

Документация на проект по курс CITB679 Проект: Бизнес Информационни Системи

Изготвили:

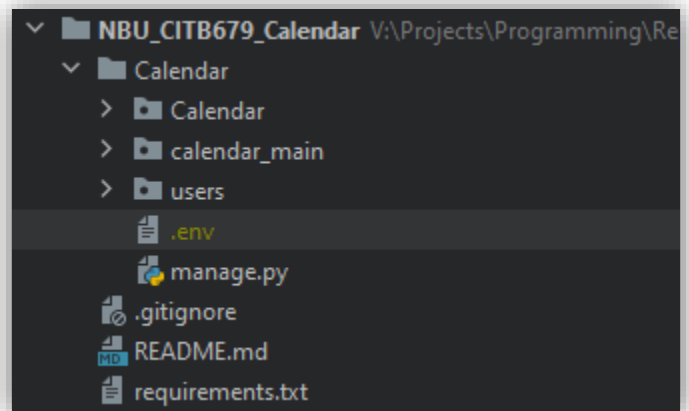
- Боян Стефанов Бонев F91233
- Стоян Георгиев F92067
- Иван Стоянов F90980

Специалност: „Информационни Технологии“

Installation

Проектът използва .env файл, в който се намират променливите, чрез които Django работи с базата, както и тези за OAuth2 авторизация (използвани от модула social-auth-django-app). Той трябва да съществува в root-а на Django проектът (на същото ниво като manage.py).

1) Трябва да се създаде MySQL база и потребител, след което нужната информация да се изведе в следните променливи:



DB_NAME	<името на базата с която Django ще работи>
DB_USER	<името на потребителя, чрез който Django ще работи с базата>
DB_PASS	<паролата на горе-споменатия потребител на базата>

DB_HOST	<Default localhost>
DB_PORT	<Default 3306>

MySQL download: <https://www.mysql.com/downloads/>

След инсталацията трябва да се влезе в MySQL Server и да се създадат база и потребител с необходимите права:

```
> mysql
> CREATE DATABASE 'database_name'
> USE 'database_name'
> CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';
> GRANT ALL PRIVILEGES ON * . * TO 'new_user'@'localhost';
```

2) За да се използва OAuth2 интеграцията със GitHub, трябва апликацията да се регистрира тук: <https://github.com/settings/applications/new>, след което да се запишат необходимите променливи в .env файлът.

Дори ако функционалността няма да се използва, променливите трябва да присъстват във файла с някаква стойност, за да тръгне проекта!

```
SOCIAL_AUTH_GITHUB_KEY
SOCIAL_AUTH_GITHUB_SECRET
```

3) Трябва да се инсталират необходимите Python модули. Използваната версия на Python е 3.8.8, но се предполага да работи и със по-стари версии. Необходимите пакети са описани във файла requirements.txt и те могат да се изтеглят с командата:

```
> pip install -r requirements.txt
```

Забележка: Може да възникнат проблеми при инсталацията на модула mysqlclient под Линукс. Подробна информация относно инсталацията на модула има на официалната страница на пакета на: <https://pypi.org/project/mysqlclient/>

4) Трябва да се мигрира базата. Това става чрез изпращането на команда до Django посредством файла `manage.py`

```
> python.exe manage.py migrate (Windows)
```

```
> python3 manage.py migrate (Linux)
```

5) За да се стартира сървърът трябва да се извика скриптът `manage.py` с аргумент

Implementation Documentation

Backend

За разработката на backend е използван Django framework (3.2.3), както и следните пакети:

- ✓ **mysqlclient** (2.0.3) - за комуникация с базата
- ✓ **python-dotenv** (0.17.1) - за зареждане на променливите от файлът `.env`
- ✓ **social-auth-app-django** (4.0.0) - за OAuth2 авторизация с GitHub

Django framework използва design pattern на име MVT (Model View Template). Това е аналог на MVC (Model View Controller), в който обаче логиката е следната:

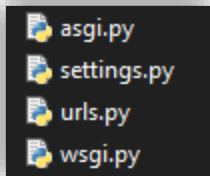
- ✓ **Model** - представлява моделите на обектите в базата
- ✓ **View** - това са функциите менажиращи темплейтите
- ✓ **Template** - това са front-end html файловете

Django работи с компоненти наречени Apps. В root-а на един Django проект винаги има поне 1 app, (който е главен) и файлът `manage.py`, който служи като command-line api за работи с Django. Всяка уникална функционалност в Django се разработва чрез свой собствен App. Този проект се състои от 3 такива:

- ✓ **Calendar** - това е главния ап, в него се намира конфигурацията на целия проект

- ✓ **calendar_main** – това е логиката на видимия календар и всичката му функционалност
 - ✓ **users** – този ап се занимава с регистрацията на потребители
-

Главният ап се състои от следните файлове:



Основни за работата на приложението са settings.py и urls.py.











Файлът settings.py съдържа пълната конфигурация на приложението. Най-основни конфигурации в този файл са списъците съдържащи инсталираните апликации (определя кои Django apps да бъдат използвани), междинен софтуер (например AuthenticationMiddleware, който handle-ва създаването на потребители), както и ROOT_URLCONF, който определя основния urls файл, който да бъде използван.

```
34
35 INSTALLED_APPS = [
36     'calendar_main',
37     'users',
38     'social_django',
39     'django.contrib.admin',
40     'django.contrib.auth',
41     'django.contrib.contenttypes',
42     'django.contrib.sessions',
43     'django.contrib.messages',
44     'django.contrib.staticfiles',
45 ]
46
47 MIDDLEWARE = [
48     'django.middleware.security.SecurityMiddleware',
49     'django.contrib.sessions.middleware.SessionMiddleware',
50     'django.middleware.common.CommonMiddleware',
51     'django.middleware.csrf.CsrfViewMiddleware',
52     'django.contrib.auth.middleware.AuthenticationMiddleware',
53     'django.contrib.messages.middleware.MessageMiddleware',
54     'django.middleware.clickjacking.XFrameOptionsMiddleware',
55 ]
56
57 ROOT_URLCONF = 'calendar.urls'
```

Файлът urls.py съдържат всички endpoints на апликацията. В него програмистично са добавени и линковете от останалите апове (от съответните им urls.py файлове).

```
urlpatterns = [
    path('', include('users.urls')),
    path('', include('calendar_main.urls')),
    path('admin/', admin.site.urls),
]
```

Структурата на един Django App (в случай, че не става въпрос за главния) изглежда по следния начин:

 migrations	migrations – съдържа файловете, които се използват при миграция на базата.
 templates	templates – аналог на views в MVC design pattern. Съдържа html кода, който се ползва за визуализиране на сайта.
 <code>__init__.py</code>	
 <code>admin.py</code>	admin.py – настройки за админския интерфейс
 <code>apps.py</code>	apps.py – съдържа конфигурации на апа
 <code>forms.py</code>	forms.py – настройки за формите
 <code>models.py</code>	models.py – съдържа всички модели, използвани в съответния ап (с изключение на тези предоставени от Django)
 <code>tests.py</code>	test.py – съдържа unit тестове (ние нямаме 😞)
 <code>urls.py</code>	urls.py – тук са адресите дефинирани от съответния ап.
 <code>views.py</code>	view.py – аналог на контролерите в MVC design pattern. Чрез тези методи се оправляват темплейтите.

Frontend

За frontend на проекта се използва bootstrap4 - <https://getbootstrap.com/>

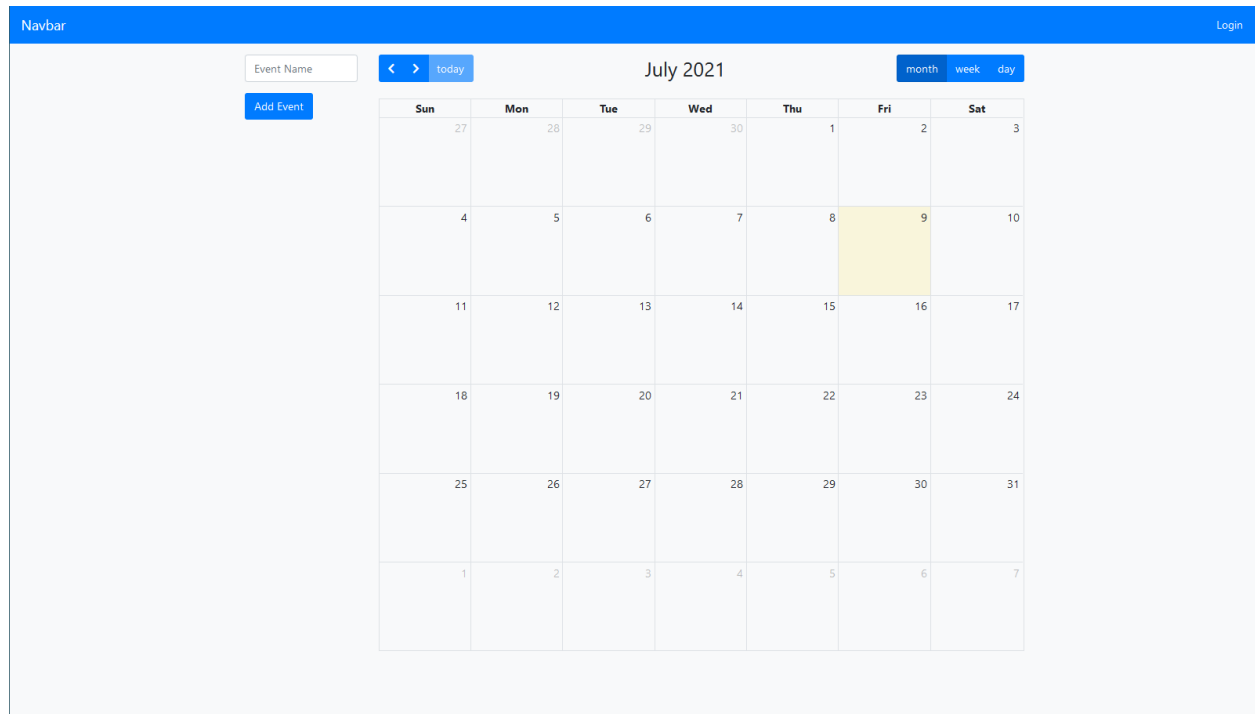
Календарът е реализиран посредством библиотеката fullcalendar.io - <https://fullcalendar.io/>

За комуникация със сървъра се използва jQuery AJAX.

```
$.ajax({
  url: '/save/',
  type: 'POST',
  data: JSON.stringify({
    events: eventObjects
  }),
  contentType: 'application/json; charset=utf-8',
  dataType: 'json',
  success: function(result) {
    console.log(result)
  },
  error: function(err) {
    console.log(err)
    alert('There was an error while adding event! Please try again later!');
  },
});
```

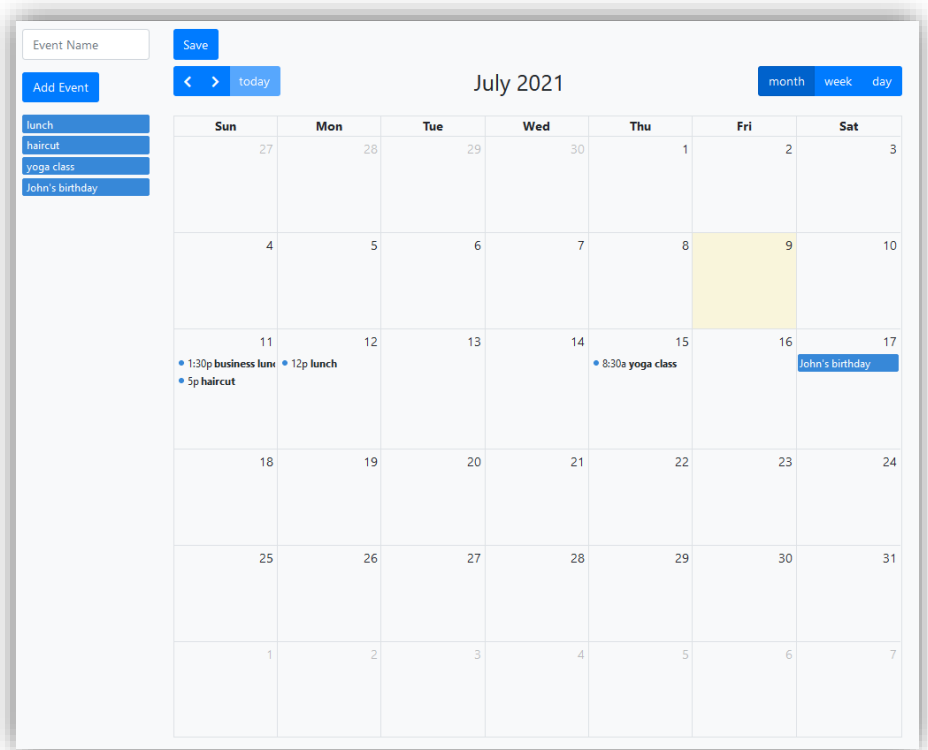
Използва се CDN, вместо локални статични файлове.

Функционалност



Календарът дава възможност да се дефинират Event обекти, които да се поставят в календара, след което тяхното заглавие и дължина може да бъде променяно. Стандартно, при поставяне на ново събитие в календара, то се счита с целодневна продължителност.

Нищо по календара не се запазва преди натискането на бутона Save.



Event Name

Save

Add Event

<

>

today

Jul 11 – 17, 2021

month

week

day

lunch

haircut

yoga class

John's birthday

	Sun 7/11	Mon 7/12	Tue 7/13	Wed 7/14	Thu 7/15	Fri 7/16	Sat 7/17
all-day							John's birthday
6am							
7am							
8am							
9am					8:30 - 10:00 yoga class		
10am							
11am							
12pm		12:00 - 12:30 - lunch					
1pm							
2pm	1:30 - 3:00 business lunch						
3pm							
4pm							
5pm	5:00 - 5:30 - haircut						
6pm							
7pm							
8pm							

Event Name

Save

Add Event

<

>

today

July 11, 2021

month

week

day

lunch

haircut

yoga class

John's birthday

	Sunday
all-day	
10am	
11am	
12pm	
1pm	
2pm	1:30 - 3:00 business lunch
3pm	
4pm	
5pm	5:00 - 5:30 - haircut
6pm	
7pm	
8pm	
9pm	
10pm	
11pm	

Код

calendar_main

Тук са 2-те основни view функции calendar и save:

calendar – извежда от базата събитията дефинирани от потребителя и ги зарежда в „контекста“ на темплейта (контекстът представлява чист Python dict обект (речник), достъпен в HTML посредством темплейтният език използван от Django framework – Django HTML). Това се случва само в случай, че имаме автентизиран потребител.

```
12 def calendar(request: WSGIRequest):
13     context = {'calendar_events': []}
14
15     if request.user.is_authenticated:
16         calendar_events = []
17         events = models.Event.objects.filter(created_by=request.user.id)
18
19         for event in events:
20             calendar_events.append({
21                 "id": event.pk,
22                 "title": event.title,
23                 "start": event.start_date.isoformat(),
24                 "end": event.end_date.isoformat() if (event.end_date is not None) else None,
25                 "allDay": True if (event.end_date is None) else False
26             })
27         context["calendar_events"] = calendar_events
28
29     return render(request, "calendar.html", context)
```

save – Синхронизира събитията на календара с тези в базата. Ако в базата има събития, които не се намират върху календара, те биват изтрети, а ако на календара има събития, които не се намират в базата – те се създават. За останалите събития се извършва update.

```
31 def save(request: WSGIRequest):
32     response = {'isSuccessful': False}
33
34     if request.method == "POST":
35         all_events = models.Event.objects.filter(created_by_id=request.user.id)
36         data = json.loads(request.body)
37         events = data['events']
38
39         try:
40             valid_ids = [ev['id'] for ev in events]
41             for ev in all_events:
42                 if ev.id not in valid_ids:
43                     ev.delete()
44
45             to_update = []
46             to_create = []
47
48             for event in events:
49                 model = models.Event(
50                     title=event["title"],
51                     start_date=event["start"],
52                     end_date=None if (event["allDay"] is True) else event["end"],
53                     created_by=request.user,
54                     created_on=datetime.utcnow()
55                 )
56
57                 if not event["id"]:
58                     to_create.append(model)
59                 else:
60                     model.pk = event["id"]
61                     to_update.append(model)
62
63             models.Event.objects.bulk_update(to_update, ['title', 'start_date', 'end_date'])
64             models.Event.objects.bulk_create(to_create)
65
66         except Exception as exc:
67             response["isSuccessful"] = False
68             return JsonResponse(response)
69
70     return JsonResponse({})
```