

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**  
**СТОПАНСКИ ФАКУЛТЕТ**  
**Катедра "Мениджмънт и бизнес информационни системи"**

---

## **КУРСОВА ЗАДАЧА**

по дисциплината "Мениджмънт на облачни организации и центрове за данни"

от бакалавърската специалност „Мениджмънт и бизнес информационни системи“

на тема:

„Системи за контрол на версиите. GitHub“

Разработили:	Ръководители:
Даниел Димитров ф.н. 291221007	Гл.ас. д-р Р. Костев
Десислава Борисова ф.н. 291221047	
Реджеп Карамолла ф.н. 291221052	

София, 2024

# СЪДЪРЖАНИЕ

<b>СЪДЪРЖАНИЕ.....</b>	<b>2</b>
<b>ПЪРВА ГЛАВА</b>	
<b>ВЪВЕДЕНИЕ В СИСТЕМИ ЗА КОНТРОЛ НА ВЕРСИИТЕ.....</b>	<b>3</b>
1.1. Системи за контрол на версиите.....	3
1.1.1. Въведение.....	3
1.1.2. Локални системи за контрол на версиите.....	3
1.1.3. Централизиран системи за контрол на версиите.....	4
1.1.4. Разпределени системи за контрол на версиите.....	5
1.1.5. Git.....	7
1.2. GitHub.....	8
<b>ВТОРА ГЛАВА</b>	
<b>ПРОЕКТНА ЧАСТ.....</b>	<b>10</b>
2.1. Идея на проекта.....	10
2.2. Създаване на хранилище за приложението.....	10
2.3. Използване на Git & GitHub за реализиране на проекта.....	13
2.4. Описание на кода на проекта.....	16
<b>ИЗПОЛЗВАНА ЛИТЕРАТУРА.....</b>	<b>24</b>

# **ПЪРВА ГЛАВА**

## **ВЪВЕДЕНИЕ В СИСТЕМИ ЗА КОНТРОЛ НА ВЕРСИИТЕ**

### **1.1. Системи за контрол на версиите**

#### **1.1.1. Въведение**

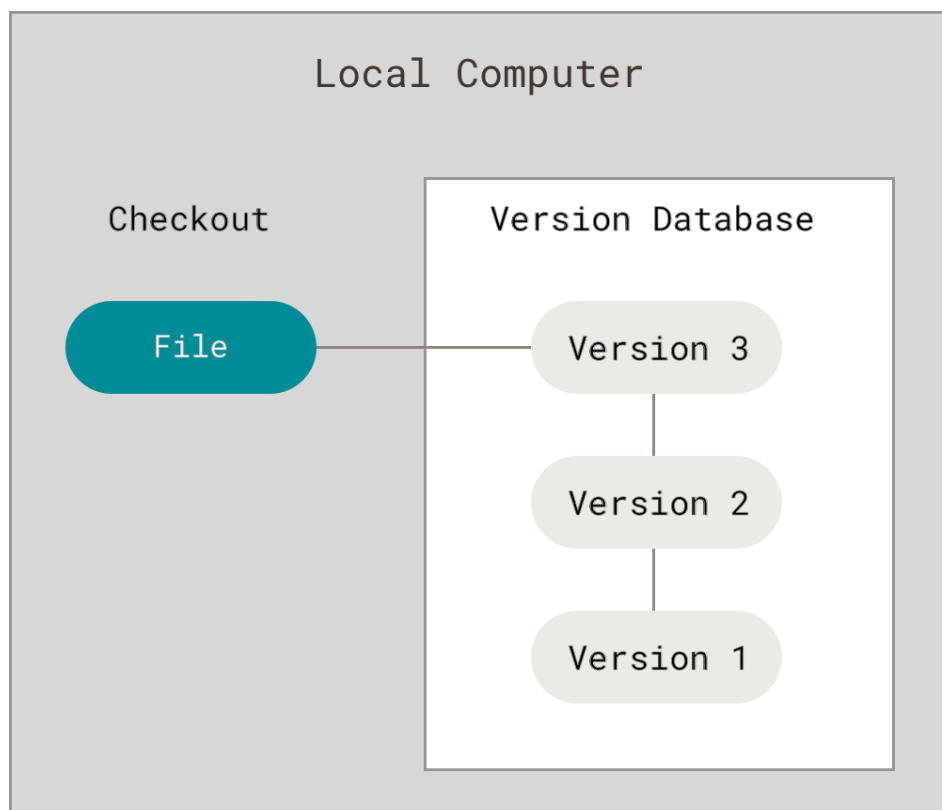
Системата за контрол на версиите е средство за проследяване и запис на промените по файл (или множество файлове) във времето, така че да можете да възстановите произволна версия на файла/файловете във всеки един момент по-късно. За примерите в тази книга, вие ще използвате софтуерен код под формата на файлове, чиито версии ще бъдат контролирани, макар в действителност да можете да правите това с произволен тип файл във вашия компютър. [1]

Системите за контрол на версиите се класифицират в следните основни видове: Локални системи за контрол на версиите, централизирани системи за контрол на версиите, децентрализирани (разпределени) системи за контрол на версиите.

#### **1.1.2. Локални системи за контрол на версиите**

Чрез локалните системи за контрол на версиите просто се копират файловете в друга директория (ако са достатъчно изобретателни - посочвайки дата и час в името ѝ). Този начин е често използван защото е много прост, но и също толкова податлив на грешки. Лесно е да се забрави в коя директория е потребителят в момента и има риск неволно да бъде записан грешен файл или пък да бъде копиран, където не трябва.

За да се справят с това неудобство, програмистите са разработили локални VCS с прости бази данни, които съхраняваха промените по файловете, които се проследяват. [1]



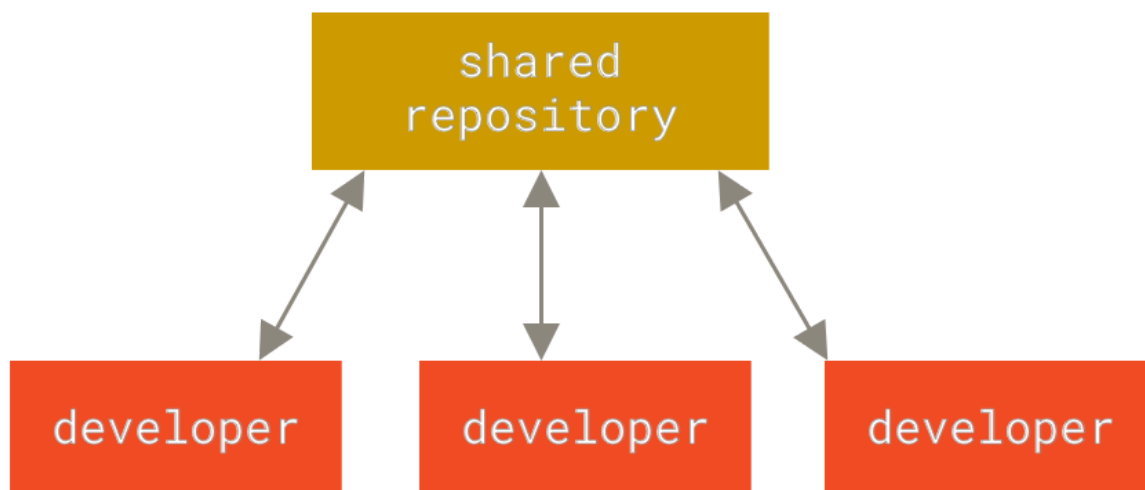
Фиг. 1.1. Локални системи за контрол на версиите [1]

Една от най-популярните подобни системи се нарича RCS и все още се разпространява с много компютри и до днес. RCS работи съхранявайки множество от пачове (разликите във файловете) в специален формат на диска и може да възстанови състоянието на файла към произволен момент добавяйки всички тези пачове. [1]

### 1.1.3. Централизираните системи за контрол на версиите

Централизираните системи за данни помагат колаборацията между екипи и софтуерни разработчици.

Такива централизираните Version Control системи (CVCS) - CVS, Subversion, Preforce, използват един сървър, съдържащ всички контролирани файлове и множество от клиенти, които издърпват файловете от това централно място. В продължение на много години това беше стандарт за контрол на версиите. [1]



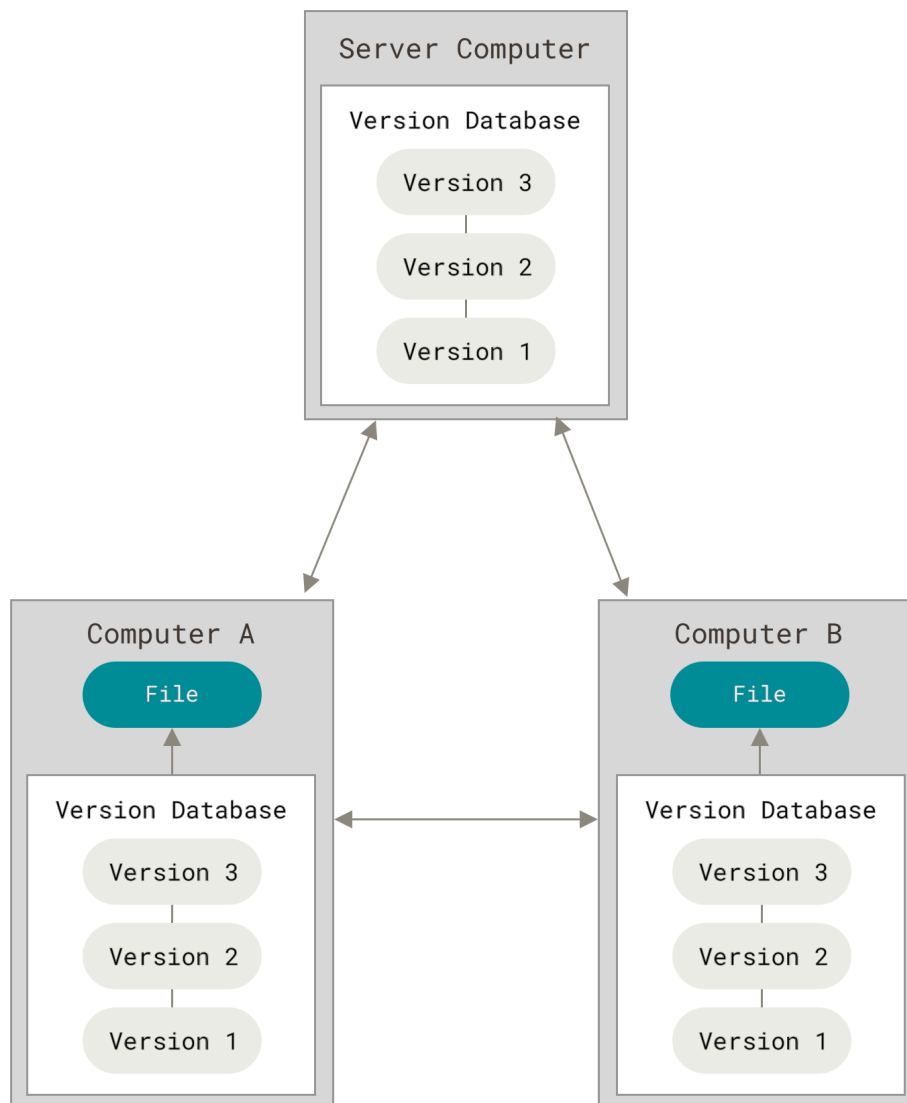
Фиг. 1.2. Централизиран системи за контрол на версиите [1]

Този подход предлага много предимства спрямо локалните VCS. Всеки член на екипа има добра представа за работата на останалите, а администраторите разполагат с детайлен контрол върху правата на достъп, като администрирането на CVCS е по-лесно от това на локалните бази данни. [1]

Въпреки това, подходът има сериозни недостатъци. Основният е, че всички данни се съхраняват на едно място и зависят от надеждността на сървъра. Ако сървърът спре за известно време, никой не може да прави промени през този период. В случай на повреда на сървъра и липса на архиви, съществува риск от загуба на цялата история на проекта, освен ако някой няма локален snapshot. Същият риск съществува и при локалните VCS, когато данните са централизирани. [1]

#### 1.1.4. Разпределени системи за контрол на версиите

В DVCS (каквито са Git, Mercurial, Bazaar или Darcs), клиентите не просто изтеглят последния snapshot на файловете - те изцяло клонират цялото хранилище, включително пълната му история. По този начин, ако сървърът загине, хранилището на даден проект може да се възстанови от локалното копие на всеки клиент. Всяко копие по същество е пълен архив на всички данни. [1]



Фиг. 1.3. Разпределени системи за контрол на версиите [1]

В допълнение, много от тези системи се справят доста добре със задачата да могат да обслужват няколко отдалечени хранилища, така че софтуерните разработчици да си сътрудничат едновременно, в рамките на един и същи проект. Това позволява да създаването на няколко различни работни потока, което е невъзможно в централизираните системи като йерархични модели.

### 1.1.5. Git

Git е създаден като резултат от комбинация от креативна разрушителност и остри спорове. Поддържането на Linux ядрото, което представлява open-source софтуерен проект с голям мащаб, първоначално е било осъществявано чрез изпращане на пачове и архивирани файлове в периода 1991-2002 г. През 2002 г. проектът Linux kernel започва да използва патентована децентрализирана система за контрол на версиите, наречена BitKeeper. [2]

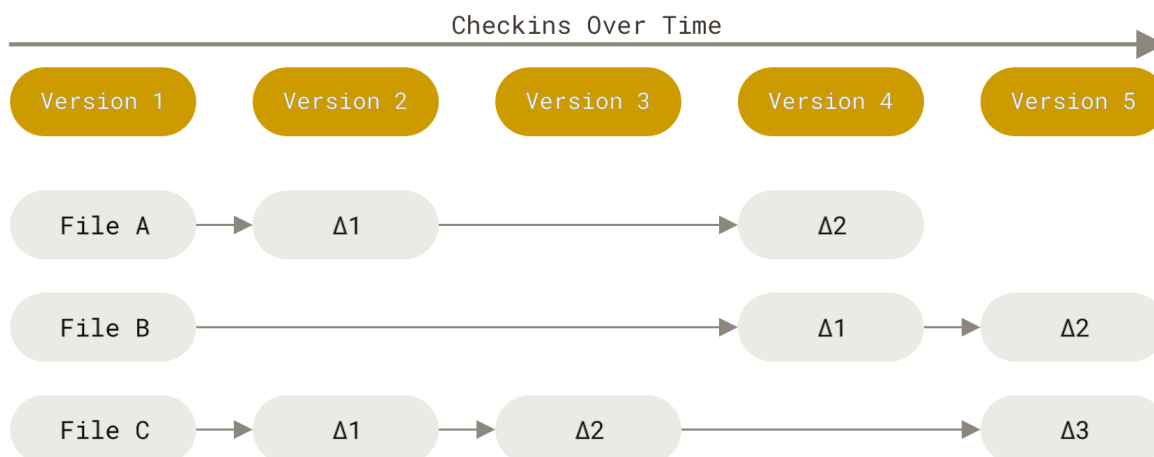
През 2005 г., след разпадането на отношенията между общността на разработчиците на Linux ядрото и компанията, която поддържа BitKeeper, достъпът до инструментариума на BitKeeper е спрял. Това събитие провокира създателя на Linux Линус Торвалдс, заедно с общността от разработчици, да разработи нов инструмент за контрол на версиите. За основа са използвани част от знанията, придобити по време на работата с BitKeeper. При създаването на новата система са поставени конкретни цели и изисквания, включително:

- Висока скорост
- Опростен дизайн
- Мощна поддръжка за нелинейна разработка, позволяваща хиляди паралелни клонове код
- Напълно разпределена работа
- Ефективно обслужване на големи проекти, като Linux ядрото, по отношение на скорост и обем на данни

От създаването си през 2005 г. Git е еволюирала и усъвършенствана, за да бъде лесна за използване, като същевременно поддържа първоначалните си цели. Днес Git се отличава с висока скорост, ефективност при работа с големи проекти и мощна система за управление на клонове, която позволява нелинейна разработка.

Основната разлика между Git и другите системи за контрол на версиите е начинът, по който Git третира данните. Концептуално, повечето други системи записват информацията като списък от файлово-базирани промени. Тези системи (CVS, Subversion, Perforce, Bazaar) виждат информацията, която съхраняват като колекция от

файлове и промените направени във файловете във времето (известно още като delta-based version control). [3]



Фиг. 1.4. Съхраняване на данните като списък от промени в базовата версия на всеки от файловете [3]

## 1.2.GitHub

GitHub е облачна платформа, в която може да се съхранява, споделя и съвместно разработва код.

Съхраняването на кода в „репозитория“ в GitHub позволява следното:

- Показване и споделяне на работата.
- Проследяване и управление на промените в кода във времето.
- Позволяване на други да преглеждат кода и да дават предложения за подобрения.
- Съвместна работа по общ проект, без риск промените на един участник да повлияят на работата на останалите, преди те да бъдат интегрирани.
- Съвместната работа, която е основна характеристика на GitHub, се осъществява благодарение на open-source софтуера Git, върху който е изградена платформата GitHub.

GitHub не само улеснява съхранението и споделянето на код, но и предоставя мощни инструменти за управление на проекти и екипи. Чрез функции като pull requests, issues и GitHub Actions, разработчиците могат лесно да организират задачите си, да автоматизират работните процеси и да осигуряват високо качество на кода.



Платформата позволява контрол на версиите и история на промените, което помага на екипите да проследяват напредъка на проекта, да преглеждат предишни версии на кода и да връщат по-ранни версии при нужда. GitHub също така поддържа интеграции с множество инструменти и услуги, което улеснява разработката, тестването и внедряването на софтуер.

Заедно с това, GitHub е и социална платформа за разработчици, където професионалисти и ентусиасти могат да се свързват, да следят работата на други разработчици, да участват в open-source проекти и да подобряват уменията си чрез сътрудничество и споделяне на знания.



*Фиг. 1.5. GitHub logo [4]*

# ВТОРА ГЛАВА

## ПРОЕКТНА ЧАСТ

### 2.1. Идея на проекта

Ticket-App е уеб приложение, което сканира (web scraping) уебсайтове за билети, за да предостави информация за предстоящи събития в различни региони на България. Приложението използва Flask като web framework (уеб фреймуърк) и включва техники за уеб скрапинг за събиране на данни за събития от различни източници.

За целите на курсовата задача бе разделена на следните етапи:

- **Първи етап** - внедряване на код за визуализиране на картата на България и разделението ѝ на административни области. За целта е използван **Bulgaria-geocoding** [5] проект в гитхъб и следния файл **provinces.geojson**
- **Втори етап** - създаване на логика за скрейпване на различни платформи за продажба на билети. За този етап бяха избрани платформите **grabo.bg** и **bilet.bg**, като такива позволяващи web scraping
- **Трети етап** - представяне на резултата от web scraping при избор на област. За реализация на този етап бяха свързани резултатите от първите два етапа и бе реализирана фронтенд частта на приложението.

### 2.2. Създаване на хранилище за приложението

С цел по - лесна колаборация между разработчиците на проекта, бе създадено хранилище в платформата GitHub. Хранилището се намира на адрес:

<https://github.com/dhdimitrov/Ticket-App/>

И бе създадено по следния начин: от **github.com** бе избран в горния десен ъгъл бутон **New**, който препраща към страница за създаване на хранилището:

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

Required fields are marked with an asterisk (\*).

**Owner \*** dhdimitrov ▾ / **Repository name \*** Tickets-App

✔ Tickets-App is available.

Great repository names are short and memorable. Need inspiration? How about [effective-octo-guide](#) ?

**Description (optional)**

---

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: None ▾

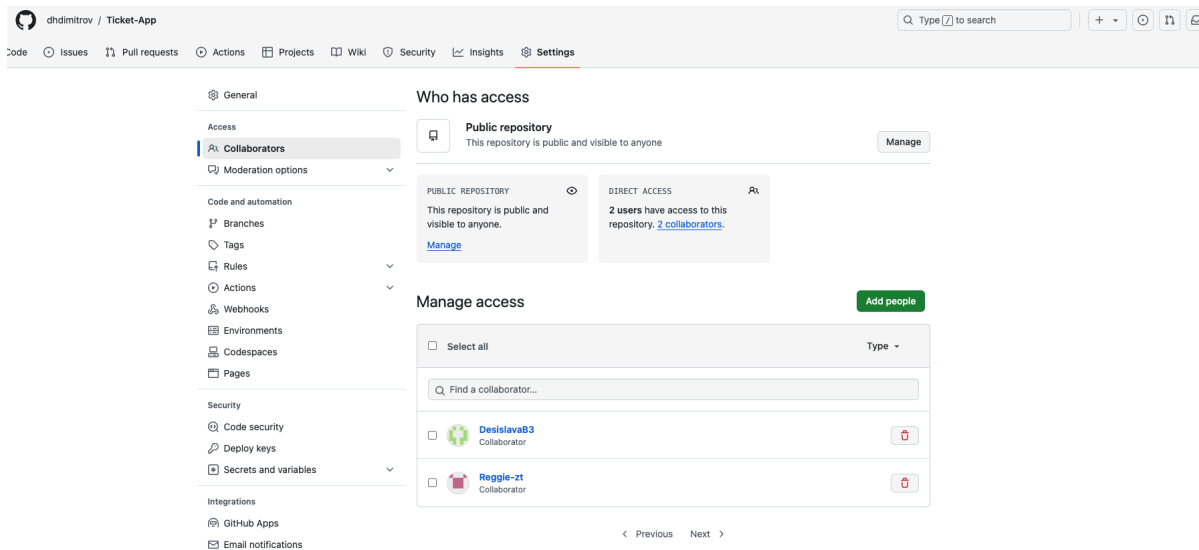
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

---

ⓘ You are creating a public repository in your personal account. Sp...

*Фиг. 2.1. Създаване на хранилище в GitHub*

Членовете на екипа са добавени като Collaborators от настройките на хранилището:



Фиг. 2.2. Настройки на GitHub хранилище

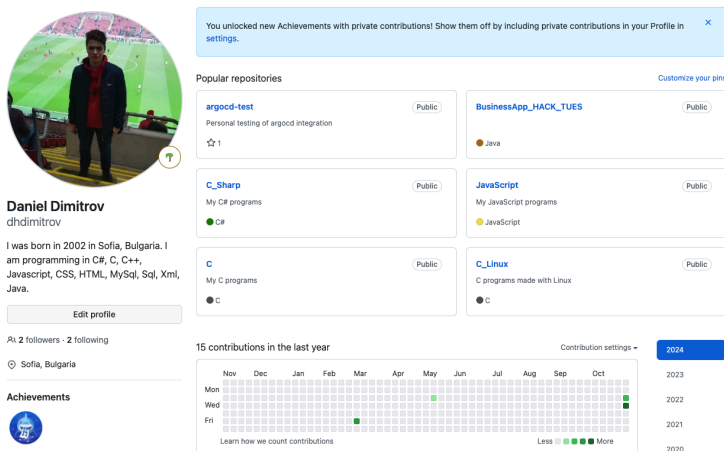
Членовете на екипа са както следва:

**DesislavaB3:** <https://github.com/DesislavaB3>



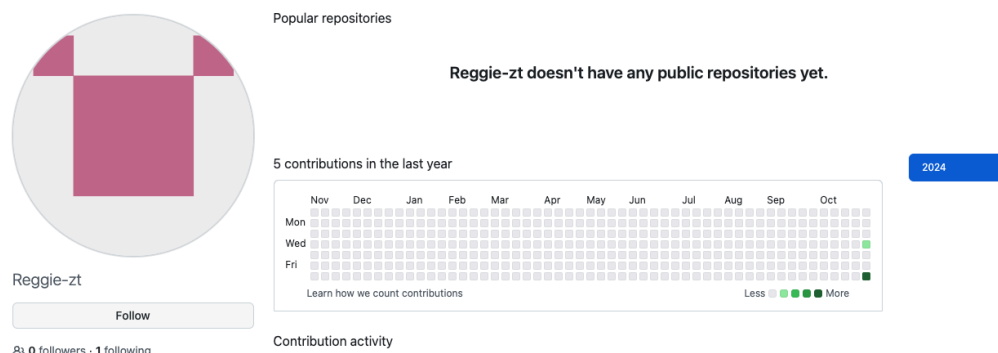
Фиг. 2.3. Профил на Десислава Борисова

**dhdimitrov:** <https://github.com/dhdimitrov>



Фиг. 2.4. Профил на Даниел Димитров

Reggie-zt: <https://github.com/Reggie-zt>



Фиг. 2.5. Профил на Реджеп Карамолла

## 2.3. Използване на Git & GitHub за реализиране на проекта

С цел лесна колаборация между всички членове на екипа, проекта бе разделен на отделни branches (клонове), като в края, всеки от тях бе обединен в основния branch - **main**. За тази цел бяха реализирани няколко Pull Requests - заявка за сливане:

Branches New branch

Overview Yours Active Stale All

Search branches...

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
main	2 hours ago				Default

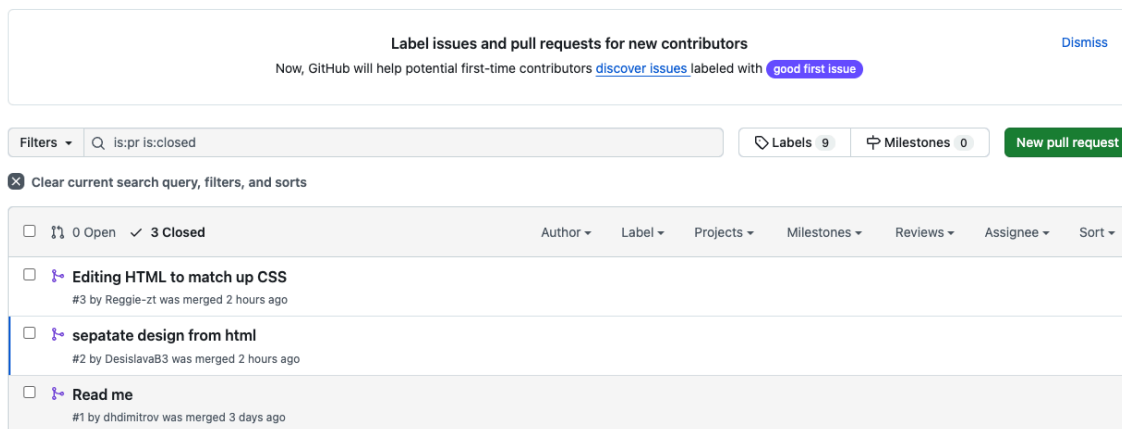
Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
Bulgaria-Map	3 days ago		10	0	#1

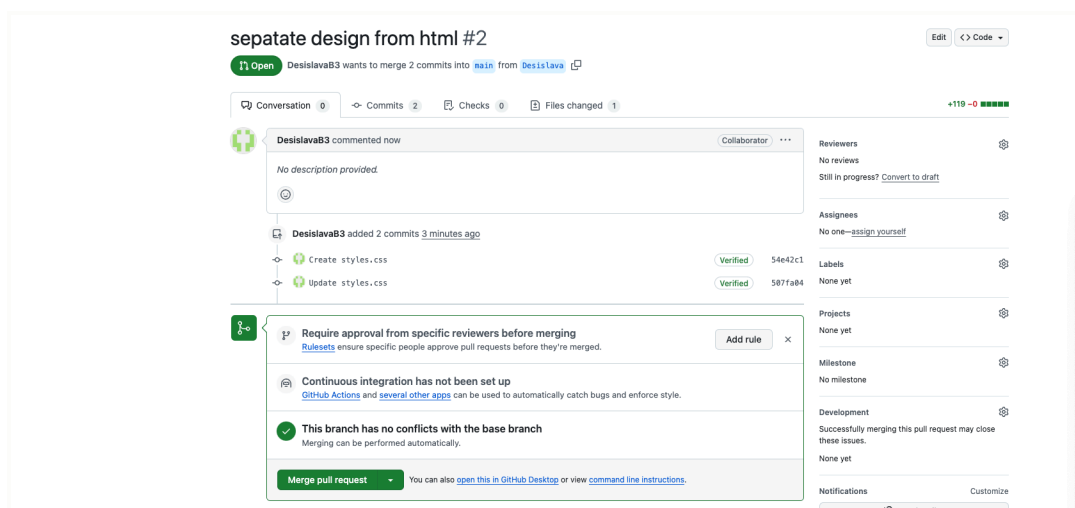
Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
Documents	2 hours ago		0	1	
Redzhep	2 hours ago		1	0	#3
Desislava	2 hours ago		4	0	#2
Bulgaria-Map	3 days ago		10	0	#1

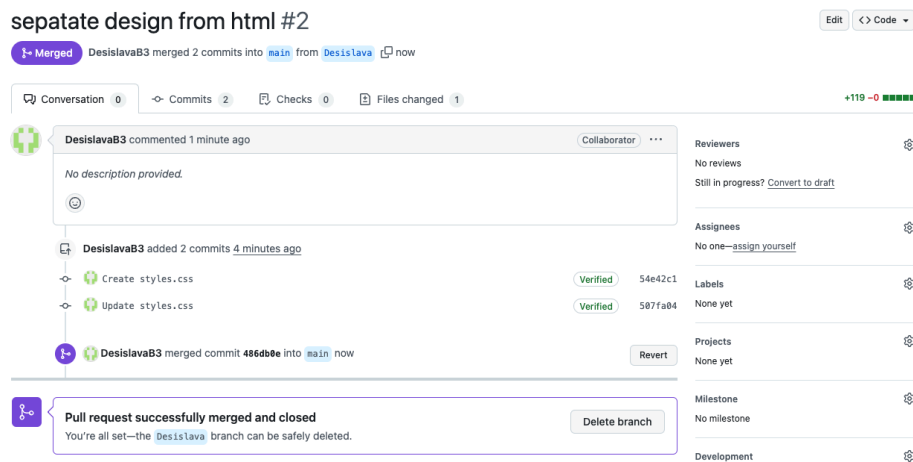
Фиг. 2.6. Клонове (branches) на проекта



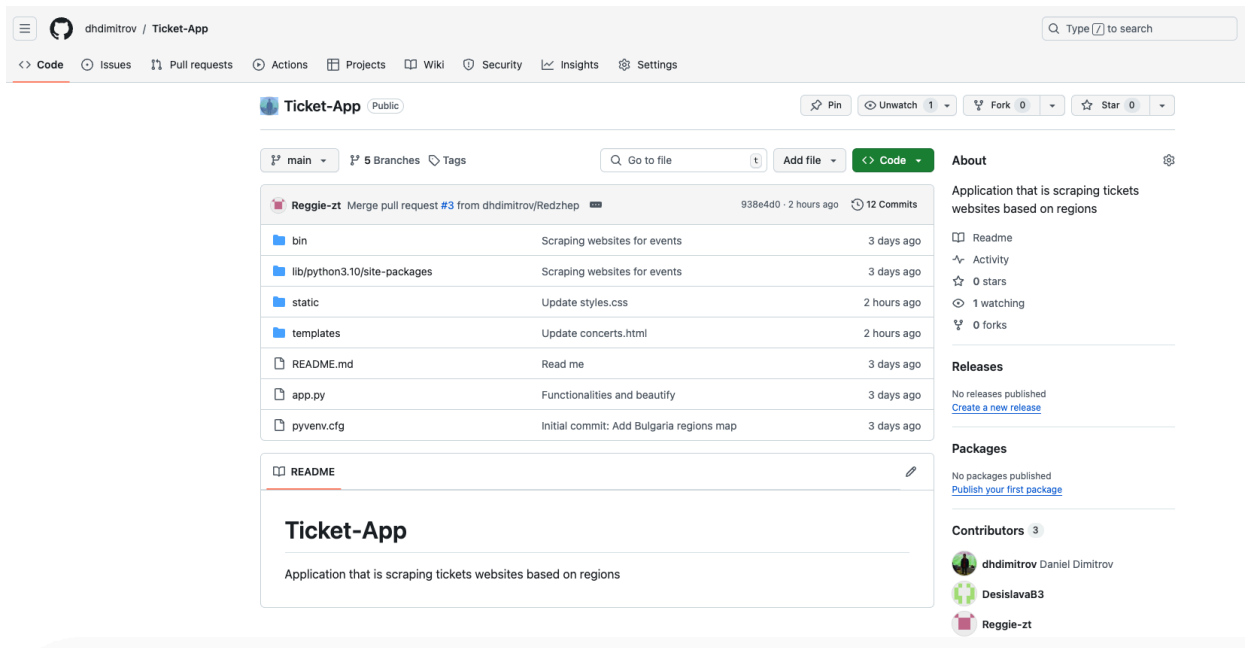
Фиг. 2.7. Част от заявките за сливане (pull requests)



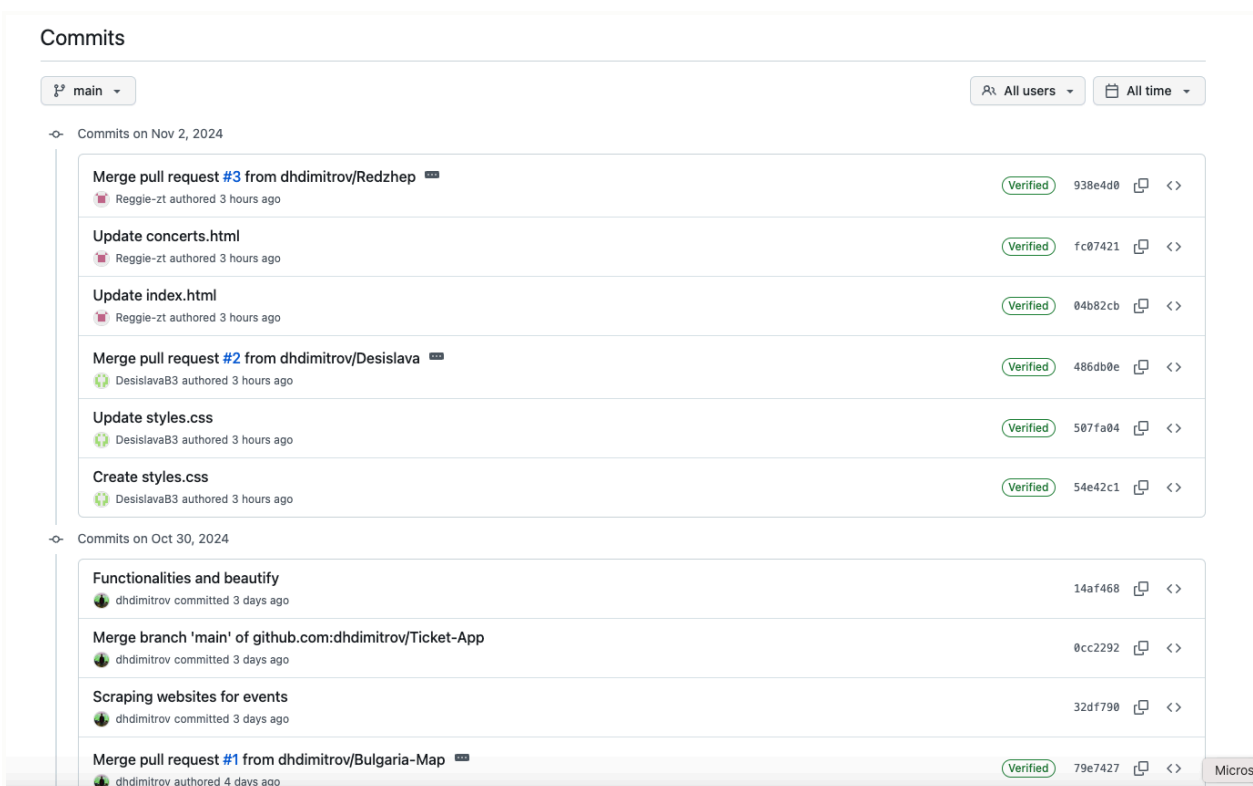
Фиг. 2.8. Примерна заявка (pull request)



Фиг. 2.9. Успешно преминало сливане (merge)



Фиг. 2.10. Краен изглед на проекта като разделение по файлове



Фиг. 2.11. История на комитите

## 2.4. Описание на кода на проекта

Както бе описано в т. 2.1., проекта е реализиран, чрез програмния Python, web framework Flask, JavaScript за визуализиране на картата на България, JSON за геометричните данни на регионите, както и HTML и CSS за визуализация на страниците (Front-End).

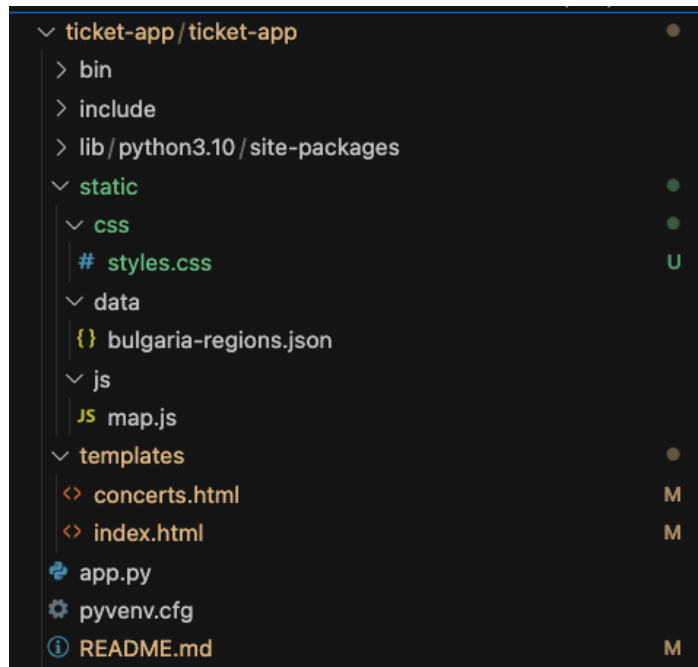
Основни компоненти:

- **app.py**: Съдържа маршрутите на Flask и основната логика на приложението. Реализира функции за уеб scraping на [bilet.bg](http://bilet.bg) и [grabo.bg](http://grabo.bg). Обработва заявки за различни региони и визуализира съответните шаблони.

Ключови маршрути:

- **/**: Показва началната страница с интерактивната карта.
- **/concerts/**: Извлича и показва събития за избрания регион.
- **index.html**: Показва интерактивната карта на България. Осигурява навигационно меню за бърз достъп до различни региони.
- **concerts.html**: Показва извлечената информация за събития в табличен формат. Включва детайли за събития като изображение, име, локация и дата.
- **map.js**: Рендира интерактивната карта на България с помощта на D3.js. Позволява на потребителите да кликат върху региони, за да видят събития.





Фиг. 2.12. Файлова структура на приложението

На Фиг. 2.12. е представена на `app.py`. Той започва с внедряване на конкретни библиотеки: **Flask**, **bs4**, **deep\_translator**, **re**.

На база `http` заявка са представени и функциите **index** и **scrape\_concerts**, който се използват рутиране към `html` страниците и събирането на данни:

```
ticket-app > ticket-app > app.py > biletbg
You, 3 days ago | 1 author (You)
1 from flask import Flask, render_template, jsonify
2 import requests
3 from bs4 import BeautifulSoup
4 from deep_translator import GoogleTranslator
5 import re
6
7 app = Flask(__name__)
8
9 @app.route('/')
10 def index():
11     return render_template('index.html')
12
13 @app.route('/concerts/<region>')
14 def scrape_concerts(region):
15     concerts = []
16     if region == 'София' or region == "София област":
17         concerts = biletbg()
18         concerts += grabo(region)
19     else:
20         concerts = grabo(region)
21
22     return render_template('concerts.html', concerts=concerts, region=region)
```

Фиг. 2.13. Начало на файл `app.py`

На Фиг. 2.13. и 2.14. са представени функции за web scraping на сайтовете билет.bg и grabo.bg. Билет бг се използва за събитията в София и София област, докато при Грабо бг заявките са базирани на региона на България:

```
24 def билетbg():
25     url = f"https://bilet.bg/bg/calendar"
26     response = requests.get(url)
27     response.raise_for_status()
28
29     soup = BeautifulSoup(response.text, 'html.parser')
30
31     concerts = []
32     for event in soup.find_all('div', class_='upcoming-events-box'):
33         image_div = event.find('div', class_='img-top')
34         image = extract_image_url(image_div)
35         name = event.find('h5').text.strip()
36         place = event.find(class_='upcoming-events-place').text.strip()
37         date = event.find('span').text.strip()
38         concerts.append({"image": image, "name": name, "place": place, "date": date})
39     return concerts
```

Фиг. 2.14. Скрейпване на данни от билет.bg

```
41 def grabo(region):
42     city = translate_city(region).lower().replace(" ", "-")
43     url = f"https://grabo.bg/events/{city}/popular#s"
44     response = requests.get(url)
45     concerts = []
46     if response.status_code == 200:
47         soup = BeautifulSoup(response.text, 'html.parser')
48
49         events = soup.find_all('div', class_='e_deal')
50
51         for event in events:
52             image_div = event.find('div', class_='ed_image')
53             title_div = event.find('div', class_='ed_event_title')
54             place_div = event.find('div', class_='ed_venue')
55             date_div = event.find('div', class_='ed_date')
56
57             if title_div:
58                 image = extract_image_url(image_div)
59                 title = title_div.text.strip()
60                 place = place_div.text.strip() if place_div else ""
61                 date = date_div.text.strip() if date_div else ""
62                 concerts.append({"image": image, "name": title, "place": place, "date": date})
63     return concerts
```

Фиг. 2.15. Скрейпване на данни от grabo.bg

За картата на България е използван следния JSON файл, който представя всички 28 административни области, чрез координати. Тази карта се визуализира посредством JavaScript код:

```

ticket-app > ticket-app > static > data > {} bulgaria-regions.json > {} features > {} 1 > {} properties > {} nuts3
You, 3 days ago | 1 author (You)
1  {"type": "FeatureCollection", "features": [{"type": "Feature", "properties": {"nuts3": "Благоевград"}, "geometry": {"type": "Polygon", "coordinates":
2  [{"type": "Feature", "properties": {"nuts3": "Ловеч"}, "geometry": {"type": "Polygon", "coordinates": [[24.904, 43.289], [24.931, 43.306], [24.977,
3  [{"type": "Feature", "properties": {"nuts3": "Добрич"}, "geometry": {"type": "Polygon", "coordinates": [[27.504, 43.84], [27.504, 43.841], [27.502,
4  [{"type": "Feature", "properties": {"nuts3": "Сливен"}, "geometry": {"type": "Polygon", "coordinates": [[26.667, 42.759], [26.668, 42.745], [26.687
5  [{"type": "Feature", "properties": {"nuts3": "Кърджали"}, "geometry": {"type": "Polygon", "coordinates": [[25.763, 41.325], [25.718, 41.317], [25.7
6  [{"type": "Feature", "properties": {"nuts3": "Варна"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[27.154, 43.031], [27.183, 43.046], [2
7  [{"type": "Feature", "properties": {"nuts3": "Габрово"}, "geometry": {"type": "Polygon", "coordinates": [[24.995, 43.194], [24.994, 43.193], [25.01
8  [{"type": "Feature", "properties": {"nuts3": "Бургас"}, "geometry": {"type": "Polygon", "coordinates": [[27.667, 42.639], [27.66, 42.639], [27.654,
9  [{"type": "Feature", "properties": {"nuts3": "Пазарджик"}, "geometry": {"type": "Polygon", "coordinates": [[24.3, 41.744], [24.298, 41.741], [24.29
10 [{"type": "Feature", "properties": {"nuts3": "Видин"}, "geometry": {"type": "Polygon", "coordinates": [[22.997, 43.807], [23.004, 43.792], [23.045,
11 [{"type": "Feature", "properties": {"nuts3": "Смолян"}, "geometry": {"type": "Polygon", "coordinates": [[25.011, 41.373], [24.949, 41.387], [24.917
12 [{"type": "Feature", "properties": {"nuts3": "Търговище"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[26.546, 43.345], [26.545, 43.348
13 [{"type": "Feature", "properties": {"nuts3": "София област"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[23.414, 42.464], [23.425, 42.
14 [{"type": "Feature", "properties": {"nuts3": "Велико Търново"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[26.068, 43.309], [26.068, 4
15 [{"type": "Feature", "properties": {"nuts3": "Силистра"}, "geometry": {"type": "Polygon", "coordinates": [[27.351, 43.807], [27.32, 43.808], [27.32
16 [{"type": "Feature", "properties": {"nuts3": "Стара загора"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[25.154, 42.506], [25.142, 42.
17 [{"type": "Feature", "properties": {"nuts3": "Хасково"}, "geometry": {"type": "Polygon", "coordinates": [[25.285, 41.812], [25.276, 41.821], [25.27
18 [{"type": "Feature", "properties": {"nuts3": "Шумен"}, "geometry": {"type": "Polygon", "coordinates": [[26.591, 42.991], [26.608, 43.006], [26.609,
19 [{"type": "Feature", "properties": {"nuts3": "Ямбол"}, "geometry": {"type": "Polygon", "coordinates": [[26.917, 42.342], [26.922, 42.294], [26.968,
20 [{"type": "Feature", "properties": {"nuts3": "София"}, "geometry": {"type": "Polygon", "coordinates": [[23.338, 42.865], [23.363, 42.852], [23.364,
21 [{"type": "Feature", "properties": {"nuts3": "Враца"}, "geometry": {"type": "Polygon", "coordinates": [[23.451, 43.239], [23.428, 43.251], [23.413,
22 [{"type": "Feature", "properties": {"nuts3": "Пловдив"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[24.72, 42.715], [24.755, 42.709], [
23 [{"type": "Feature", "properties": {"nuts3": "Перник"}, "geometry": {"type": "Polygon", "coordinates": [[22.915, 42.789], [22.918, 42.79], [22.93, 4
24 [{"type": "Feature", "properties": {"nuts3": "Плевен"}, "geometry": {"type": "Polygon", "coordinates": [[25.139, 43.277], [25.115, 43.269], [25.107
25 [{"type": "Feature", "properties": {"nuts3": "Монтана"}, "geometry": {"type": "Polygon", "coordinates": [[23.465, 43.512], [23.453, 43.5], [23.427,
26 [{"type": "Feature", "properties": {"nuts3": "Кюстендил"}, "geometry": {"type": "Polygon", "coordinates": [[22.976, 42.389], [22.976, 42.388], [22.
27 [{"type": "Feature", "properties": {"nuts3": "Русе"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[25.62, 43.671], [25.667, 43.688], [25.
28 [{"type": "Feature", "properties": {"nuts3": "Разград"}, "geometry": {"type": "Polygon", "coordinates": [[26.375, 43.488], [26.37, 43.492], [26.369

```

Фиг. 2.16. JSON файл с административните области на България

```

ticket-app > ticket-app > static > js > JS map.js > ...
You, 3 days ago | 1 author (You)
1  document.addEventListener('DOMContentLoaded', function() {
2      const width = 800;
3      const height = 600;
4
5      const svg = d3.select('#map')
6          .append('svg')
7          .attr('width', width)
8          .attr('height', height);
9
10     const projection = d3.geoMercator()
11         .center([25.5, 42.7])
12         .scale(3500)
13         .translate([width / 2, height / 2]);
14
15     const path = d3.geoPath().projection(projection);
16
17     d3.json('/static/data/bulgaria-regions.json').then(function(data) {
18         svg.selectAll('.region')
19             .data(data.features)
20             .enter()
21             .append('path')
22             .attr('class', 'region')
23             .attr('d', path)
24             .on('click', function(event, d) {
25                 // Redirect to the concerts page for the clicked region
26                 window.location.href = '/concerts/' + encodeURIComponent(d.properties.nuts3);
27             });
28     });
29 });

```

Фиг. 2.17. JavaScript файл за визуализиране на картата на база JSON файл - фиг. 2.15

В app.ru има допълнителни функции които се използват за превеждане на областите от български на английски език, за различните заявки, и заявки за извеждане на изображения от сайтовете грабо и билет бг.

```

65 def translate_city(city_name):
66     # Translate the city name from Bulgarian to English
67     translated = GoogleTranslator(source='bg', target='en').translate(city_name)
68     return translated
69
70 def extract_image_url(image_div):
71     if image_div and 'style' in image_div.attrs:
72         style = image_div['style']
73         url_match = re.search(r'url\([\\"'']?(.+?)[\\"'']?\)', style)
74         if url_match:
75             return url_match.group(1)
76     return ""
77
78 if __name__ == '__main__':
79     app.run(debug=True)

```

Фиг. 2.18. Допълнителни функции в app.py

За визуализацията на този код са използвани шаблоните index.html и concerts.html представени в следните фигури:

```

ticket-app > ticket-app > templates > index.html > html > body > nav > ul > li > a
You, 3 hours ago | 1 author (You)
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Tickets App – Upcoming Events</title>
7     <script src="https://d3js.org/d3.v7.min.js"></script>
8     <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap" rel="stylesheet">
9     <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
10 </head>
11 <body>
12     <nav>
13         <ul>
14             <li><a href="{{ url_for('index') }}">Начало</a></li>
15             <li><a href="{{ url_for('scrape_concerts', region='София') }}">София</a></li>
16             <li><a href="{{ url_for('scrape_concerts', region='Пловдив') }}">Пловдив</a></li>
17             <li><a href="{{ url_for('scrape_concerts', region='Варна') }}">Варна</a></li>
18             <li><a href="{{ url_for('scrape_concerts', region='Бургас') }}">Бургас</a></li>
19             <li><a href="{{ url_for('scrape_concerts', region='Русе') }}">Русе</a></li>
20             <li><a href="{{ url_for('scrape_concerts', region='Велико Търново') }}">Велико Търново</a></li>
21         </ul>
22     </nav>
23     <div class="content">
24         <h1>Престоящи събития из цялата страна</h1>
25         <div id="map"></div>
26     </div>
27     <script src="{{ url_for('static', filename='js/map.js') }}"></script>
28 </body>
29 </html>

```

Фиг. 2.19. index.html

```

22 | <div class="content">
23 |     <h1>Предстоящи събития в {{ region }}</h1>
24 |     <table>
25 |         <thead>
26 |             <tr>
27 |                 <th>Снимка</th>
28 |                 <th>Име</th>
29 |                 <th>Локация</th>
30 |                 <th>Дата</th>
31 |             </tr>
32 |         </thead>
33 |         <tbody>
34 |             {% for concert in concerts %}
35 |             <tr>
36 |                 <td></td>
37 |                 <td>{{ concert.name }}</td>
38 |                 <td>{{ concert.place }}</td>
39 |                 <td>{{ concert.date }}</td>
40 |             </tr>
41 |             {% endfor %}
42 |         </tbody>
43 |     </table>
44 | </div>
45 </body>
46 </html>

```

Фиг. 2.20. Част от *concerts.html*

Дизайнът е изнесен в отделен шаблон - *style.css*. Проектът се пуска с командата: **python app.py**. Със стартиране на приложението се създава локален webserver който работи по подразбиране на <http://127.0.0.1:5000>. Фигури на приложението може да видите по долу:

```

ticket-app > ticket-app > static > css > # styles.css > h1
1  body {
2      font-family: 'Roboto', sans-serif;
3      line-height: 1.6;
4      color: #333;
5      max-width: 1200px;
6      margin: 0 auto;
7      padding: 20px;
8      background-color: #f4f4f4;
9      display: flex;
10     flex-direction: column;
11     min-height: 100vh;
12 }
13
14 h1 {
15     text-align: center;
16     color: #2c3e50;
17     margin-bottom: 30px;
18     font-weight: 700;
19 }
20
21 table {
22     border-collapse: separate;
23     border-spacing: 0;
24     width: 100%;
25     background-color: #fff;
26     box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
27     border-radius: 8px;
28     overflow: hidden;
29 }
30
31 th, td {
32     padding: 15px;
33     text-align: left;
34     border-bottom: 1px solid #e0e0e0;
35 }
36
37 th {
38     background-color: #3498db;
39     color: #fff;

```

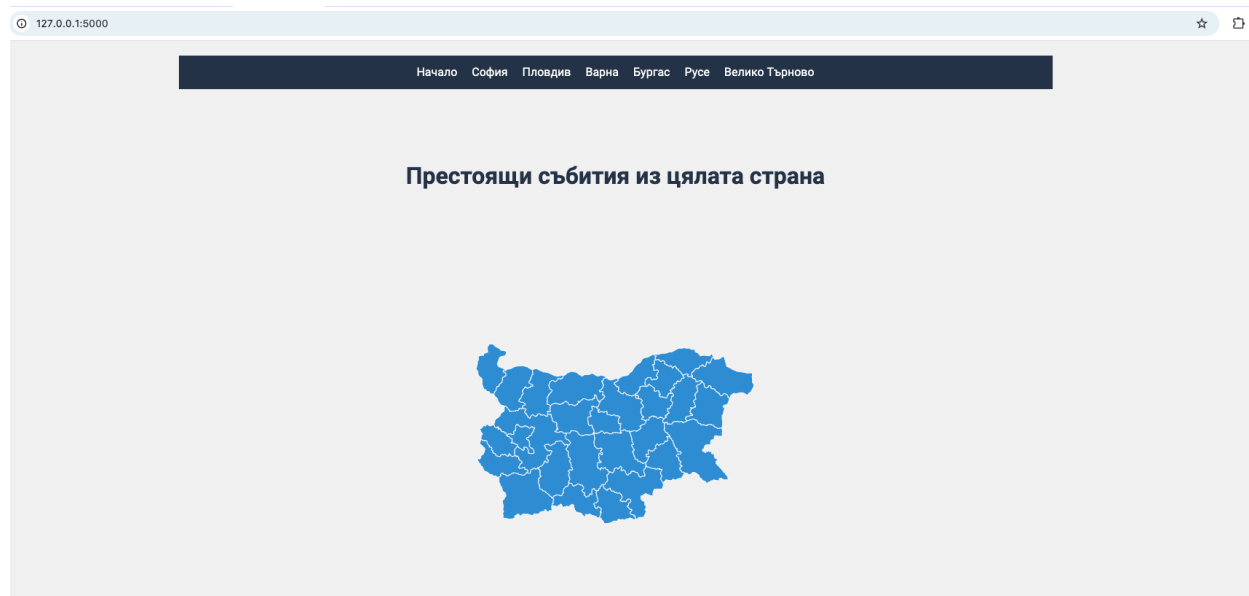
Фиг. 2.21. Част от style.html

```

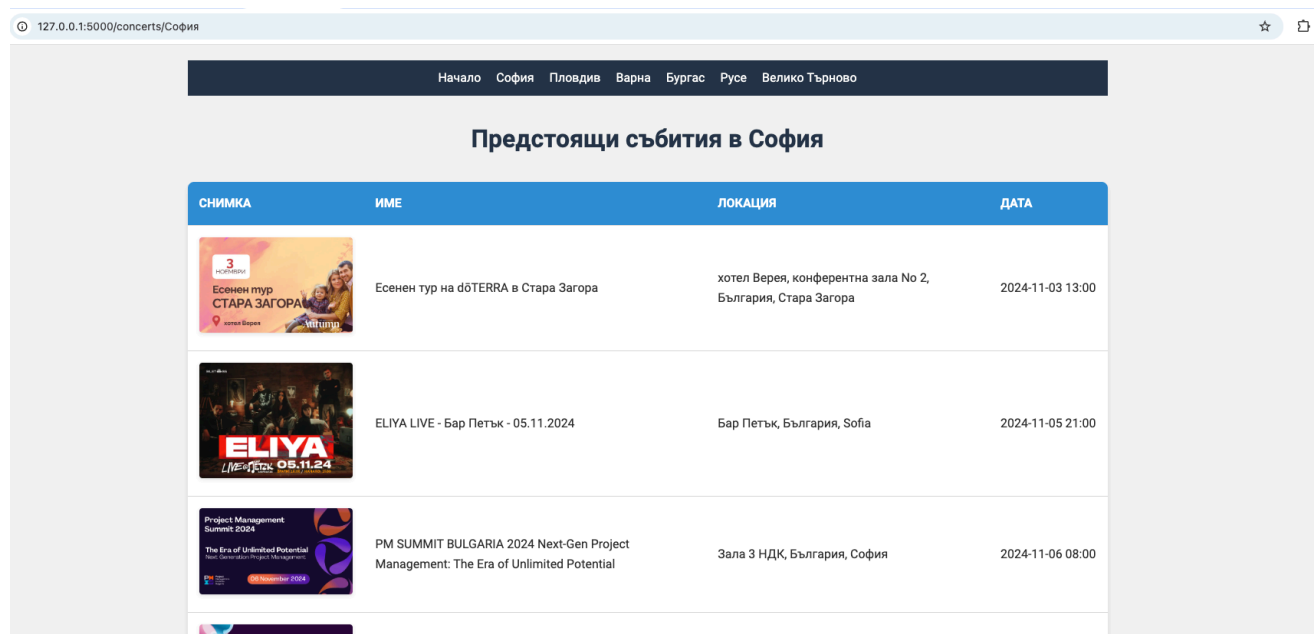
Daniels-MacBook-Pro:ticket-app danieldimitrov$ python3.10 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 729-453-896
127.0.0.1 - - [02/Nov/2024 21:53:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:53:08] "GET /static/js/map.js HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:53:08] "GET /static/css/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:53:08] "GET /static/data/bulgaria-regions.json HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:53:10] "GET /concerts/София HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:53:10] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Nov/2024 21:53:57] "GET /concerts/Враца HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:53:57] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Nov/2024 21:54:06] "GET /concerts/София HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:54:06] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Nov/2024 21:57:55] "GET /concerts/Пыце HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:57:55] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Nov/2024 21:57:59] "GET /concerts/Варна HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:57:59] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Nov/2024 21:58:04] "GET /concerts/Бургас HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:58:04] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Nov/2024 21:58:09] "GET /concerts/Стара Загора HTTP/1.1" 200 -
127.0.0.1 - - [02/Nov/2024 21:58:09] "GET /static/css/styles.css HTTP/1.1" 304 -

```





Фиг. 2.22. Стартиране на приложението







Фиг. 2.23. Начална страница на приложението



Фиг. 2.24. <http://127.0.0.1:5000/concerts/София>

<a href="#">Начало</a> <a href="#">София</a> <a href="#">Пловдив</a> <a href="#">Варна</a> <a href="#">Бургас</a> <a href="#">Русе</a> <a href="#">Велико Търново</a>			
Предстоящи събития в Русе			
СНИМКА	ИМЕ	ЛОКАЦИЯ	ДАТА
	Сватба със закъснител	Русе, Доходно здание - Голяма сцена	21 ноември (чт), 19:00ч
	Оперен рецитал на Анна Дитри-Калудов		5 ноември (вт), 19:00ч
	Симфоничен концерт - 100 години от рождението на Константин Илиев	Русе, Зала Филхармония	8 ноември (пт), 19:00ч
			

Фиг. 2.25. <http://127.0.0.1:5000/concerts/Русе>

<a href="#">Начало</a> <a href="#">София</a> <a href="#">Пловдив</a> <a href="#">Варна</a> <a href="#">Бургас</a> <a href="#">Русе</a> <a href="#">Велико Търново</a>			
Предстоящи събития в Бургас			
СНИМКА	ИМЕ	ЛОКАЦИЯ	ДАТА
	Кошмарна комедия	Бургас, Културен дом на нефтохимика - НХК	4 ноември (пн), 19:00ч
	Костенурката Нурка	Бургас, Държавен куклен театър Бургас	3 ноември (нд), 11:00ч
	Концерт на Райко Кирилов и Йорданка Варджийска	Бургас, Културен дом на нефтохимика - НХК	5 ноември (вт), 19:00ч
			

Фиг. 2.26. <http://127.0.0.1:5000/concerts/Бургас>



# ИЗПОЛЗВАНА ЛИТЕРАТУРА

[1] За Version Control системите -

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

[2] Кратка история на Git -

<https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>

[3] What is Git? -

<https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

[4] About GitHub and Git -

<https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>

[5] Bulgaria-geocoding -

[https://github.com/yurukov/Bulgaria-geocoding/blob/master/provinces.geojson?short\\_path=eba900c](https://github.com/yurukov/Bulgaria-geocoding/blob/master/provinces.geojson?short_path=eba900c)