**Гимназија „Јован Јовановић Змај“**

Нови Сад

**Матурски рад из Програмирања**

**КОНТРОЛА И ПРЕТРАГА ЛЕТА СА ГРАФОВИМА**

Професор ментор: Ученик:

Милан Станковић Лазар Половина IV-5

Нови Сад, мај 2022. год.

ПРЕДГОВОР

Предговор се не наводи у садржају рада тако да је његов наслов у стилу **Naslov**, а наслови који ће ући у садржај су редом у стиловима **Heading 1**, **Heading 2** и даље. Овај текст се пише у стилу **Style\_Tekst** као и сав остали основни текст у самом раду.

САДРЖАЈ

**Страна**

[1. Теоретски део о графовима 5](#_Toc101729777)

[1.1. Врсте графа 5](#_Toc101729778)

[1.1.1. Усмерен и неусмерен граф 5](#_Toc101729779)

[1.1.2. Тежински графови 6](#_Toc101729780)

[1.2. Проналажење најкраћег пута у графу 6](#_Toc101729781)

[1.2.1. Релаксација грана 7](#_Toc101729782)

[1.2.2. Дајкстрин алгоритам претраге 7](#_Toc101729783)

[1.2.3. Флојд – Варшалов алгоритам 8](#_Toc101729784)

[2. Програмски језици и технологије које ће се користити у практичном делу рада 10](#_Toc101729785)

[2.1. Django 10](#_Toc101729786)

[2.1.1. Post и Get методе 10](#_Toc101729787)

[2.1.2. Како са базама у Django 11](#_Toc101729788)

[3. Контрола и претрага лета 12](#_Toc101729789)

[3.1. Контрола лета 12](#_Toc101729790)

[3.2. Претрага лета 14](#_Toc101729791)

[Прилог / Прилози 16](#_Toc101729792)

[Прилог 1 16](#_Toc101729793)

[Садржај CD-а 17](#_Toc101729794)

[Литература 18](#_Toc101729795)

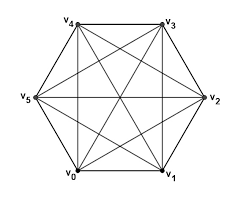
[БИОГРАФИЈА МАТУРАНТА 19](#_Toc101729796)

Увод

Теорија графова је област математике која је веома заступљена у програмирању и информатици уопште. Ова област своје корене вуче још из 18. века када је познати швајцарски математичар Леонард Ојлер формулисао проблем под називом Седам мостова Кенинзбурга. У овом огранку математике постоји много теоретског дела и због тога су и приказане неке основе које морају да се знају пре саме примене, али исто тако је и веома променљиво на реалне проблеме попут умрежавања,

# Теоретски део о графовима

**Граф** представља један од основних апстрактних структура у програмирању помоћу кога можемо да представимо објекте и релације које важе међу њима, од физичких до мисаоних. Сваки граф G се састоји од два скупа: **скупа грана** E и **скупа чворова** V.



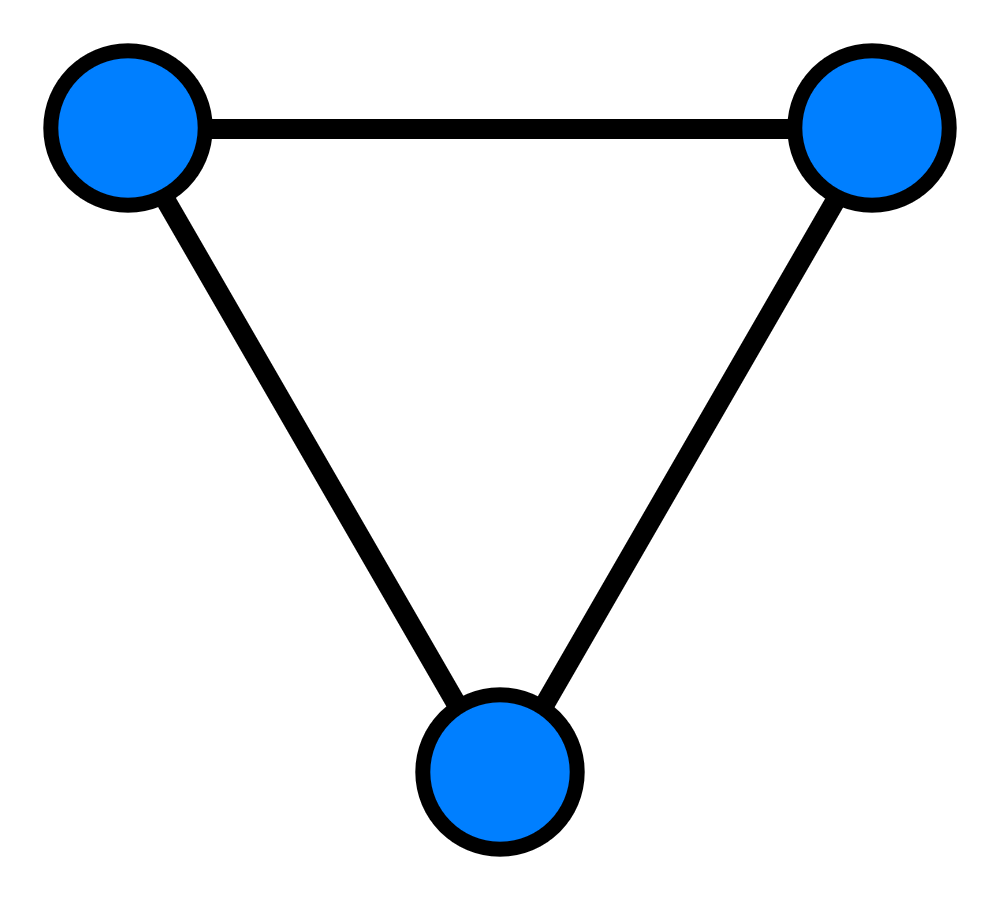
Слика Граф

На слици 1. је приказан граф од пет чворова V = {v1, v2, v3, v4, v5} и двадесет грана E = {v1-v2, v2-v3, v4-v5...}.

## Врсте графа

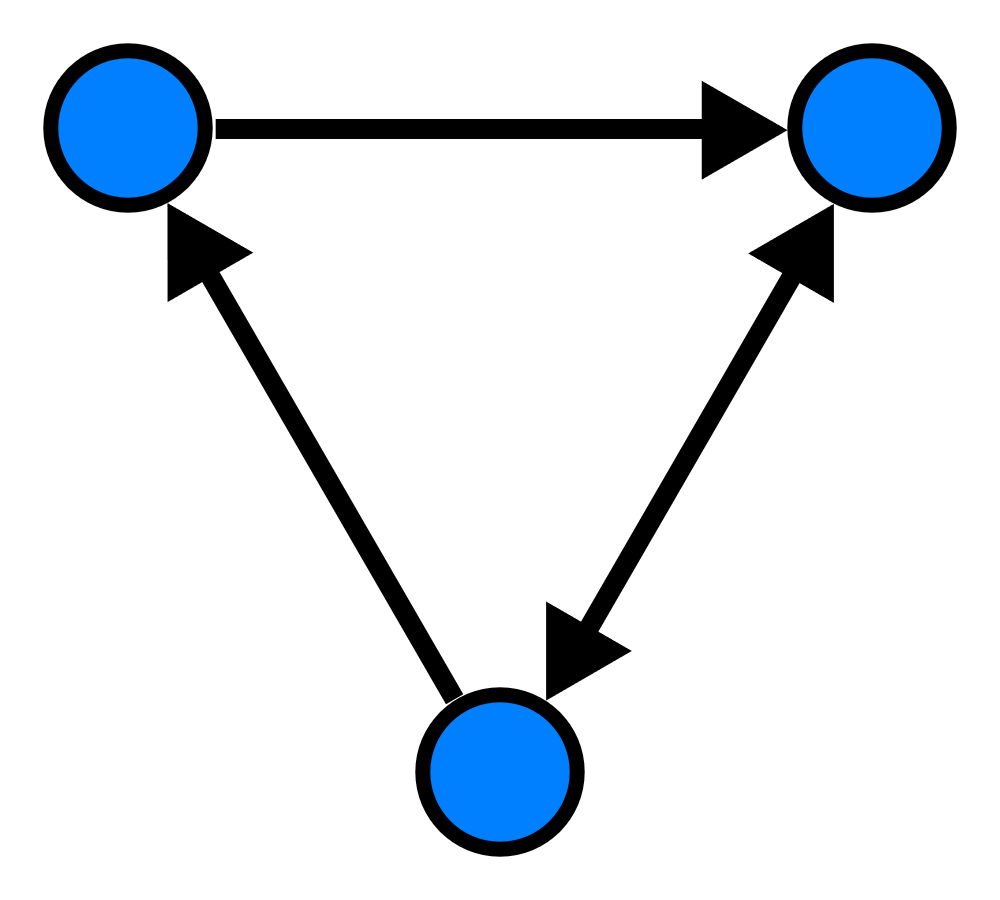
### Усмерен и неусмерен граф

Код **неусмереног графа** (слика 2) не морамо да нагласимо одакле докле иде грана и степен чвора у таквом графу је једнак броју грана везаних за тај чвор.



Слика Неусмерен граф

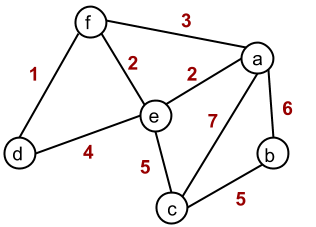
За разлику од њега код **усмереног графа** (слика 3) морамо нагласити одакле почиње и где се завршава грана. Због ове особине усмерени графови имају **улазни** и **излазни степен** за сваки чвор. Колико пута чвор представља одредишну тачку за неку грану би била најбоља дефиниција за улазни степен тог чвора док излазни степен чвора представља број грана у којима тај чвор представља полазну тачку.



Слика Усмерен граф

### Тежински графови

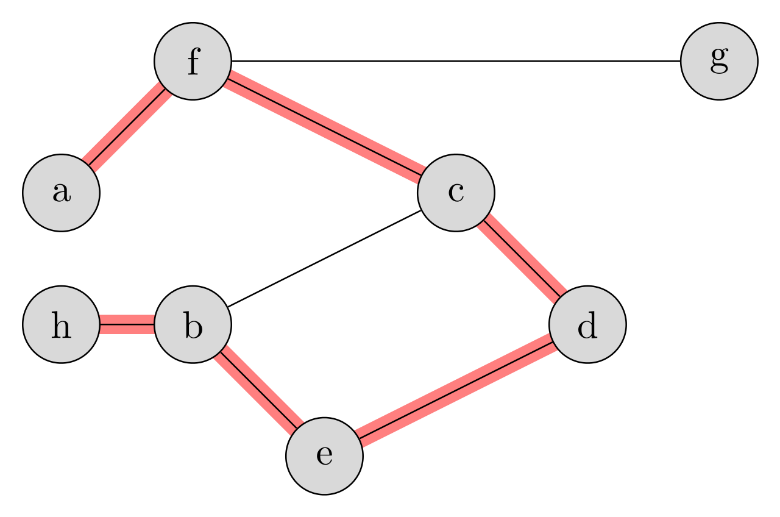
Графови код којих гране имају нумерисане или су означене било којом другом ознаком упоредивог типа представљају **тежинске графове** (слика 4). Овакви графови могу бити и усмерени и неусмерени и најчешће се ова врста графа примењује за проналажење најкраћег или најдужег пута у реалном свету.



Слика Тежински граф

## Проналажење најкраћег пута у графу

**Шетња** по графу представља кретање по међусобно повезаним чворовима где чворови могу да се понављају. Шетња у којој су почетни и крајњи чвор исти се назива **контура** или **циклус**. **Пут** у графу је шетња која не садржи циклусе тј. код кога се чворови не понављају (слика 5).



Слика Репрезентација пута у графу

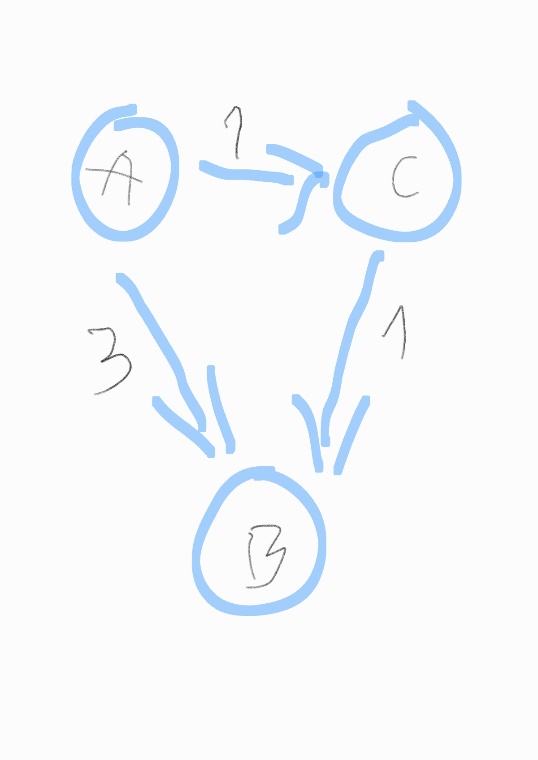
Да би разумели кључне функције у пројекту прво морамо да разумемо неке од алгоритама који се користе за проналажење најкраћег пута у графу, а то су Дајкстрин и Флојд Варшалов алгоритам претраге.

### Релаксација грана

Мењање дистанце која се чува као најкраћа између чворова А и B неком другом краћом дистанцом која се добија проналаском краћег пута преко средњег чвора на усмереном путу се назива **релаксација гране**.

Пример:

Ако посматрамо граф приказан на слици, најкраће растојање од чвора А до чвора B није једнак тежини гране AB већ збиру тежина грана АС и СB

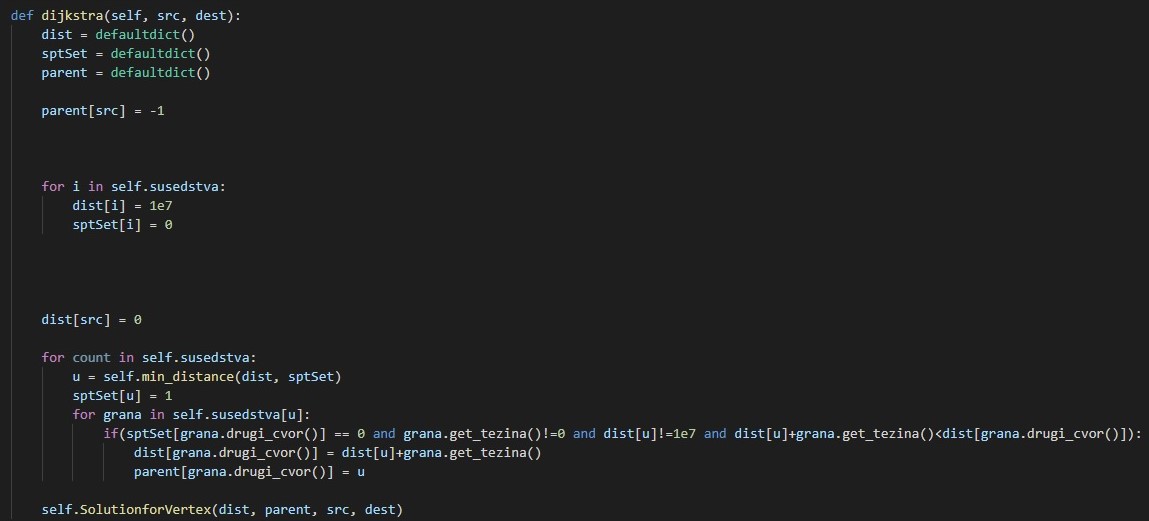


Слика Релаксација гране у графу

### Дајкстрин алгоритам претраге

**Дајкстрин алгоритам** претраге је после претраге у дубину и ширину вероватно најчувенији алгоритам везан за графове. Име је добио по холандском информатичару Едсгеру Дајкстри (1930-2002). Алгоритам се извршава над тежинским графовима код којих гране имају ненегативне вредности.

Алгоритам корак по корак:



Слика   
Дајкстрин алгоритам имплементиран у *Python*-u

Ако је број чворова у графу једнак V направићемо матрицу *dist* димензија V\*V која ће представљати раздаљине између чворова. Такође мораћемо да памтимо које чворове смо посетили, и који чвор је претходио посећеном чвору стога нам требају још две матрице. У овом случају смо користили мапе како би лакше проналазили потребне податке.

### Флојд – Варшалов алгоритам

Сврха алгоритма: Проналажење најкраћег пута између сваког пара чворова.

Повратна вредност: матрица растојања међу чворовима

Концепт алгоритма је да се релаксирају гране бирањем једног по једног чвора као средњег чвора на усмереном путу. Итерираћемо кроз граф чвор по чвор. Када се обрађује чвор V за сваки пар чворова (а, b) постоје два случаја:

1. Ако V није чвор који се налази на путу од a до b дистанца између та два чвора остаје иста
2. Ако се V налази на путу од а до b извршавамо релаксацију гранa

У примеру испод користићемо две матрице то јест хеш мапе dis и path. Прва ће представљати матрицу која садржи раздаљине од свих могућих парова чворова у графу. Уколико два чвора нису повезана и не постоји пут између њих вредност за тај пар ће отићи у бесконачно, а уколико су у питању два идентична чвора вредност ће бити 0.

Са друге стране матрица path помаже у памћењу чворова који се налазе на путу између два чвора А и Б.



Слика   
Флојд Варшалов алгоритам имплементиран у *Python*-у

# Програмски језици и технологије које ће се користити у практичном делу рада

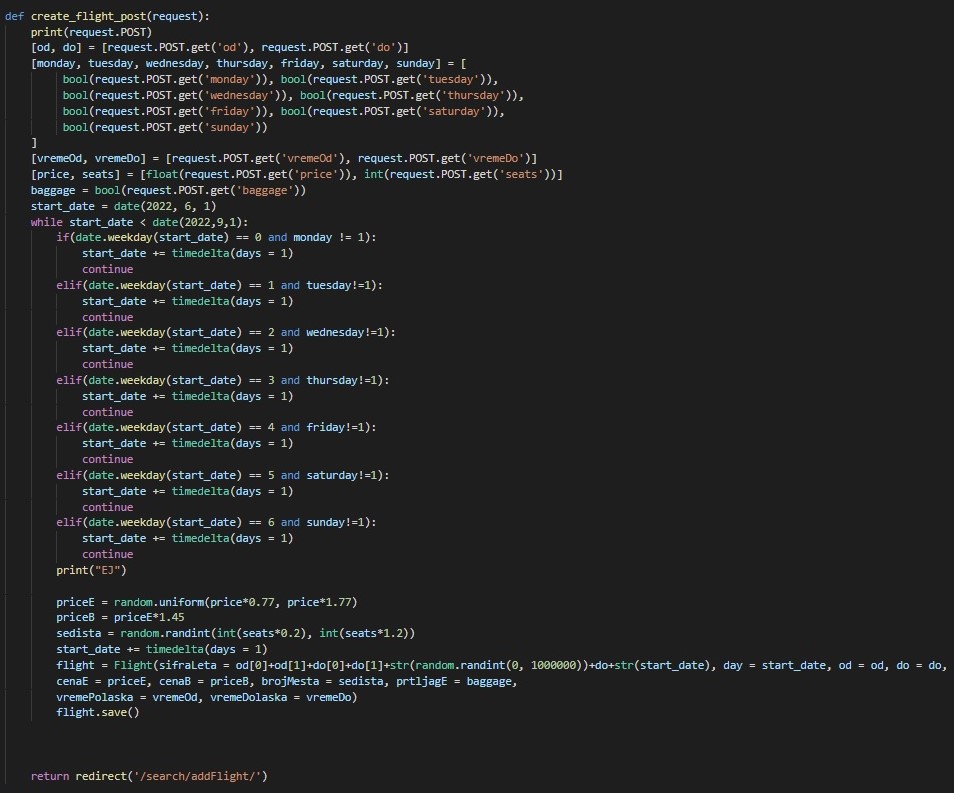
За алгоритамско-логички део апликације (backend) користићу програмски језик *Python*, за кориснички интерфејс употребићу стандардна три језика *HTML*, *CSS* и *Java Script*, док ћу за комуницирање између ова два дела апликације користити *Django* веб фрејмворк.

## Django

Django је једна модерна свеобухватна и широко применљиваOpen Source платформа која служи за креирање веб сајтова и других веб апликација. Настала је 2005. године од непрофитне програмерске организације. Ова технологија је од велике помоћи при комуникацији између сервера и корисничког интерфејса. Најбитнија функција ове технологије јесте примање и резоновање различитих захтева од стране корисника.

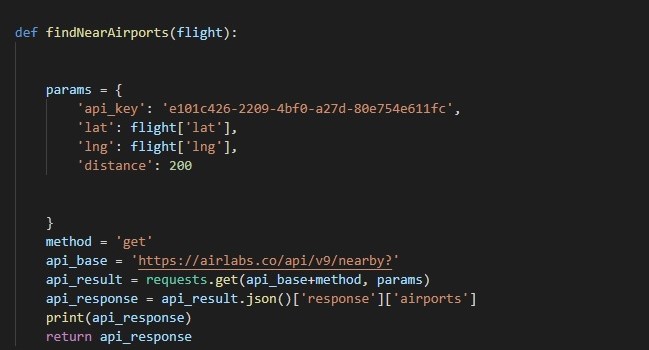
### Post и Get методе

**Post** метод се користи када корисник хоће да измени, дода и на крају сачува било какве промене на серверу. У практичном (слика 8) делу ћемо ову методу користити само при додавању нових авио линија у базу података.



Слика   
POST метода

Са друге стране **Get** метод се користи када је потребно прикупити податке са сервера. У пројектном делу ћемо га користити при претрази летова и аеродрома, само што ћемо при претрази летова купити податке са локалне базе, а при претрази аеродрома податке са API-а.



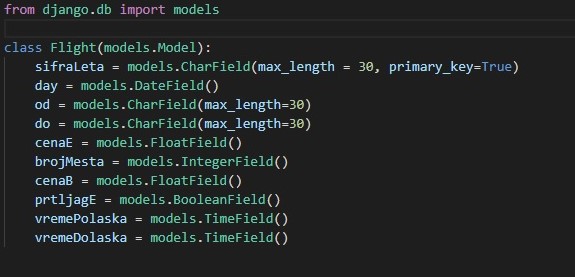
Слика   
Функција за проналажење аеродрома



Слика   
Проналажење летова у локалној бази

### Како са базама у Django

Свака *Django* апликација садржи датотеку под именом models.py у којој се чувају модели различитих објеката које желимо да чувамо. Свако поље у моделу ће се превести на *SQL* језик. На пример поље *sifraLeta* у моделу лета приказаном на слици 11 ће се превести у *VARCHAR*.



Слика   
Модел лета

# Контрола и претрага лета

У овом делу ћемо видети како графови и алгоритми који се користе за њих могу да се примене у пракси. Конкретније графове и алгоритме који се користе у раду са њима ћемо применити за претрагу и контролу летова.

## Контрола лета

Овај део практичног рада је назван тако јер симулира давање инструкција пилоту у случају квара или нежељених ситуација током летења. За почетак морамо да унесемо шифру лета (Слика 12) да би знали ко шаље поруку о квару или проблему.



Слика 13  
Control HTML



Слика 14  
Прослеђивање шифре лета пилот страници

Потом ћемо унешени код проследити пилот страници (Слика 13) којa једино што има јесте Error дугме које пилот притиска у случају квара (Слика 14)



Слика 15  
Pilot html

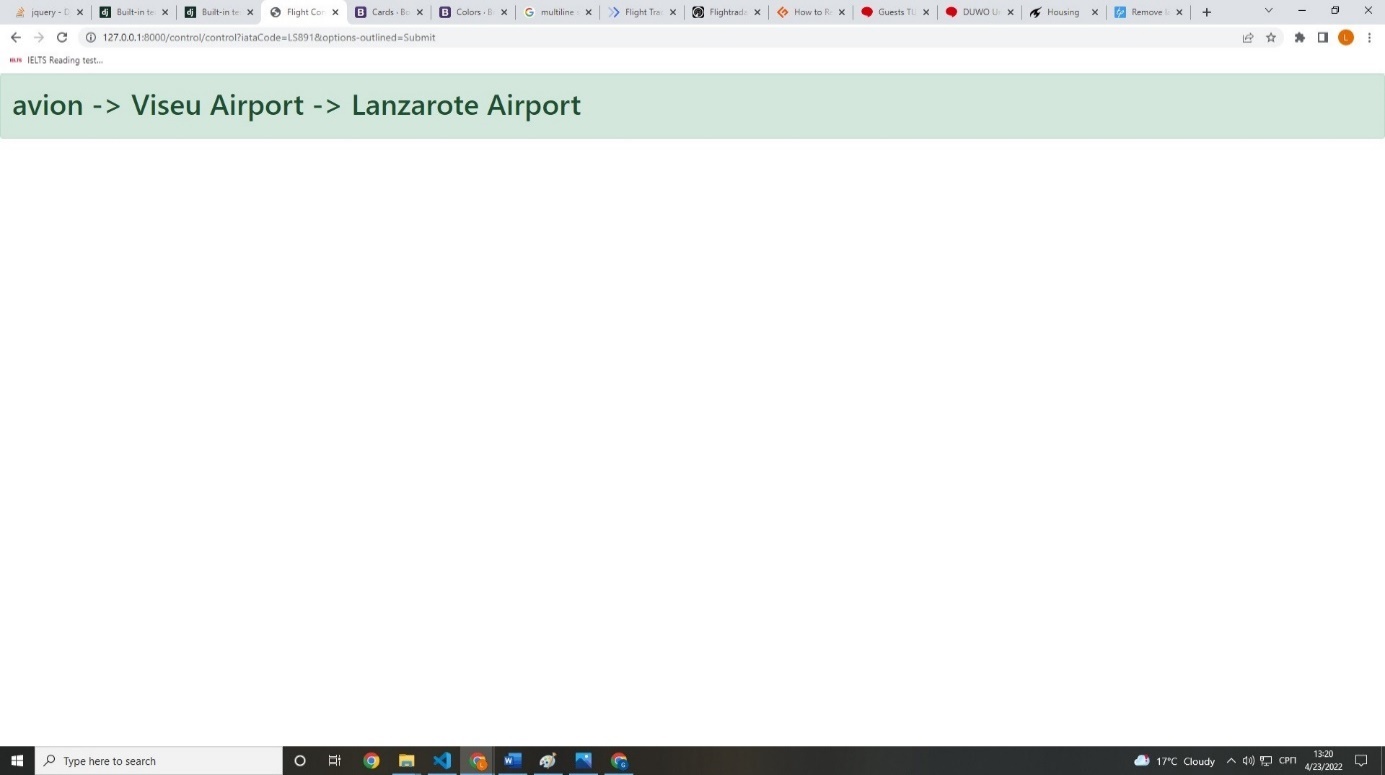
Када се кликне на приказано дугме дешавају се три ствари у следећем поретку:

1. Претражују се сви аеродроми који су највише 200 километара удаљени од тренутних тачака авиона
2. Потом се од кордината авиона и могућих прилазних тачака направи усмерени тежински граф који ће за параметре који одређују тежину путу имати раздаљину коју авион треба да пређе и колико је популаран аеродром на који авион мора привремено да слети.
3. На крају се покреће Дајкстрин алгоритам претраге који ће одредити решење то јест најбржи пут из тренутне ситуације.



Слика 16

На последњој страници (Слика 16) се исписује на који аеродром авион треба да слети.



Слика 17  
Путања

## Претрага лета

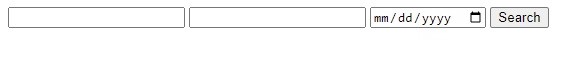
У овом делу ћемо за претрагу такође користити тежински граф само што ће у овом случају параметри бити цена, број преседања, број слободних места и пртљаг. Будући да корисник није у обавези да упише одакле и где жели да иде постоје три случаја:

1. Уколико је корисник уписао и место полетања и место слетања програм ће извршити претрагу, али ће приказати само понуду за коју мисли да је најоптималнија по споменутим параметрима. У овом случају ћемо користити Дајкстрин алгоритам претраге.
2. Са друге стране уколико је уписао само полазно место приказаће му се неколико добрих понуда за унети датум. Помоћу Дајкстриног алгоритма претраге пронаћи ће по једну понуду за сваку локацију за коју постоји рута.
3. И на крају, уколико остави оба поља празна програм ће приказати летове на основу Флојд-Варшаловог алгоритма. Овај алгоритам користимо баш зато што није ограничен на чвор од ког се креће већ приказује најкраће путеве за све могуће чворове-

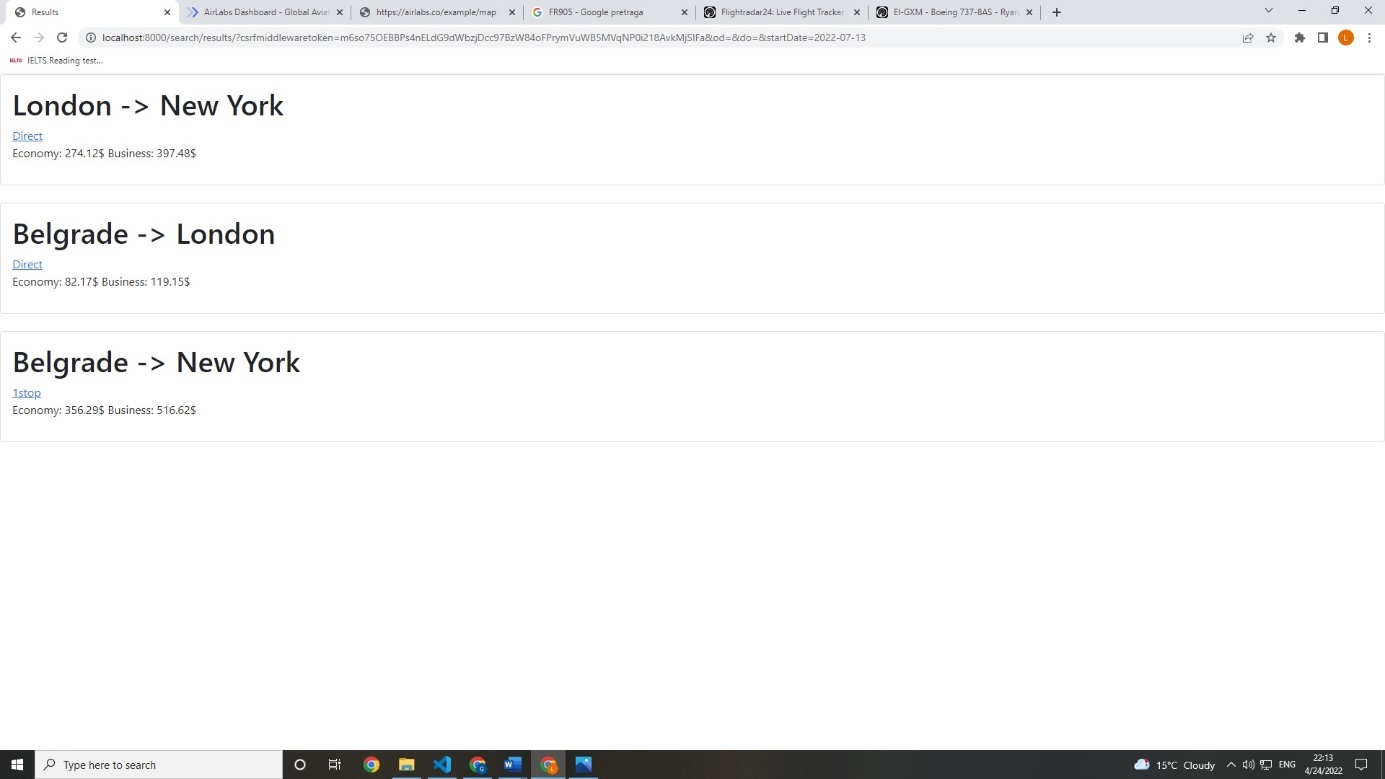


Слика   
*search()*

Сваки пут свака од позваних функција (*dijkstraForVertex()*, *dijkstra()*, *floyd\_warshall()*) (Слика 17) ће као повратну вредност вратити string написан у HTML форми који ће се приказати на страници са резултатима претраге. Уколико је променљива htmlText празна, то значи да програм није пронашао ниједан лет за унете параметре и стога ће се на страници и приказати одговарајућа порука.



Слика



Закључак

Текст...

# Прилог / Прилози

## Прилог 1

Овај део рада је предвиђен уколико има потребе да се уз рад приложе неке слике, скице, резултати истраживања и слично што није већ сасатвни део ранијих поглавља, него се из њих упућује на прилог. Уколико нема прилога, треба обрисати ову страницу.

# Садржај CD-а

Овај део рада се користи само ако се уз рад заиста прилаже CD, тада треба навести шта се налази на њему. У супротном, треба обрисати ову страницу.

# Литература

Пример навођења литературе (по азбучном реду):

1. Душанић, С. (2011). *Родне норме и ризична понашања младића у региону*. У  Д. Бранковић (ур.), часопис: *Култура и образовање – детерминанте друштвеног прогреса*, стр 263-285. Бањалука: Филозофски факултет.
2. Јанковић, Б. Милојевић, С. (2011).*Међународна полицијска сарадња у борби против насиља на фудбалским утакмицама*, Зборник радова, „Сузбијање криминала у оквиру међународне полицијске сарадње, Тара, стр. 149—161. - зборник радова
3. Станојчић, Ж., Поповић, Љ., (2010), *Граматика српског језика*, Завод за уџбенике, Београд. - књига

следи навођење литературе са интернта, на пример,

1. Word 2010 videos and tutorials, <https://support.office.com/en-us/article/Word-2010-videos-and-tutorials-cfa75118-e522-4ea5-963e-2b56d25fb9a5?ui=en-US&rs=en-US&ad=US> (јануар 2018)
2. Гимназија Јован Јовановић Змај Нови Сад, <http://jjzmaj.edu.rs> (јануар 2018)

Слике

[Слика 1 Граф 5](#_Toc101729841)

[Слика 2 Неусмерен граф 5](#_Toc101729842)

[Слика 3 Усмерен граф 6](#_Toc101729843)

[Слика 4 Тежински граф 6](#_Toc101729844)

[Слика 5 Репрезентација пута у графу 7](#_Toc101729845)

[Слика 6 Релаксација гране у графу 7](#_Toc101729846)

[Слика 7 Дајкстрин алгоритам имплементиран у *Python*-u 8](#_Toc101729847)

[Слика 8 Флојд Варшалов алгоритам имплементиран у *Python*-у 9](#_Toc101729848)

[Слика 9 POST метода 10](#_Toc101729849)

[Слика 10 Функција за проналажење аеродрома 11](#_Toc101729850)

[Слика 11 Проналажење летова у локалној бази 11](#_Toc101729851)

[Слика 12 Модел лета 11](#_Toc101729852)

[Слика 13 Control HTML 12](#_Toc101729853)

[Слика 14 Прослеђивање шифре лета пилот страници 12](#_Toc101729854)

[Слика 15 Pilot html 12](#_Toc101729855)

[Слика 16 13](#_Toc101729856)

[Слика 17 Путања 13](#_Toc101729857)

[Слика 18 *search()* 14](#_Toc101729858)

# БИОГРАФИЈА МАТУРАНТА



Лазар Половина рођен је 30, марта 2003, у Новом Саду.

Основно образовање стекао је 2018. године као ученик основне школе „Жарко Зрењанин“ у Новом Саду .

Нижу музичку школу за виолончело завршио је 2018. године у музичкој школи „Исидор Бајић“

Ученик је информатичког IV – 5 одељења Природно математичке Гимназије „Јован Јовановић Змај“ у Новом Саду.

Датум предаје матурског рада: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Комисија:

Председник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Испитивач \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члан \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Коментар:

Датум одбране: \_\_\_\_\_\_\_\_\_\_\_\_\_ Оцена\_\_\_\_\_\_\_\_\_\_ (\_\_\_)