промпт чији назив одговара називу виртуелног окружења. За деактивацију овог окружења користи се позив *deactivate*. Инсталације у витуеланом окружењу се смештају, за дат пример, у фолдер *my_env/lib/python3.8/site-packages*. Инсталација *Django* оквира се изводи позивом *pip install “Django==3.0.*”* . На крају студенти проверавају да ли је инсталација коректна из *Python* интерактивног промпта (*import django; django.get_version()*).

2.2. Креирање *Django* пројекта *ProdajaAutomobila*

Сада студенти креирају *Django* пројекат. Да би се креирао *Django* пројекат *ProdajaAutomobila* користи се позив *django-admin startproject ProdajaAutomobila*. Директоријум *ProdajaAutomobila* је сада креиран у директоријуму у коме је наведен дати позив. У оквиру овог директоријума су датотека *manage.py* и опет директоријум *ProdajaAutomobila*. Датотека *manage.py* је услужни програм који омогућује прихватање командне *Django* линије за интеракцију са пројектом. У директоријуму *ProdajaAutomobila*/*ProdajaAutomobila* се налазе следеће датотеке: *asgi.py* (конфигурација за асинхрони веб сервер *ASGI*), *settings.py* (иницијална подразумевана

подешавања пројекта), *urls.py* (*URL* мапирање у view), *wsgi.py* (конфигурација за веб сервер гејтвеј интерфејс *WSGI*), *__init__.py* (директоријум *ProdajaAutomobila/ ProdajaAutomobila* је модул). Студенти се упознају са поставкама датим у датотеци *settings.py* (*DEBUG* дебаг мод; *ALLOWED_HOSTS* – домен за продукцију; *INSTALLED_APPS* - активне апликације пројекта, иницијално су: django.contrib.admin (администрација сајта), django.contrib.auth (аутентификација), django.contrib.contenttypes (управљање садржајем), django.contrib.sessions (управљање сесијама), django.contrib.messages (управљање порукама), django.contrib.staticfiles (управљање статичким фајловима); *MIDDLEWARE* – листа коришћених middleware, *ROOT_URLCONF* – корен *URL* патерна; *DATABASES* – речник подешавања база података пројекта; *LANGUAGE_CODE* – подразумевани језик сајта; *USE_TZ* - (де)активација подршке за временску зону).

За активацију миграције базе података (подразумевано се користи база *SQLite3*), односно, за креирање табела према иницијалној апликацији, из директоријума *ProdajaAutomobila* користи се позив *python manage.py migrate* (сада студенти уочавају датотеку *db.sqlite3*). Након овога студенти из директоријума *ProdajaAutomobila* активирају развојни сервер позивом *python manage.py runserver* и у коришћеном веб претраживачу иду на адресу 127.0.0.1:8000 где се налази поздравна страница (слика 1).



Слика 1 – Поздравна страница

2.3. Креирање апликација пројекта

У оквиру креирања апликација студенти креирају три апликације у пројекту *ProdajaAutomobila* које су за овакав тип пројекта уобичајене, а то су: *SalonAutomobila*, *KorpaZaKupovinu* и *Porudzbina* (Mele 2020). Апликација *SalonAutomobila* намењена је за вођење података о аутомобилима. *KorpaZaKupovinu* омогућује функционалност да корисник може онлајн да одабере шта купује и у којој количини, нпр. набавка више возила за рентакар. Апликација *Porudzbina* омогућује да се уради поруџбина одабраних аутомобила. Све статичке ресурсе: *CSS*, *HTML Jinja2* шаблоне и слике студенти добијају готове, при чему се од њих очекује да за домаћи радураде измене по жељи али у складу са модерним дизајном (Esposito 2016, Којић 2018).

2.3.1. Апликација *SalonAutomobila*

Креирање апликације *SalonAutomobila* у оквиру пројекта *ProdajaAutomobila* ради се тако да се из директоријума *ProdajaAutomobila* уради позив *django-admin startapp SalonAutomobila*. Овим је креиран директоријум *SalonAutomobila* који садржи следеће датотеке и директоријум у које студенти буду упућени: *admin.py* (за администрацију,

регистровање модела); *apps.py* (главна конфигурација апликације), фолдер *migrations/* директоријум у коме ће бити миграције базе података које омогућују праћење промене модела и потребну синхронизацију са базом података), *models.py* (модели апликације), *tests.py* (за тестирање апликације), *views.py* (логика која прихвата HTTP захтев, обрађује га и враћа одговор), *__init__.py* (директоријум *SalonAutomobila* је модул). Студенти након овог постављају да је апликација *SalonAutomobila* активна додавањем ставке у листу *INSTALLED_APPS* пројеката, дакле *INSTALLED_APPS = [django.contrib.admin', ... , 'SalonAutomobila.apps.SalonautomobilaConfig',]*. Klasa *SalonautomobilaConfig* је дефинисана у датотеци *SalonAutomobila/apps.py*.

Потребно је да студенти креирају моделе апликације *SalonAutomobila*. Предвиђена су два модела: модел за сегмент аутомобила (од малог градског аутомобила до теренца) (слика 2) и модел за податке о аутомобилу (слика 3). Модели се креирају у датотеци *SalonAutomobila/models.py* као класе које наслеђују класу *models.Model* (потребан увоз је *from django.db import models*). Овде је потребно креирати методу која враћа апсолутну *URL* адресу објекта (*get_absolute_url*) која је потребна за шаблоне (користи се *from django.urls import reverse*).

```
class Segment(models.Model):
   naziv = models.CharField(max_length=100, db_index=True) # naziv segmenta
   slug = models.SlugField(max_length=100, unique=True) # slug za SEO
   class Meta:
      ordering = ('naziv',) # sortiranje po rastucem redosledu atributa naziv
      verbose_name = 'segment' # jednina
      verbose_name_plural = 'segmenti' # mnozina
   def __str__(self): return self.naziv # string reprezentacija objekta
   def ApsolutniURL(self): # apsolutna url adresa objekta potrebna za sablone
      return reverse('SalonAutomobila: ListaAutomobilaPoSegmentu',
                     args=[self.slug])
```

Слика 2 – Класа модела Segment

```
class Automobil(models.Model):
   segment = models.ForeignKey(Segment,
                               related_name='automobili',on_delete=models.CASCADE)
   # za vezu 1:N Segment:Automobil, brisanje zapisa je kaskadno
   naziv = models.CharField(max_length=100, db_index=True)
   slug = models.SlugField(max_length=100, db_index=True) # slug za SEO
   cena = models.DecimalField(max_digits=9, decimal_places=2)
   opis = models.TextField(blank=True) # inicijalno bey opisa
   slika = models.ImageField(upload_to='automobili/%Y/%m/%d',blank=False)
   # slika je obavezno polje i snima se u folder odredjen datumom
   raspoloziv = models.BooleanField(default=True) # ima li tog automobila u salonu
   class Meta:
      ordering = ('naziv',) # sortiranje po rastucem redosledu atributa naziv
      index_together = (('id', 'slug'),) #indeksiranje u paru atributa id i slug
      verbose_name_plural = 'automobili'
   def __str__(self): return self.naziv
   def ApsolutniURL(self): # apsolutna url adresa objekta potrebna za sablone
      return reverse('SalonAutomobila:DetaljiAutomobila',
                     args=[self.id, self.slug])
```

Слика 3 – Класа модела Automobil

Након овога, студенти креирају оба модела тако што се у директоријуму *ProdajaAutomobila* позове *pyhon manage.py makemigrations* након чега се у датотеци

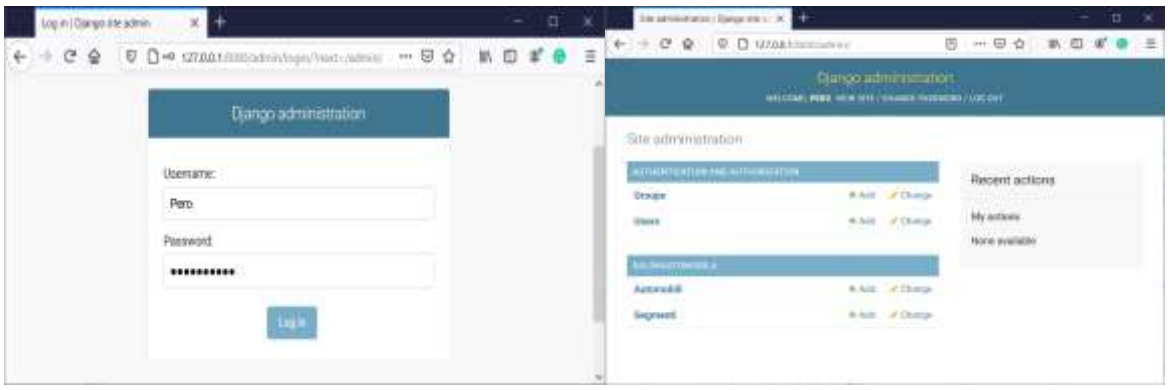
SalonAutomobila\migrations\0001_initial.py ажурира листа операција за миграцију оба модела. За синхронизацију базе према моделима из наведеног директоријума се уради позив *pyhon manage.py migrate.*

За администрацију оба модела на *admin* страници потребно је у датотеци *SalonAutomobila/admin.py* урадити регистровање модела за администрацију (слика 4). Студенти раније слушају лекцију метапрограмирање у оквиру које се објашњава примена декоратора (овде је то *@admin.register(ClassName)*).

```
from django.contrib import admin
from .models import Segment, Automobil
@admin.register(Segment)
class SegmentAdmin(admin.ModelAdmin):
   list_display = ['naziv', 'slug'] # polja koja se prikazuju
   prepopulated_fields = {'slug': ('naziv',)} # automatsko kreiranje
@admin.register(Automobil)
class AutomobilAdmin(admin.ModelAdmin):
   list_display = ['naziv', 'slug', 'cena','raspoloziv'] # polja za prikaz
   list_filter = ['raspoloziv'] # polja za filter
   list_editable = ['cena', 'raspoloziv'] #editabilna polja
   prepopulated_fields = {'slug': ('naziv',)} # automatsko kreiranje
```

Слика 4 – Датотека SalonAutomobila/admin.py

За администрацију на admin страници потребно је из директоријума *ProdajaAutomobila* креирати супер-корисника позивом *python manage.py createsuperuser* (уносе се корисничко име, имејл адреса, лозинка и поновљена лозинка). Стартовањем сервера и одласком на страницу *127.0.0.1:8000/admin* десиће се редирекција на страницу логовања *http://127.0.0.1:8000/admin/login/?next=/admin/* (слика 5.а) те ће након успешног логовања студент бити редиректован на *admin* страницу за администрацију оба модела апликације *SalonAutomobila* (слика 5.б).



Слика 5 – а) Логовање на admin страници б) Станица за администрацију модела

Страница за администрацију модела омогућује инсертовање, едитовање и брисање записа оба модела где студенти додају све сегменте (*A, B, C, D, E, F, S, M, J*) и произвољно додају атомобиле у датом сегменту (слика 6). Сада је потребно написати код у датотеци *SalonAutomobila/views.py* који омогућује да се види каталог аутомобила на страници (слика 7).

Слика 6 – а) Ставке segmenta б) Ставка аутомобила

```
from django.shortcuts import render, get_object_or_404
from .models import Segment, Automobil
# definisanje funkcije za vraćanje kataloga proizvoda prema prosleđenoj kategoriji
def ListaAutomobila(request, segment_slug=None): # vraca katalog kao html odgovor
    segment = None
    segmenti = Segment.objects.all()
    automobili = Automobil.objects.filter(raspoloziv=True)
    if segment_slug:
        segment = get_object_or_404(Segment, slug=segment_slug)
        automobili = automobili.filter(segment=segment)
    return render(request, 'SalonAutomobila/automobil/list.html',
                  {'segment':segment,'segmenti':segmenti,'automobili':automobili})
# vraćanje podatka o automobilu prema prosleđenom id  i slug automobila:
def DetaljiAutomobila(request, id, slug):
    # vraca podatke o automobilu za dati id i slug kao html odgovor
    automobil = get_object_or_404(Automobil, id=id, slug=slug, raspoloziv=True)
    return render(request, 'SalonAutomobila/automobil/detail.html',
                           {'automobil': automobil})
```

Слика 7 – Враћање каталога или података о једном аутомобилу као html одговор

За везивање *URL-view* потребно је креирати датотеку *SalonAutomobila/urls.py* и у њој написати код дат на слици 8 који повезује *URL* и view функцију (*ListaAutomobila* са *ListaAutomobila* без параметара, *ListaAutomobilaPoSegmentu* са *ListaAutomobila* са параметром segment_slug типа slug, *DetaljiAutomobila* са позивом *DetaljiAutomobila* са два параметра *id* и *slug* типа *int* и *slug* респективно*).*

```
from django.urls import path
from . import views
app_name = 'salonautomobila'
urlpatterns = [ path('', views.ListaAutomobila,      name='ListaAutomobila'),
path('<slug:segment_slug>/',views.ListaAutomobila,
                                        name='ListaAutomobilaPoSegmentu'),
path('<int:id>/<slug:slug>/', views.DetaljiAutomobila, name='DetaljiAutomobila'),]
```

Слика 8 – Повезивање URL и view функције

Да би наведено *URL* мапирање радило потребно је да се оно укључи у пројекат тако да се у датотеци *ProdajaAutomobila/ProdajaAutomobila/urls.py* дода код дат на слици 9. Проширење *URL* узорка за *DEBUG* мод омогућује да сервер опслужи постављање слике аутомобила. У датотеци пројекта *settings.py* потребно је поставити вредности за *MEDIA_URL* ('/*media*/') где се обрађују фајлови медије које поставља корисник те за

MEDIA_ROOT (*os.path.join(BASE_DIR, 'media/')*) где се локално налазе фајлови медије (*ProdajaAutomobila/media/*) .

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [ path('admin/', admin.site.urls),
      path('', include('SalonAutomobila.urls', namespace='SalonAutomobila')),]
if settings.DEBUG:
   urlpatterns += static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

Слика 9 – Повезивање URL и view функције

2.3.2. Апликација *KorpaZaKupovinu*

Апликацију *KorpaZaKupovinu* студенти креирају по узору на апликацију *SalonAutomobila*. У апликацији *KorpaZaKupovinu* студенти се упознају са сесијом за комуникацију са корисником. У корпи се за сваки одабрани аутомобил смештају јединствени број аутомобила, цена аутомобила и број аутомобила који се купује (наручује). Конструктор класе *KorpaZaKupovinu* дат је на слици 10. Студенти креирају методе за додавање претходно наведених података о аутомобилу у корпу, снимање корпе, уклањање аутомобила из корпе, приказ садржаја корпе и вредности у корпи те брисање корпе из сесије. Сада се ради креирање view за корпу за стављање у корпу, уклањање из корпе те приказ садржаја корпе. За *view* стављања у корпу потребна је и *Django* форма помоћу које корисник бира колико комада датог аутомобила ће ставити у корпу. Обрађиваће се само *post* захтеви корисника коришћењем декоратора *@require_POST*. У овој апликацији у *view* за преглед детаља производа додаје се форма за одабирање броја аутомобила те програмско дугме за додавање аутомобила у корпу.

```
from django.conf import settings
class KorpaZaKupovinu(object):
 def __init__(self, request):
   self.session = request.session
   korpa = self.session.get(settings.KORPAZAKUPOVINU_SESSION_ID)
   if not korpa: korpa=self.session[settings.KORPAZAKUPOVINU_SESSION_ID]={}
   self.korpa= korpa
```

Слика 10 – Конструктор класе Korpa Za Kupovinu

Креира се *context processor* за корпу који за прослеђени захтев враћа једноставан речник који има пар кључ-вредност као назив-објекат креиран позивом конструктора корпе чији је параметар наведени захтев. Овим је креирани објекат корпе доступан шаблонима под датим називом. У датотеци *settings.py* пројекта у листи *TEMPLATES* у оквиру речника чији је кључ *context_processors* потребно је додати вредност *korpa.context_processors.korpa* што омогућује да контекст корпе буде позван у сваком шаблону који се рендерује позивом *RequestContext.*

2.3.3. Апликација *Porudzbine*

У апликацији *Porudzbine* потребно је снимити податке о купцу и аутомобилу или аутомобилима које је тај купац наручио. По узору на апликацију *SalonAutomobila* студенти креирају моделе *Porudzbina* і *StavkaPorudzbine* (као међутабела за релацију

N:N Automobil:Porudzbina), покрећу иницијалну миграцију и синхронизацију креираних модела са базом података те региструју моделе у датотеци *admin.py* ове апликације. Потребно је креирати *Django* форму за детаље поруџбине која ће бити коришћена у *view* ове апликације који служи за креирање нове поруџбине, а онда додати и *URL* мапирање за креирање поруџбине те га укључити у *urls.py* пројекта.

На крају се студентима даје упутство за припрему за вежбу на којој ће имплементирати асинхронизам коришћењем *RabbitMQ* (Toshev 2015) за брокер порука и *Celery* (Solem 2020) за асинхрону обраду порука. Студенти на крају добијају као задатак даураде *Django* веб апликацију као семинарски рад који обухвата елементе показане на пројекту са предавања.

3. ЗАКЉУЧАК

У оквиру предмета Програмирање у интегрисаним технолгијама на АТУСС – Одсек ВИШЕР за поглавље предмета који се односи на агилно веб програмирање предвиђено је коришћење веб радног оквира *Django*. У раду је дат пример пројекта за агилан развој веб апликације за продају аутомобила који би се креирао у оквиру једног пара предавање-вежба. Студенти се при креирању пројекта ослањају на знања из претходних лекција из наведеног предмета (рад са базама података, метапрограмирање, паралелено програмирање, процесирање слике, *FLASK*) те усвајају *MTV* парадигму која се примењује за *Django*. У току реализације пројекта студенти креирају три веб апликације (салон аутомобила, корпу за купце и апликацију за поруџбине) при чему практично користе моделе (креирање, миграције и синхронизацију модела са базом података), погледе, сесије, шаблоне (*Jinja2*), контекст процесор, администрацију модела, креирање корпе за купце, креирање поруџбине и синхронизам порука. Предвиђено је да студенти на основу датог пројекта ураде свој пројекат како семинарски рад у коме би били заступљени сви елементи показани на пројекту у оквиру предавања, укључујући и креирање властитих статичних ресурса.

4. ЛИТЕРАТУРА

[1] Aggarwal S. (2019). *Flask Framework Cookbook – Second Edition*, Birmingham. Packt Publishing.

[2] Esposito, D. (2016). *Modern Web Development: Understanding domains, technologies, and user experience.* Microsoft Press.

[3] Frain, B. (2020). *Responsive Web Design with HTML5 and CSS - 3rd Edition*, Birmingham. Packt Publishing.

[4] Којић, Н., Никитин, Д., Михајловић, Н., Лукић, Л., Вугделија, Н. (2018). *Трендови развоја клијентских web оријентисаних језика*, 4. међународна конференција управљање знањем и информатика, Копаоник, стр. 83-88.

[5] Mele, A. (2020). *Django 3 By Example, Third Edition*, Birmingham. Packt Publishing.

[6] Meloni, J., Kyrnin, J. (2018). *HTML, CSS, and JavaScript All in One - 3rd Edition*, Sams Publishing.

[7] Solem, A. (2020). *Celery Documentation,* Ask Solem.

[8] Toshev, M. (2015). *Learning RabbitMQ*, Birmingham. Packt Publishing.