# Web servis za sinkronizaciju događaja iz više kalendara

**Cajbert, Tomislav**





**SVEUČILIŠTE JOSIPA JURAJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni studij**

# WEB SERVIS ZA SINKRONIZACIJU DOGAĐAJA IZ VIŠE KALENDARA

**Diplomski rad**

**Tomislav Cajbert**

**Osijek, 2020.**

# SADRŽAJ

# 1. UVOD

U ovom diplomskom radu rješavat će se problem automatske sinkronizacije događaja iz više kalendara koji se nalaze na različitim računima. Diplomski rad će objasniti zašto je potrebno napraviti automatsku sinkronizaciju za kalendare, opisat će kalendare i događaje koji se nalaze u njima, opisat će se podatkovna reprezentacija kalendara i događaja, objasnit će se način na koji se postigla sinkronizacija kalendara, proći će se kroz programsko rješenje, prepreke koje su predstavljale problem pri izradi programskog rješenja, te koji su zaključci donijeti na završetku istraživanja informacija o kalendarima i kreiranja programskog rješenja. Struktura rada je postavljenja tako da će se u drugom poglavlju govoriti o svim ključnim informacijama za kalendare, njihovo predstavljanje, na kojima se servisima mogu pronaći i sve potrebne informacije koje se moraju znati za izradu programskog rješenja. Treće poglavlje će govoriti o značajnim programskim alatima koji su se koristili u izradi programskog rješenja. Četvrto poglavlje će prikazati programske probleme i rješenja za izradu servisa za automatsku sinkronizaciju kalendara. Peto poglavlje će prikazati korisničko sučelje i dati opis funkcija koje se pojavljuju na njemu, te kako korisnik može vršiti interakcije s njim. Šesto poglavlje je zaključak u kojem će se govoriti o stvarima koje su bile ključne kroz rad i koje bi se potencijalne nadogradnje mogle dodati.

## 1.1. Postojeća rješenja

Mali je broj ljudi koji imaju potrebu za rješenjem koje je opisano u ovome radu, tako da za tu funkciju i nema velike ponude rješenja. Ljudi koji imaju korist od ovakvog rješenja su najčešće oni koji blisko rade s više poslovnih kompanija odjednom, te od svake imaju poslovni mail za interakciju s njima. Google ima vlastito rješenje gdje daje mogućnost da se podijeli određeni kalendar s određenom osobom. Iako ova solucija rješava problem davanje pregleda svog rasporeda drugim stranama na uvid dostupnosti, ovo nije idealno rješenje zato što se sadržaj cijelog kalendara da na uvid drugim osobama, što može predstavljati veliki problem zbog brige o osjetljivim informacijama, ali je zato potpuno siguran zato što se nalazi u sklopu Google servisa te se može sa sigurnošću reći da neće zlorabiti svoj pristup prema kalendarima. Postoji jedno jako slično rješenje kao što je ovo ponuđeno u radu, zove se „SyncThemCalendars“, ovo rješenje je jako dobro zato što pokriva sve ključne stvari potrebne za sinkronizaciju kalendara. Prednost ovog rješenja je što se može odrediti kontinuirana sinkronizacija između dva kalendara, a nedostaci su to da postoji besplatan probni rok od četrnaest dana nakon kojeg se korištenje aplikacije naplaćuje 5$ mjesečno i to što se ne može sa sigurnošću reći da ne spremaju podatke koje dohvate s korisnikovih kalendara.

# 2. KALENDARI I DOGAĐAJI U NJIMA

Ovaj diplomski rad obrađuje temu sinkronizacije kalendara, tako da će se ovo poglavlje iskoristiti kako bi se dale neke ključne informacije vezane za kalendar što predstavlja, zašto se koristi, kako je napredovao kroz povijest. Pošto se radi o sinkronizaciji kalendara dat će se razlog sinkroniziranja, te opisati beneficije koje se dobiju nakon sinkronizacije. Kad se objasni zašto je to potrebno pregledat će se potrebne informacije i struktura podataka s kojom će se morati raditi kako bi se napravilo jedno takvo rješenje.

## 2.1. Kalendari i događaji

Kalendar je sustav koji se koristi za organizaciju vremena. U njemu se prikazuje neko vremensko razdoblje [1]. Kalendar se počeo davno koristiti na papirima u kojima je bilo prikazano neko vremensko razdoblje najčešće podijeljeno po danima, a koristio se tako da su se stavljale oznake za određena razdoblja koja ukazuju na neki religiozni, društveni ili poslovni događaj na koji se ne bi trebalo zaboraviti [2]. Kalendar je napretkom tehnologije prešao na digitalne uređaje te je tamo dobio puno više mogućnosti kao što su na primjer podsjetnici, mjesto događaja, polaznici i drugo. Kalendar je jak alat za upravljanje sastancima zato što se može postaviti mjesto događaja, mogu se poslati pozivnice, podijeliti dodatne informacije o sastanku, te se to se sve može napraviti kroz par klikova što olakšava poslovanje.

## 2.2. Zašto automatska sinkronizacija

Automatska sinkronizacija je pogodna kada osoba ima više kalendara s kojima raspolaže te treba sinkronizirati kalendare da se svi događaji nalaze na jednom mjestu. To se radi iz razloga kako bi vlasnik kalendara imao bolji pregled nadolazećih događaja, što znači da bi postojala manja vjerojatnost da bi se neki događaj propustio, također je bitna prednost sinkroniziranih kalendara ta da se prikaže slobodna dostupnost strankama koje žele ugovoriti neki sastanak ili slično [3]. Automatska sinkronizacija već postoji na kalendarskim servisima, ali da bi se ta opcija uključila potrebno je kalendar napraviti javnim što znači da bi se vidjele sve detaljne informacije o događajima na sinkroniziranom kalendaru. Prethodno opisane beneficije najviše pogoduju za poslovnu okolinu gdje informacije kao što su ljudi s kojima se sastaje, mjesto, detaljan opis sastanka često su poslovna tajna te se ne smije dogoditi da se te informacije stave javno da ih neka treća strana može vidjeti.

## 2.3. Potrebne informacije za izradu programskog rješenja

Diplomski rad će obrađivati programsko rješenje za automatsku sinkronizaciju više kalendara s različitih profila na Google Calendar servisu. Google Calendar servis je napravljen od strane Google kompanije, a služi za upravljanje i planiranje vremenskim rasporedom, u generalnu upotrebu je pušten u Srpnju 2009. godine te je od tada primio velik broj nadogradnji kako bi imao funkcionalnosti koje ima danas [4]. Jako je pogodan poslovnom okolišu zato što Google omogućuje integraciju tuđih poslova na njihov sustav da se sva organizacija može nalaziti na jednoj platformi, te time daje alate koji su međusobno povezani što pomaže prebacivanjem informacija iz jednog alata u druge. Za izradu programskog rješenja koristit će se Google Calendar API, on daje skupinu funkcija koja se može koristiti u svrhu vanjskog upravljanja kalendara i događaja u njemu.

## 2.4. Struktura kalendara i događaja

Google Calendar API nakon što odobri promjene na kalendaru za određen program, te određenog korisnika daje na raspolaganje set krajnjih točaka koje program koristi kako bi komunicirao korisnikove želje za promjene u njegovom kalendaru. Struktura kalendara i njegovih događaja koja se koristi za razmjenu podataka između programa i Google Calendar aplikacije je u JSON formatu može se naći na referenci [6], osnovni atributi koji će se koristiti u programskom rješenju su prikazani na slici 2.1.

```
{
  "kind": "calendar#event",
  "etag": etag,
  "id": string,
  "status": string,
  "htmlLink": string,
  "created": datetime,
  "updated": datetime,
  "summary": string,
  "description": string,
  "location": string,
  "colorId": string,
  "creator": {
    "id": string,
    "email": string,
    "displayName": string,
    "self": boolean
  },
  "organizer": {
```

```
    "id": string,
    "email": string,
    "displayName": string,
    "self": boolean
  },
  "start": {
    "date": date,
    "dateTime": datetime,
    "timeZone": string
  },
  "end": {
    "date": date,
    "dateTime": datetime,
    "timeZone": string
  },
  "endTimeUnspecified": boolean,
  "recurrence": [
    string
  ],
  "recurringEventId": string,
  "originalStartTime": {
    "date": date,
    "dateTime": datetime,
    "timeZone": string
  },
  "transparency": string,
  "visibility": string,
  "iCalUID": string,
  "sequence": integer,
  "attendees": [
    {
      "id": string,
      "email": string,
      "displayName": string,
      "organizer": boolean,
      "self": boolean,
      "resource": boolean,
      "optional": boolean,
      "responseStatus": string,
      "comment": string,
      "additionalGuests": integer
    }
  ],
}
```

**Slika 2.1.** Podatkovna struktura događaja

# 3. PROGRAMSKI ALATI

Razvoj aplikacija je znatno lakši kad se pronađu pravi alati za točne specifikacije aplikacije koja se izrađuje, tako da će ovo poglavlje dati pregled najvažnijih alata koji se koriste za izradu. Za pristupni dio je korišten React kako bi se korisniku pružilo korisničko sučelje koje je brzo, a i iskorištavanje ponavljajućih komponenti je super funkcija za korištenje pri razvoju stranice. Pozadinski sustav je napravljen pomoću Flask okvira koji pruža prilagođen način razvoja aplikacije. PostgreSQL baza je izabrana zato što je već dugo na ponudi, a i dalje se koristi što znači da je provjerena i pouzdana. Alat koji je ujedinio sve nabrojene alate je Visual Studio Code, editor za pisanje izvornog koda koji svojim mogućnostima daje puno više nego što njegova kategorizacija zvuči.

## 3.1. React (JavaScript)

React je JavaScript biblioteka koja je napravljena od strane Facebook i Instagram kompanija, puštena je na korištenje u ožujku 2013 godine, te je 29.05.2013 puštena otvoreni izvorni kod gdje su ljudi mogli dodavati svoje stvari u biblioteku. React se koristi za izradu korisničkih sučelja. Zasniva se na izradi komponenti koje se ugnježđuju jedna u drugu kako bi se prikazali podaci korisnicima, a to radi po principu *single page application* što znači da se korisniku u pregledniku učitavaju samo oni dijelovi koji su promijenjeni što omogućava veliko ubrzanje u odnosu na učitavanje cijele HTML stranice [6]. React se često pogrešno karakterizira kao okvir no on je biblioteka, ne dolazi s alatima za kreiranje HTTP zahtjeva, kao ni s mogućnošću preusmjeravanja na korisničkoj razini zato se s React bibliotekom često koristi broj drugih biblioteka kao što su React Router, Redux, Axios i druge. React je 2015 godine dobio inačicu React Native gdje su se mogle izrađivati aplikacije za iOS, Android i Windows platforme, a cilj je bio da se s istom sintaksom programiranja mogu napraviti različiti tipovi aplikacija [7].

## 3.2. Flask (Python)

Flask je mikro razvojni okvir za razvoj web aplikacija u Python programskom jeziku. Flask je dizajniran da bude proširiv programski okvir, što znači da sadrži samo osnovne stvari, a one dolaze iz tri glavne ovisnosti:

- Werkzeug
  - *debugging*,
  - preusmjeravanje,
  - *Web Server Gateway Interface* (WSGI)

- Jinja2
  - podrška za *template*
- Click
  - integracija *command-line*

Autor sve tri ovisnosti je Armin Ronacher koje je ujedno i autor Flask razvojnog okvira [8]. Iako je za razvoj web aplikacija često potrebno puno više alata kao na primjer podrška za rad s bazom podataka, validacija web formi, autentifikacija korisnika i drugo, to Flask razvojni okvir nema, ali je zato jako pogodan za proširivanje s raznim alatima koji se mogu strateški odabrati za vrstu web aplikacije koja se izrađuje.

## 3.3. PostgreSQL (SQL)

PostgreSQL je open-source sustav za upravljanjem baza podataka s objektno relacijskim sustavom upravljanja koji je započet kao istraživački projekt na University of California, Berkeley, a ima dugačku povijest koja datira još iz 1985 godine [9]. PostgreSQL ima velik broj značajki kao i velik broj naprednih značajki koje se teško mogu naći u besplatnim bazama i natječe se sa svim velikim izdavačima baza podataka kao što su Oracle, SQL server i MySQL. PostgreSQL se može koristiti s bilo kojim glavnim programskim jezicima na primjer C, C++, Pearl, Python, Java, PHP i ostalim. Koristi se i nadograđuje na SQL jezik za baze podataka. Ono što ističe PostgreSQL od svih drugih baza je da osim što ima velik broj funkcija još je i lako nadogradiva tako da se može i proširiti s funkcijama koje su potrebne [10]. U toj bazi se sigurno čuvaju podaci i može ju se lako skalirati za kompliciranе projekte s puno podatkovnog prometa.

## 3.4. Visual studio code

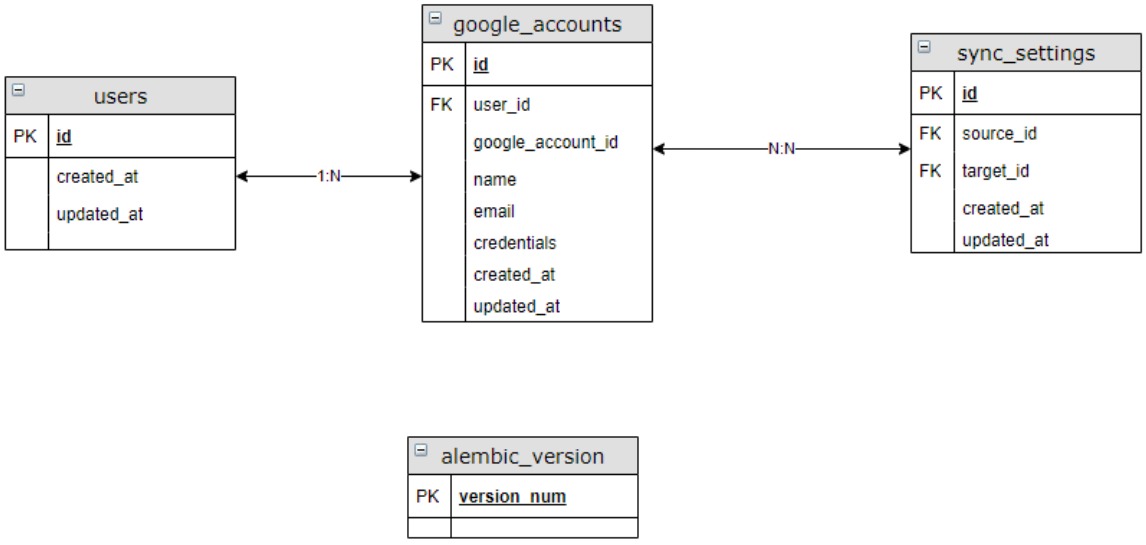
Visual studio code je prvi Microsoft Visual Studio alat koji se može pokrenuti na više različitih operacijskih sustava Windows, Linux, macOS. Visual studio code je besplatan i otvorenog izvornog koda za nadogradnju. Specifičan po tome što se fokusira oko uređivanja izvornog koda, upravljanja datotekama i datotečnim sustavom projekta. Pruža podršku za pisanje aplikacija na različitim platformama, ima velik broj podržanih jezika i bogatu kolekciju značajki za uređivanje koda kao što su „IntelliSense“, traženja referenci funkcija i varijabli, brzo dohvaćanje tipova podatka i više. Baziran je na Electron razvojnom okviru koji služi za razvijanje aplikacija koje se mogu pokrenuti na više različitih sustava. Implementira jednostavnost sa snažnim uređivačem koda s alatima koji su potrebni programeru za životni ciklusu aplikacije uključujući ispravljač grešaka i Git [11]. Znači nije samo uređivač teksta nego je kompletan razvojni alat.

# 4. RAZVOJ WEB APLIKACIJE „CalendarSync“

Ovo poglavlje će pisati o ključnim dijelovima u aplikaciji CalendarSync, najviše će se prolaziti kroz dijelove koji se odvijaju u pozadinskom servisu, tako da će se pričati o bazi i tablicama, rutama koje se pozivaju u svrhu interakcije pristupnog dijela i pozadinskog sustava aplikacije, načina kako se dolazi do Google Calendar resursa za pojedinog korisnika, te najvažnije kako sinkronizirati kalendare s dva različita Google računa.

## 4.1. Baza podataka

Za komunikaciju baze i pozadinskog dijela aplikacije koristi se SQLAlchemy, za praćenje nadogradnji baze podataka koristi se Flask-Migrate. Baza podataka će čuvati informacije o korisnicima, kada su korisnici izvršili sinkronizaciju računa, te promjene na bazi podataka. U bazi podataka se nalaze četiri tablice prikazane na slici 4.1.




**Slika 4.1.** Tablice podataka u sustavu

### 4.1.1. Tablica „users“

Ova tablica sadrži podatke o korisnicima koji su kreirali račun u aplikaciji nakon što su izabrali opciju „CREATE ACCOUNT“ na početnoj stranici, te se uspješno prijavili u svoj google račun. Tablica ima atribute: „id“ koji jedinstveno označava korisnika web aplikacije, „created_at“ kada

je korisnik kreiran, „updated_at“ kada su se promijenile neke informacije vezano za tog korisnika. U tablici se ne nalaze tipični podaci korisnika zato što se korisnici ne registriraju na našu stranicu, već se prijave preko Google računa.

```
class User(base):
    __tablename__='users'
    id = Column(UUID(as_uuid=True),primary_key=True,server_default=sqlalchemy.t
ext("uuid_generate_v4()"))
    created_at = Column(DateTime(timezone=True),nullable=False,server_default=f
unc.current_timestamp())
    updated_at = Column(DateTime(timezone=True))
    google_accounts = relationship('GoogleAccount',backref='user',lazy=True)
```

**Slika 4.2.** Tablica „users“ definirana u SQLAlchemy

### 4.1.2. Tablica „google_accounts“

U ovoj tablici se nalaze podaci od svih google računa koji se prijavljeni u aplikaciju putem „Sign up“ ili „Link account“ opcija. Tablica sadrži atribute „id“ redni broj google računa u tablici, „users_id“ strani ključ na tablicu „user“, „google_account_id“ jedinstveni google account identifikator, „name“ ime korisnika google računa, „email“ email adresa korisnika, „credentials“ kriptirani podaci korisnika koji se dobiju prilikom prijave na google, „created_at“ kada je račun kreiran u sustavu, „updated_at“ kada je račun mijenjan u sustavu.

```
class GoogleAccount(base):
    __tablename__='google_accounts'
    id = Column(Integer,primary_key=True)
    user_id = Column(UUID(as_uuid=True),ForeignKey('user.id'))
    google_account_id = Column(String,nullable=False)
    first_name = Column(String,nullable=False)
    email = Column(String(254),unique=True,nullable=False)
    credentials = Column(PickleType())
    created_at = Column(DateTime(timezone=True),nullable=False,server_default=f
unc.current_timestamp())
    updated_at = Column(DateTime(timezone=True))
```

**Slika 4.3.** Tablica „google_accounts“ definirana u SQLAlchemy

### 4.1.3. Tablica „sync_settings“

U ovoj tablici se nalaze podaci nastalih sinkronizacija u sustavu, tablica ima atribute „id“ jedinstveni redni broj sinkronizacije, „source_id“ strani ključ na „id“ iz „google_account“ tablicu

govori iz kojeg se google računa sinkronizira, „target_id“ strani ključ na „id “ iz „ google_account “ tablice govori u koji se google račun sinkroniziraju događaji.

```
class SyncSetting(base):
    __tablename__ = 'sync_settings'
    id = Column(Integer, primary_key=True)
    source_id = Column(Integer, ForeignKey('google_account.id'))
    target_id = Column(Integer, ForeignKey('google_account.id'))
    created_at = Column(DateTime(timezone=True), nullable=False, server_default
=func.current_timestamp())
    updated_at = Column(DateTime(timezone=True))
```

**Slika 4.4.** Tablica „sync_settings“ definirana u SQLAlchemy

### 4.1.4. Tablica „alembic_version“

Ova tablica predstavlja migracije iz Flask-Migrate biblioteke, a koristi se kako bi se pratile promjene na bazi podataka. Koristi se tako da se u direktoriju nalaze promijene na bazi koje se spremaju u tablicu kako bi se u nekim kasnijim slučajevima mogli vratiti na staro stanje baze.

## 4.2. Rute

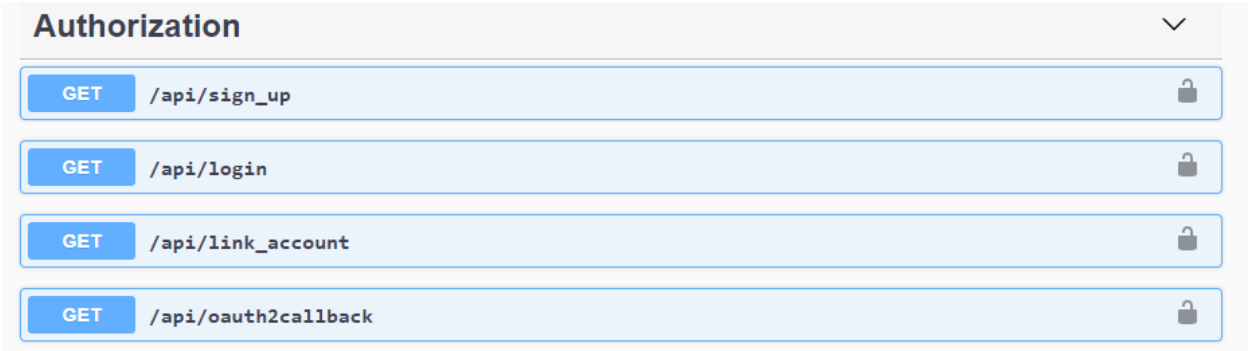
Rute se koriste kako bi se dohvatili i predali podaci pozadinskom sustavu za daljinu obradu. CalendarSync web aplikacija koristi nekoliko različitih grupa ruta koje služe za upravljanje s podacima u sustavu.

### 4.2.1. Autorizacijske rute

Autorizacijske rute služe kako bi se korisnik mogao prijaviti u CalendarSync sustav i koristiti njegove funkcionalnosti. Autorizacijske rute su:

- GET **„/api/login“** koristi se za preusmjeravanje korisnika na Google prijavu gdje je parametar „state“ jednak riječju „login“ kako bi se naznačilo da se korisnik autorizira na Google stranici u svrhu prijave u CalendarSync aplikaciju,
- GET **„/api/sign_up“** koristi se za preusmjeravanje korisnika na Google prijavu gdje je parametar „state“ jednak riječju „sign_up“ kako bi se naznačilo da se korisnik autorizira na Google stranici u svrhu kreiranja računa u CalendarSync aplikaciji,
- GET **„/api/link_account“ “** koristi se za preusmjeravanje korisnika na Google prijavu gdje je parametar „state“ jednak identifikatoru trenutnog korisnika kako bi se naznačilo aplikaciji koji korisnik želi povezati novi račun na svoj profil

- GET **„/api/callback“** služi za potvrdu autorizacijskog koda nakon uspješne autorizacije od strane Google sustava.

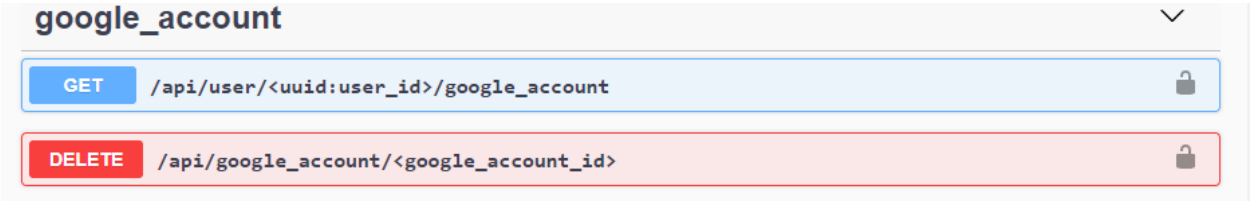



**Slika 4.5.** Autorizacijske rute

### 4.2.2. Google rute

Google rute služe za rad s podacima koji su vezani za Google račune. Google rute su:

- DELETE „/api/google_account/<google_account_id>“ je ruta koja služi za brisanje postojećeg Google računa iz baze podataka gdje je „<google_account_id>“ jednak jedinstvenom identifikatoru koji govori aplikaciji koji račun se treba obrisati.
- GET „/api/user/<uuid:user_id>/google_account“ ruta koja se koristi za dohvaćanje svih Google računa koji su povezani s trenutno prijavljenim korisnikom gdje je „<uuid:user_id>“ jedinstveni identifikator trenutno prijavljenog korisnika,



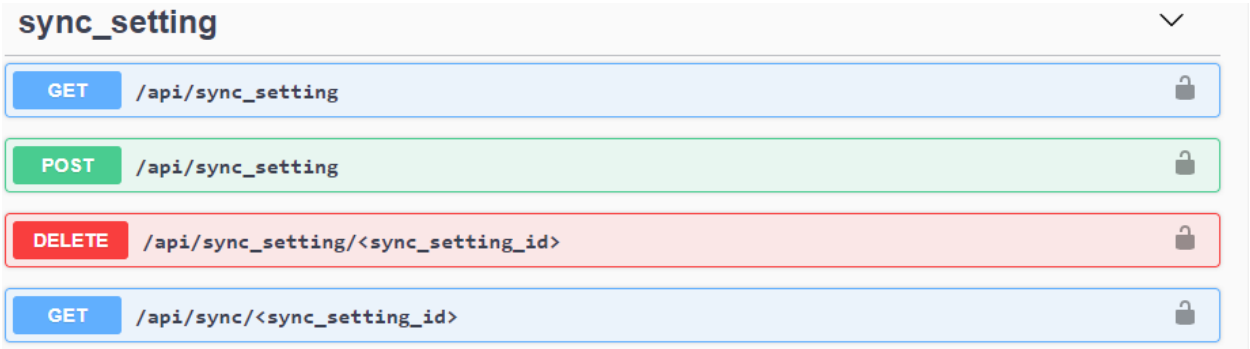

**Slika 4.6.** Google rute

### 4.2.3. Sinkronizacijske rute

Sinkronizacijske rute se koriste za rad s informacijama o sinkroniziranim računima. Rute koje se nalaze u ovoj grupi su:

- GET **„/api/sync_setting“** ruta koja dohvaća sve spremljene sinkronizacije iz baze podataka
- POST **„/api/sync_setting“** ruta za spremanje sinkronizacije u bazu podataka koja u sebi sadrži izvorišni i odredišni Google račun koji su povezani u aplikaciji

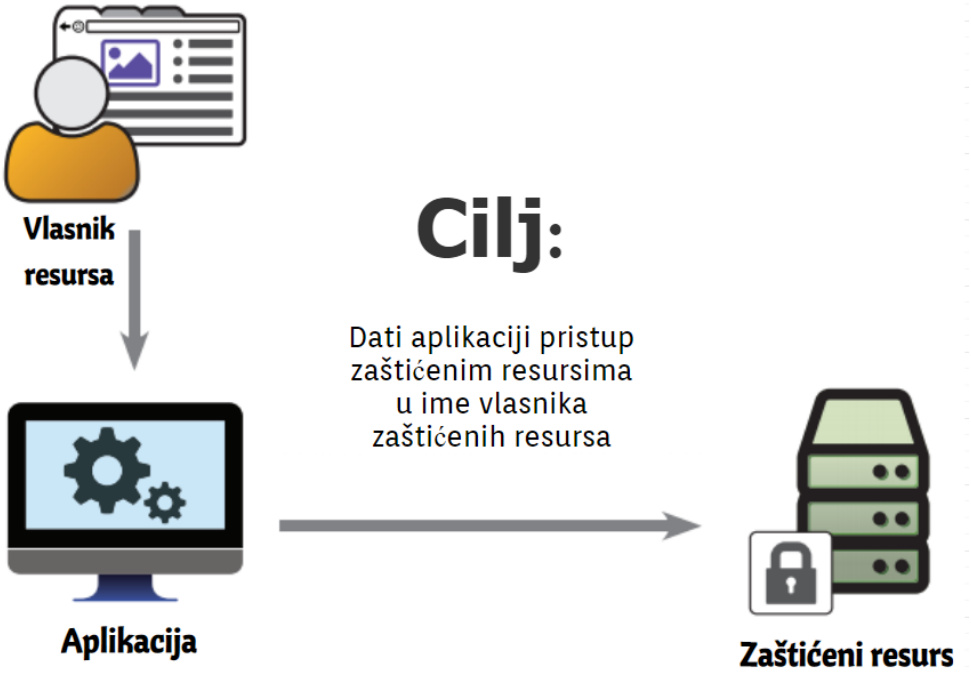
- DELETE **„api/sync_setting/<sync_setting_id>“** ruta koja briše informaciju za sinkroniziranje, a „<sync_setting_id“ je jedinstveni identifikator sinkronizacije spremljene u bazi
- GET **„/api/sync/<sync_setting_id>“** ruta se koristi za okidanje sinkronizacije između izvorišnog i odredišnog Google računa koji su povezani u aplikaciji, a <sync_setting_id> je jedinstveni identifikator sinkronizacije u bazi




**Slika 4.7.** Sinkronizacijske rute

## 4.3. OAuth

OAuth 2.0 je sigurnosni protokol. OAuth se može gledati u delegacijskom smislu, osoba koja ima neke resurse daje prava nekoj aplikaciji pristup tim podacima u njihovo ime. Taj postupak zahtjeva autorizaciju vlasnika resursa na mjestu gdje se nalazi resurs, tamo se dobije *token* koji se koristi za pristup resursu [13]. Slika 4.8 Cilj OAuth protokola.

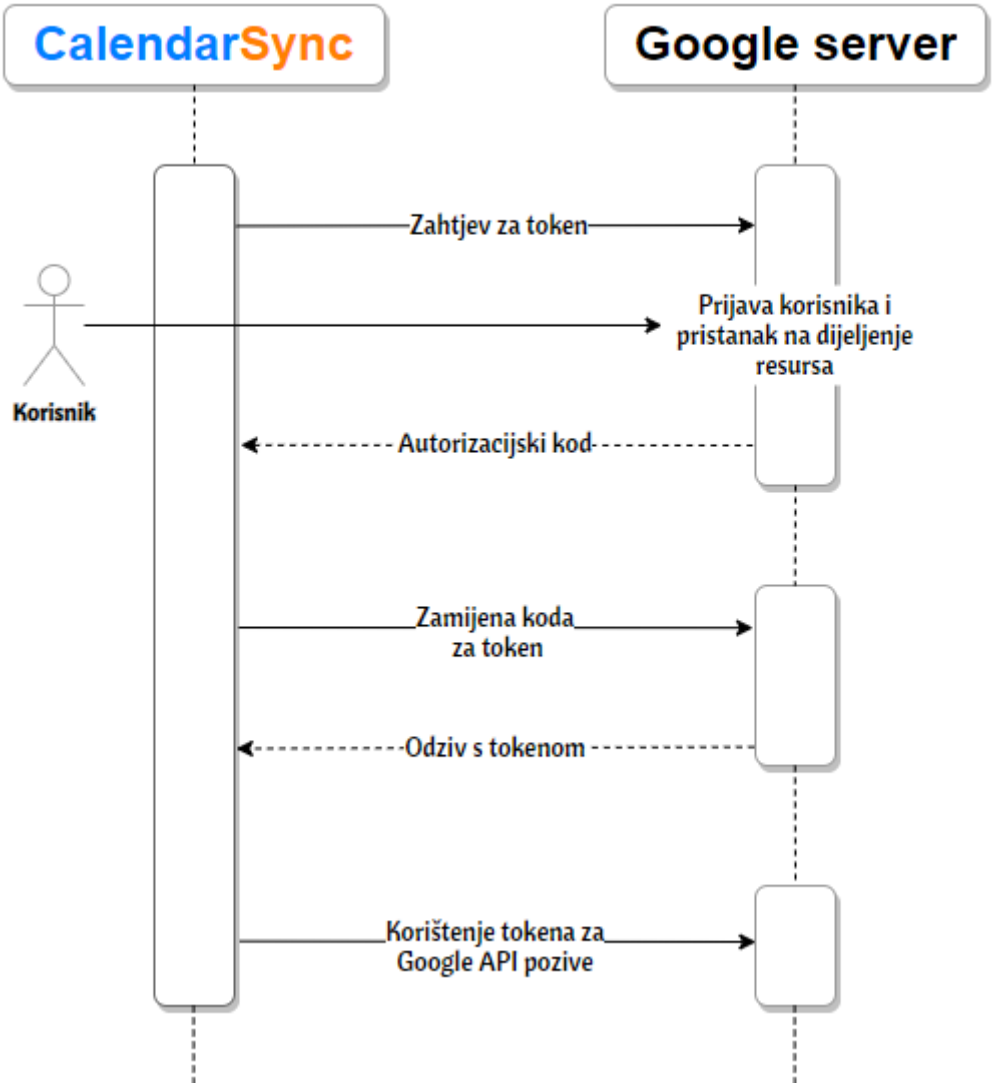



**Slika 4.8.** Cilj OAuth protokola

(**Izvor**: OAuth 2 in Action by Justin Richer Antonio Sanso. Published by Manning Publications. March. Shelter Island, NY. 2017)

### 4.3.1. Primjena Google OAuth 2.0 u aplikaciji CalendarSync

Osnovna funkcija CalendarSync web aplikacije je da radi sinkronizaciju kalendara s dva različita Google računa, kako bi se taj proces mogao odviti potrebno je imati token za pristup Google Calendar resursima. Koraci koji se moraju odviti kako bi CalendarSync aplikacija imala pristup korisnikovim kalendarima su prikazani na slici 4.9.




**Slika 4.9.** Koraci koji se odvijaju prije prava pristupa kalendaru

(**Izvor**: Developers Google – OAuth 2.0 web server, https://developers.google.com/identity/protocols/oauth2/web-server#python)

OAuth tok u CalendarSync aplikaciji:

1. Korisnik odlazi na CalendarSync stranicu i odabere „LOGIN“ ili „CREATE ACCOUNT“ opciju klikom na gumb, pristupni dio aplikacije poziva rute sa slike 4.10. koje aktiviraju proces autorizacije i šalju korisnika na autorizaciju na Google sustav.

```
@routes_blueprint.route("/api/sign_up")
def sign_up():
    state='sign_up'
    scopes = SIGN_UP_AND_LINK_SCOPES
    return authorize(scopes,state)


@routes_blueprint.route("/api/login")
def login():
    state='login'
    scopes = LOGIN_SCOPES
    return authorize(scopes,state)
```

**Slika 4.10.** Rute za prijavu i kreiranje računa

2. Nakon uspješne autorizacije korisnik se vraća nazad u sustav s autorizacijskim kodom koji pozadinski sustav aplikacije izmjeni za token koji sprema u sustav ako je različit ili ne postoji u bazi. Za „sign up“ korisnik se ubacuje u bazu, a za login se samo vrši provjera postojanja u bazi, te se u oba slučaja korisnik šalje na pristupni dio aplikacije gdje se prikazuje kontrolna ploča. Na slici 4.11. nalazi se osnovni dio *„oauth2callback“* rute koja se izvršava nakon autorizacije.

```
@routes_blueprint.route('/api/oauth2callback')
def oauth2callback():
    request = Request()
    state = url_request.args.get('state')
    if (state != 'login' and state != 'sign_up'):
      current_user_id = state
      state = 'link_account'
    scopes = SIGN_UP_AND_LINK_SCOPES if (state == 'sign_up' or state == 'link_account') else LOGIN_SCOPES
    flow = google_auth_oauthlib.flow.Flow.from_client_config(app_config.CREDENTIALS_FILE, scopes=scopes)
    flow.redirect_uri = flask.url_for('.oauth2callback', _external=True)
    authorization_response = flask.request.url
    query_string = {
            "user_id" : None,
            "message" : '',
            "status_code" : 400,
            "state" : 'unknown' }
    if state == 'sign_up':
        token=flow.fetch_token(authorization_response=authorization_response)
        credentials = flow.credentials
        id_info = id_token.verify_oauth2_token(token['id_token'], request,app_config.CREDENTIALS_FILE['web']['client_id'])
        google_account_id = id_info['sub']
        user_email=id_info['email']
        first_name=id_info['given_name']
        if (repository.query_google_account_count(google_account_id)) > 0:
            query_string['message'] = 'User already exists'
            query_string['state'] = state
        else:
            user_to_db = repository.save_user()
            pickled_token = pickle.dumps(token)
            google_account = repository.save_google_account(user_to_db.id,pickled_token,google_account_id,user_email,first_name)
            query_string['message'] = 'Successful sign up'
            query_string['status_code'] = 200
            query_string['state'] = state
    elif state == 'login':
        token_for_authorization=flow.fetch_token(authorization_response=authorization_response)
        id_info = id_token.verify_oauth2_token(token_for_authorization['id_token'], request,app_config.CREDENTIALS_FILE['web']['client_id'])
        sub = id_info['sub']
        google_account = repository.query_google_account_by_google_id(sub)
        if google_account == None:
            logging.info("There is no user with that google_id")
            query_string['message'] = 'There is no user with that google_id'
            query_string['state'] = state
        else:
            token = google_account.credentials
            token = pickle.loads(token)
            token = check_if_token_expired(token,request,sub)
            query_string['status_code'] = 200
            query_string['state'] = state
```

**Slika 4.11.** *Callback* ruta

## 4.4. Sinkronizacija Google kalendara

Glavna funkcionalnost CalendarSync web aplikacije je sinkronizacija Google kalendara iz različitih profila. Rješenje je izrađeno u nekoliko faza:

1. Slanje postavki za sinkroniziranje
2. Traženje dodatnih informacija o računima za sinkroniziranje
3. Provjera dostupnosti traženih resursa,
4. Čitanje svih događaja s oba računa
5. Predprovjera događaja
6. Sortiranje događaja
7. Unos, brisanje i ažuriranje događaja

### 4.4.1. Slanje postavki za sinkroniziranje

Ovo je korak u kojem korisnik preko pristupnog dijela vrši interakciju s korisničkim sučeljem kako bi odabrao izvorišni i odredišni Google račun na kojem se nalaze kalendari, te izabrao postavke sinkroniziranja na način da u „*Sync setting*“ izborniku za ponuđene atribute odabere opcije koje će kopirati atribut iz izvorišnog kalendara u odredišni ili sakriti informacije tih atributa u odredišnom kalendaru. Nakon odabira svih potrebnih postavki za sinkroniziranje i pritiskom na gumb za sinkronizaciju pristupni dio aplikacije šalje zahtjev za kreiranje sinkronizacije u bazi kako bi se mogla iskoristiti nakon dodavanja kontinuirane sinkronizacije kalendara. Nakon kreiranja sinkronizacije u bazi šalje se zahtjev za sinkroniziranjem te postavke iz baze. Kako bi korisnik sinkronizirao dva računa isti trebaju biti povezani u aplikaciji prethodno.

### 4.4.2. Traženje dodatnih informacija o računima za sinkroniziranje

Nakon primitka zahtjeva za sinkronizaciju pozadinska aplikacija izvlači iz zahtjeva informacije o kojem se izvorišnom i odredišnom računu radi, te iste traži u bazi podataka. Ako se traženi računi ne nađu vrati se poruka s nastalom greškom, u suprotnom iz baze se dohvaćaju vjerodajnice koje se trebaju verificirati dali su još uvijek valjane ili ih treba osvježiti koristeći *Refresh token* koji zatraži od Google servisa novi token za pristupanje kalendarskim resursima.

```
def sync(sync_setting_id):
    message = None
    sync_setting = repository.get_sync_setting_by_id(sync_setting_id)
    if not sync_setting:
      return {'message': 'No matching sync setting id found'}
    google_account_source = repository.get_google_account_by_id(sync_setting.so
urce_id)
    google_account_target = repository.get_google_account_by_id(sync_setting.ta
rget_id)
```

**Slika 4.12.** Dohvaćanje podataka iz baze

### 4.4.3. Provjera dostupnosti traženih resursa

Potvrdom valjanosti vjerodajnica dohvaćenih iz baze podataka pozadinska aplikacija koristi klasu „*Credentials*“ koju koristi za konstruiranje vjerodajnica koje su potrebne za pristup resursima na Google Calendar API servisu. „*Credentials*“ vjerodajnice sastoje se od:

- „access_token“ token koji se koristi za pristupanje resursima
- „refresh_token“ token koji se koristi za osvježavanje isteklih tokena
- „id_token“ token koji nosi informacije o korisniku
- „token_uri“ adresa odakle se dohvaća token
- „client_id“ jedinstveni identifikator aplikacije koja želi pristupiti resursima
- „client_secret“ skrivena riječ potrebna za verificiranje aplikacije
- „scope“ lista potrebnih resursa koja će se zatražiti od Google servisa

```
    try:
      credentials_source = construct_credentilas(pickle.loads(google_account_so
urce.credentials))
      credentials_target = construct_credentilas(pickle.loads(google_account_ta
rget.credentials))
    except AttributeError:
      return {'message': 'Did not find google account '}
```

**Slika 4.13.** Provjera dostupnosti Google kalendara

### 4.4.4. Čitanje svih događaja s oba računa

Završetkom kreiranja vjerodajnica potrebnih za pristup Google Calendar resursim pozadinska aplikacija kreira objekt nad kojem se pozivaju funkcije za rad s Google kalendarima. Kreiranje

objekta se radi na način da se poziva funkcija *build* koja dolazi iz „googleapiclient“ biblioteke predaju joj se parametri:

- riječ koji opisuje traženu vrstu resursa, u ovom slučaju je to riječ „*calendar*“
- riječ koji predstavlja verziju aplikacijsko programskog sučelja
- prethodno kreirane vjerodajnice
- varijabla „*cache_discovery*“ koja nosi vrijednost istina ili laž, a služi za brzo dohvaćanjem podataka koji već postoje u sustavu negdje u pričuvnoj memoriji

nakon izvršenja opisane funkcije ona vraća objekt preko kojeg se pozivaju funkcije koje rade s Google resursom. U aplikaciji se čitanje događaja poziva preko objekta na način da se nad objektom pozovu funkcije „*events“* koja označuje da će se raditi operacija nad događajima, te se na nju nadoveže funkcija „*list“* koja ispisuje sve događaje ovisno o parametrima funkcije. U aplikaciji je definirano da se dohvate svi događaji primarnog kalendara koji se nalaze u vremenskom razdoblju od sedam dana nakon poziva funkcije.

```
def read_events(credentials):
    """Function for retriving events from specific account calendar

    Keyword arguments:
    credentials -- credentials saved in database
    """
    service = build('calendar', 'v3', credentials=credentials, cache_discovery=
False)
    now=datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%S.%fZ')
    NUMBER_OF_DAYS = 7
    seven_days_after=((datetime.now()+timedelta(days=NUMBER_OF_DAYS)).strftime(
'%Y-%m-%dT%H:%M:%S.%fZ') )
    events_result = service.events().list(calendarId='primary', timeMin=now,sho
wDeleted=False,
                                          timeMax=seven_days_after).execute()
    events = events_result.get('items', [])
    if not events:
      logging.info('No events to read, inside read_events function')
    return events
```

**Slika 4.14.** Dohvaćanje događaja iz kalendara

### 4.4.5. Predprovjera događaja

Nakon dohvaćanja događaja iz kalendara traženih resursa treba se utvrditi postoje lidogađaji koji bi se trebali kopirati u odredišni kalendar ili ne. Predprovjera vrši brzu provjeru događaja dohvaćenih iz kalendara. Brza Predprovjera obuhvaća:

- provjeru postoje li događaji u izvorišnom kalendaru, ako ne onda ne postoje događaji koji se trebaju ubaciti u odredišni račun te se napravi bilješka „*Nothing to insert*“.
- ako se u prvoj provjeri utvrdi da postoje događaji u izvornom kalendaru, onda se vrši postojanje provjera u odredišnom kalendaru, ako u odredišnom kalendaru ne postoje događaji znači da se mogu svi događaji iz izvorišnog kalendara kopirati u odredišni.

Ako se u oba kalendara nalaze događaji onda se mora izvršiti detaljna analiza događaja koji se trebaju ubaciti u odredišni kalendar.

```
    if not source_events:
      message = 'Nothing to insert'
    elif not target_events:
      insert_events(source_events, credentials_target)
      message = 'Inserting all events'
    else:
      events_to_insert, events_to_update, events_to_delete = same_events(source
_events, target_events, credentials_target)
      insert_events(events_to_insert, credentials_target)
      update_events(events_to_update, credentials_target)
      delete_events(events_to_delete, credentials_target)
      message = 'Accounts synced'
    return {'message': message}
```

**Slika 4.15.** Brza provjera događaja

### 4.4.6. Sortiranje događaja

Nakon potvrde da se u događaji nalaze u oba kalendara potrebna je detaljna analiza između dvije liste događaja kako bi se utvrdilo koji od tih događaja se trebaju kopirati u odredišni račun. U aplikaciji se to radilo tako da se obje liste događaja šalju funkciji koja izvršava algoritam za sortiranje događaja prema onima koje treba da:

- doda,
- ažurira,

- obriše

u odredišnom računu. Algoritam vrši još jednu podjelu događaja zato što u Google Calendar sustavu postoje tri različita tipa događaja:

1. Normalni događaj koji se dogodi samo jednom
2. Ponavljajući događaj koji ima pravila kada se treba ponoviti
3. Instanca ponavljajućeg događaja kojoj se može pomaknuti vrijeme ili se može potpuno otkazati

Nakon što su se događaji dodatno podijelili uspoređuju se s događajima koji se nalaze u odredišnom kalendaru gdje se na temelju istog identifikatora određuje dali se promijenilo neki od atributa kao na primjer: ime, vrijeme, opis. Nakon što se usporede atributi događaja zna se koja se radnja treba izvršiti za taj događaj, te se sukladno tome stavlja na listu koje predstavljaju određenu grupu događaja nad kojima se treba izvršiti gore navedene radnje.

```
def same_events(source_events, target_events, credentials_target):
    """ Function for comparing events from two accounts and returning
    events to insert and update to calendar

    Keyword arguments:
    source_events -- collection of events from account one calendar
    target_events -- collection of events from account two calendar
    """
    service = build('calendar', 'v3', credentials=credentials_target, cache_dis
covery=False)
    events_to_insert = []
    events_to_update = []
    events_to_delete = []
    for event_from_source_account in source_events:
      do_nothing_with_event = False
      for event_from_target_account in target_events:
          if(is_normal_event(event_from_source_account)):
            do_nothing_with_event = handle_normal_event(event_from_source_accou
nt, event_from_target_account, events_to_update, do_nothing_with_event)
          elif(is_recurring_event(event_from_source_account)):
            do_nothing_with_event = handle_recurring_event(event_from_source_ac
count, event_from_target_account, events_to_update, do_nothing_with_event)
          elif(is_instance(event_from_source_account)):
            do_nothing_with_event = handle_instance(event_from_source_account,
event_from_target_account, events_to_update, events_to_delete, do_nothing_with_
event, service)
      if (do_nothing_with_event == False):
        if(is_instance(event_from_source_account)):
          handle_instance_not_read_from_target(event_from_source_account, event
_from_target_account, events_to_update, events_to_delete)
        else:
          handle_new_event(event_from_source_account, events_to_insert)
    return events_to_insert, events_to_update, events_to_delete
```

**Slika 4.16.** Sortiranje događaja

### 4.4.7. Unos, brisanje i ažuriranje događaja

Nakon izvršene detaljne analize događaja između izvorišnog i odredišnog kalendara dobile su se liste koje sadrže grupe događaja koje predstavljaju podjelu događaja prema akciji koja se treba izvršiti. Tako da se za listu:

- „*events_to_insert*“ izvršava funkcija za dodavanje događaja s ove liste u odredišni kalendar,

- „*events_to_update*“ izvršava funkcija za ažuriranje događaja s ove liste u odredišnom kalendaru prema promjenama nastalim u izvornom kalendaru,
- „*events_to_delete*“ izvršava funkcija koja briše događaje s ove liste u odredišnom kalendaru prema otkazanim događajima u izvorišnom kalendaru.

Izvršavanje ovih funkcija radi se na sličan način kao čitanje svih događaja iz kalendara, definira se objekt koji komunicira s Google Calendar resursima, te se preko njega pozivaju funkcije i predaju parametri koji izvršavaju tražene akcije nad kalendarom. Slike 4.17., 4.18., 4.19. predstavljaju funkcije za unos, ažuriranje i brisanje događaja.

```
def insert_events(events, credentials):
    """Function for inserting events collected from events parameter

    Keyword arguments:
    events -- collection of events to be inserted in calendar
    credentials -- credentials saved in database
    """
    service = build('calendar', 'v3', credentials=credentials, cache_discovery=
False)
    batch = service.new_batch_http_request(callback=callback)
    for event in events:
      new_event = cococal_event(event)
      batch.add(service.events().insert(calendarId = 'primary',body=new_event))
    batch.execute()
```

**Slika 4.17.** Unos događaja

```
def update_events(events, credentials):
    """Function for updating events collected from events parameter

    Keyword arguments:
    events -- collection of events to be updated in calendar
    credentials -- credentials for specific google account saved in database
    """
    service = build('calendar', 'v3', credentials=credentials, cache_discovery=
False)
    batch = service.new_batch_http_request(callback=callback)
    for event in events:
      updated_event = cococal_event(event)
      batch.add(service.events().update(calendarId = 'primary', eventId = updat
ed_event['id'], body = updated_event))
    batch.execute()
```

**Slika 4.18.** Ažuriranje događaja

```
def delete_events(events, credentials):
    """ Function for deleting events collected from events paramater

    Keyword arguments:
    events -- collection of events to delete in calendar
    credentials -- credentials for specific google account saved in database
    """
    service = build('calendar', 'v3', credentials=credentials, cache_discovery=
False)
    batch = service.new_batch_http_request(callback=callback)
    for event in events:
      batch.add(service.events().delete(calendarId='primary', eventId= event['i
d']))
    batch.execute()
```

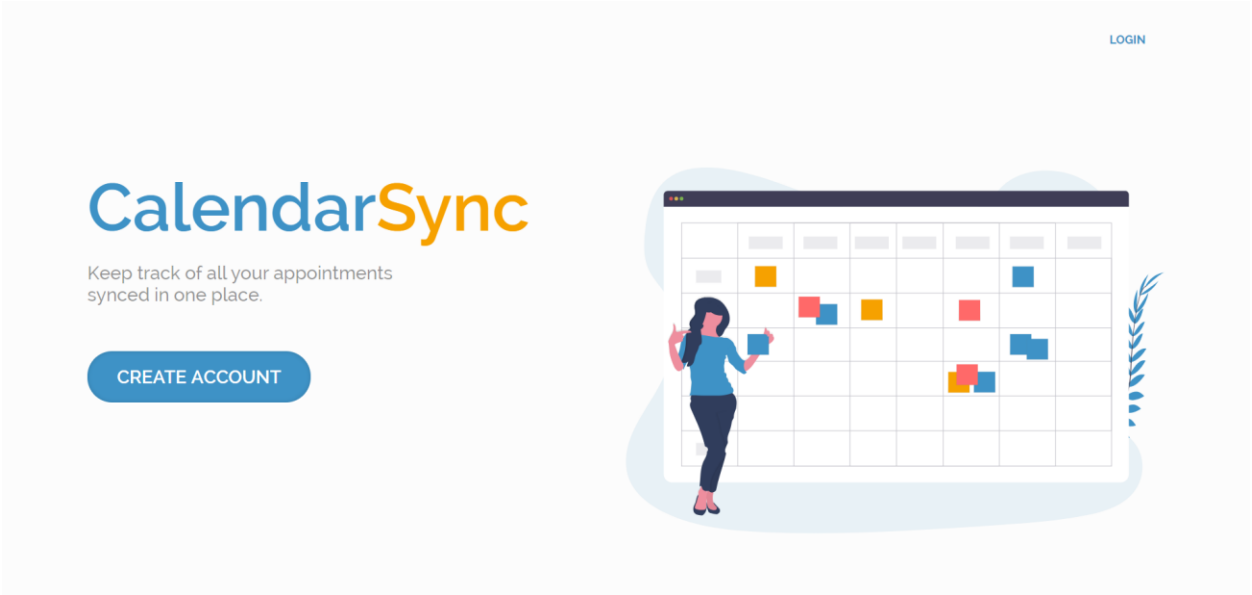
**Slika 4.19.** Brisanje događaja

# 5. KORISNIČKO SUČELJE I OPIS RADA APLIKACIJE

Korisničko iskustvo (engl. *user experience*) je jedan od najvažnijih dijelova aplikacije, ništa ne vrijedi ako se napravi aplikacija s puno mogućnosti, a korisnici neće znati kako doći do njih ili da ona uopće postoje. Danas je doba kad se razvoj aplikacija sve više provodi s „*agile*“ principom razvoja u svrhu dostavljanja proizvoda krajnjim potrošačima što prije, a to je rezultat sve veće konkurencije na tržištu. Iako se požuruju izdavanja aplikacija ne smije se zaboraviti na dizajn korisničkog sučelja [16].

## 5.1. Početna stranica

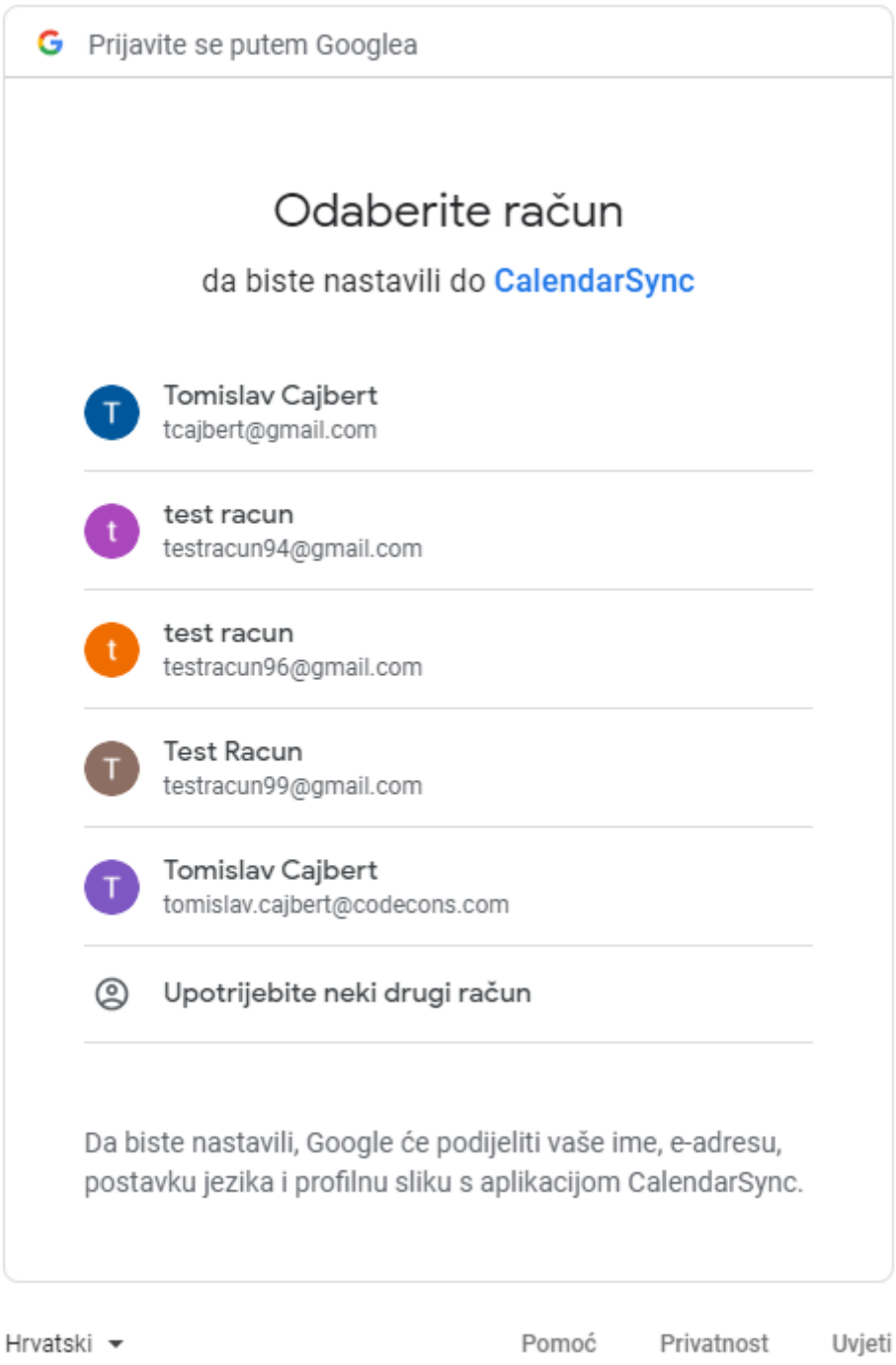
Početna stranica aplikacije je prikazana na slici 5.1. iz nje se može vidjeti jednostavan dizajn koji uključuje samo dva gumba. Jedan gumb služi za prijavu u sustav „LOGIN“, dok drugi za kreiranje računa u aplikaciji „CREATE ACCOUNT“




**Slika 5.1.** Početna stranica

## 5.2. Obrazac za prijavu

Pritiskom na gumb za prijavu ili kreiranje računa sustav vas vodi na Google obrazac za prijavu, tamo se korisniku mora ovjeriti autentičnost prije početka rada na aplikaciji CalendarSync. Obrazac na slici 5.2. se pojavljuje prilikom funkcija kreiranja računa, prijave u sustav, te povezivanja dodatnih računa.

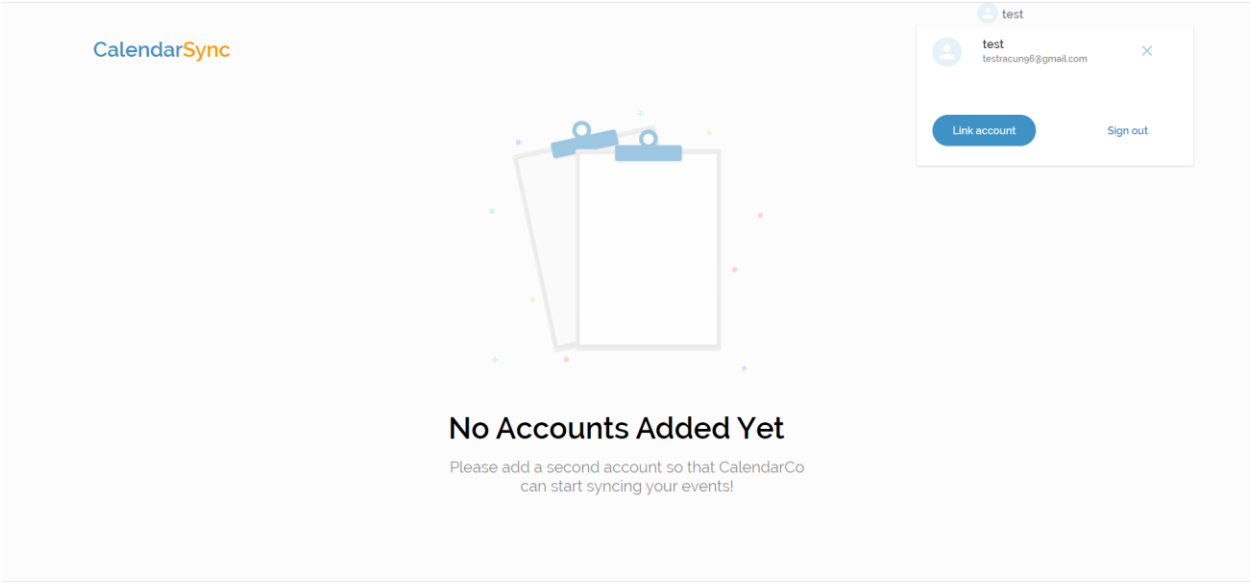



**Slika 5.2.** Google obrazac za autentifikaciju

## 5.3. Stranica za kontrolu sinkronizacija nakon prijave

Nakon uspješne prijave u sustav prikazuju se sučelja ovisno o broju spojenih računa na profil korisnika, ako nije spojen ni jedan dodatni Google račun na profil onda se prikazuje sučelje

prikazano na slici 5.3. koje označava da ne postoji kombinacija računa koja se može sinkronizirati, u suprotnom prikazuje se sučelje koje se nalazi na slici 5.4.



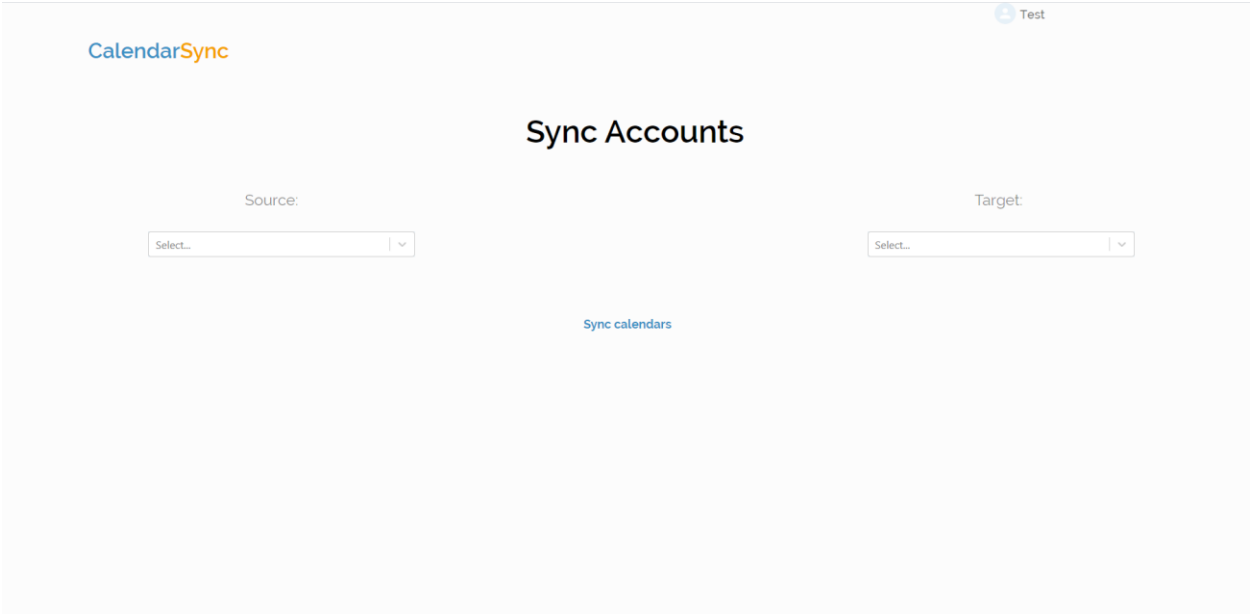

**Slika 5.3.** Stanica za kontrolu sinkronizacije prije spajanja dodatnih Google računa

Ovo korisničko sučelje služi kao interakcija korisnika s pozadinskim dijelom sustava koji se bavi sinkronizacijom i praćenjem povezanih Google računa na trenutno prijavljenog korisnika. Na ovom sučelju korisnik može:

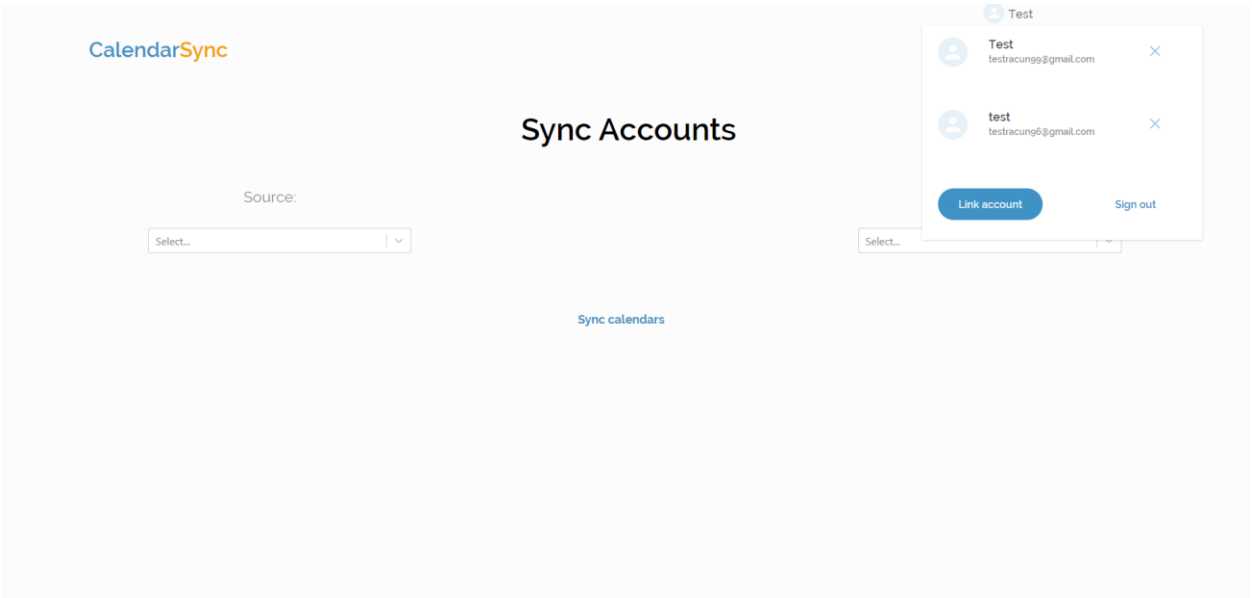
- otvoriti izbornik prikazan na slici 5.5. koji upravlja sa stanjem profila trenutnog korisnika, u njemu su prikazani:
    - svi trenutno povezani Google računi koji se mogu iskoristiti za sinkronizaciju događaja ili se klikom na „x“ mogu obrisati s profila trenutno prijavljenog korisnika
    - gumb „*Link account*“ koji predstavlja akciju spajanja novog Google računa na profil korisnika
    - gumb „*Sign out*“ koji predstavlja akciju odjavljivanja iz aplikacije za trenutno prijavljenog korisnika
- izabrati izvorišni račun i odredišni račun na kojem se nalaze kalendari koji se trebaju sinkronizirati prikazano na slici 5.6.
- izabrati gumb „*Sync calendars*“ koji otvara prozor za upravljanje postavkama sinkronizacije

Većinom će funkcija ove aplikacije služiti kako bi se iz više različitih kalendara prebacili događaji u jedan glavni kalendar, kako bi se ta funkcija odradila potrebno je za svaki kalendar iz kojeg se

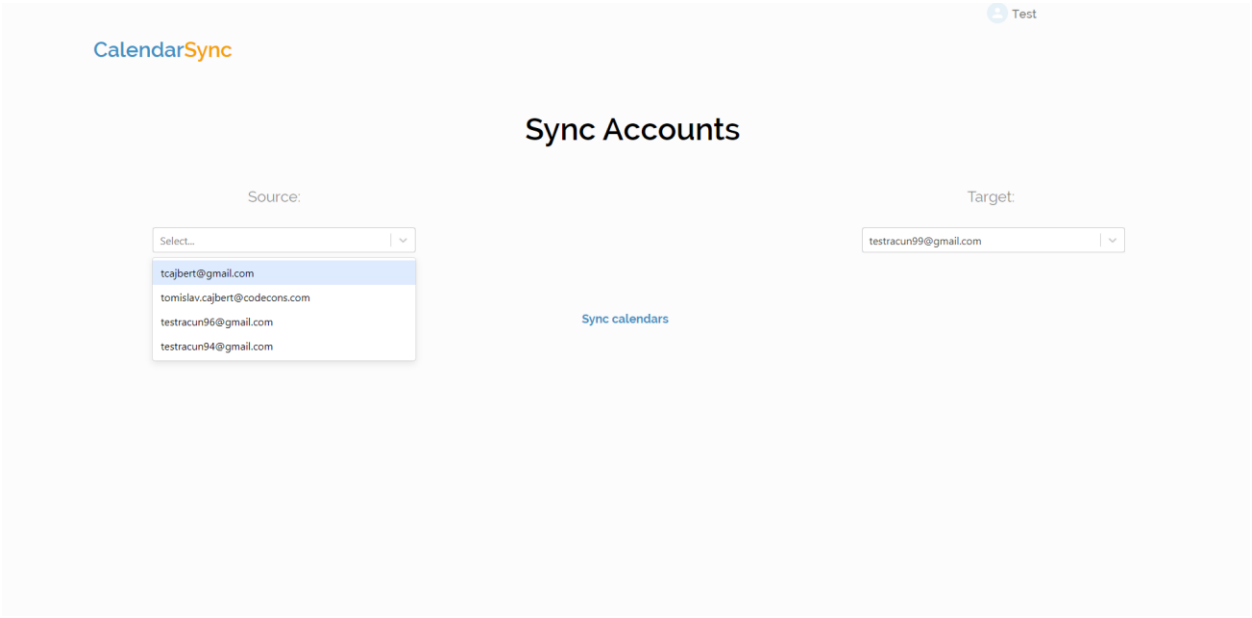
želi sinkronizirati izabrati ga kao izvorišni kalendar i odgovarajući odredišni kalendar te normalno pokrenuti sinkronizaciju.




**Slika 5.4.** Korisničko sučelje za kontrolu sinkronizacija



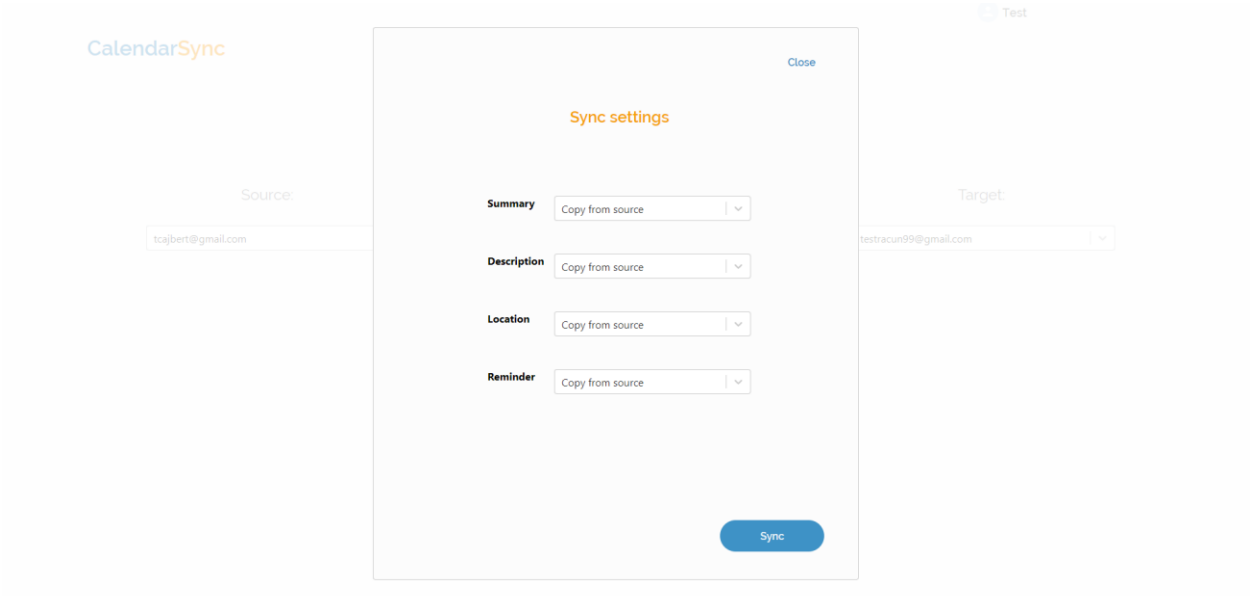

**Slika 5.5.** Izbornik profila




**Slika 5.6.** Izbronik računa za sinkronizaciju

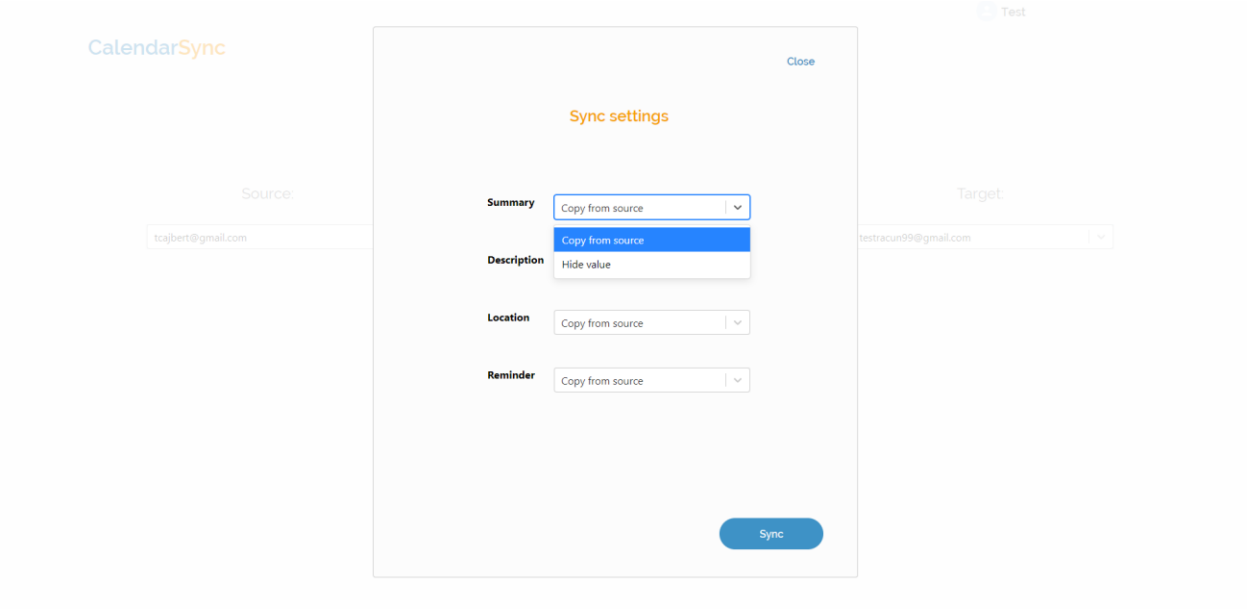
## 5.4. „Sync settings“ izbornik za anonimiziranje sinkronizacije

Pritiskom na gumb „*Sync calendars*“ na korisničkom sučelju za kontrolu sinkronizacija, otvara se „*Sync settings*“ prozor slika 5.6. u kojem se može upravljati atributima za sinkronizaciju, to se odnosi na pitanje dali se želi sačuvati vrijednost iz izvorišnog računa ili se želi anonimizirati da ljudi koji imaju pregled događaja na odredišnom kalendaru ne vide neke osjetljive informacije.




**Slika 5.7.** „Sync settings“ izbornik

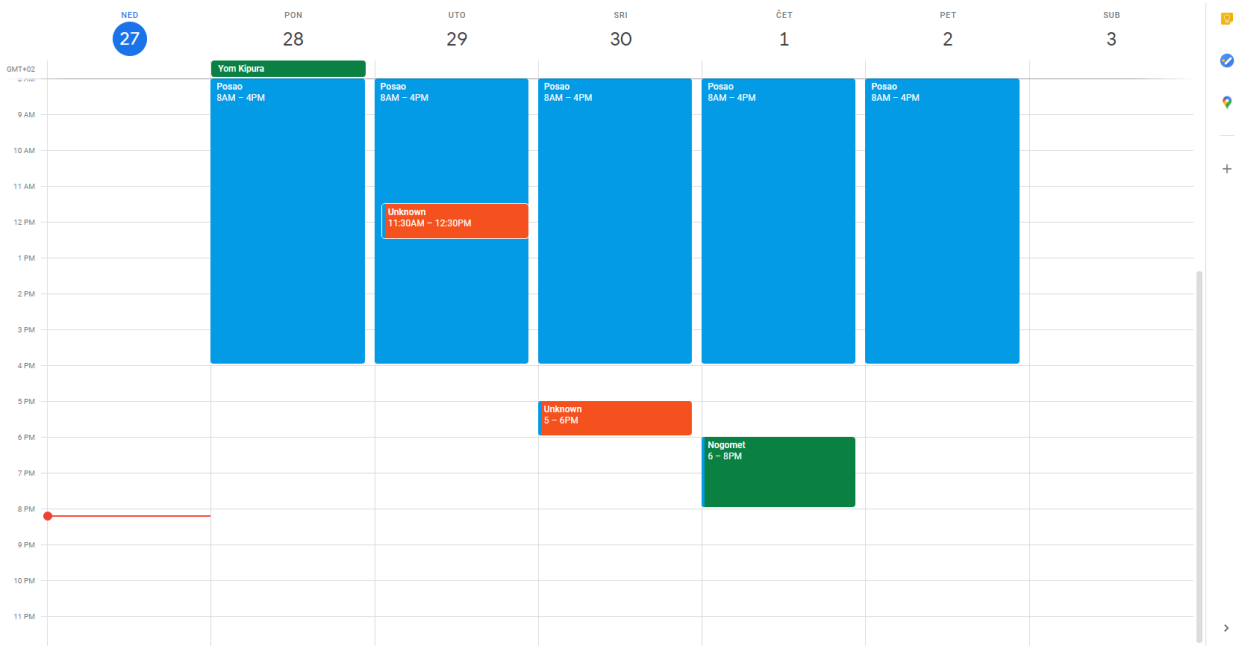
Izborom željenog atributa i klikom na padajući izbornik pored njega otvaraju se opcije „*Copy from source*“ koja kopira sadržaj tog atributa iz izvorišnog računa u odredišni i „*Hide value*“ koja vrijednost atributa iz odredišnog računa skriva pri kopiranju u odredišni slika 5.7.




**Slika 5.8.** Padajući izbornik u„“Sync settings“ prozoru

## 5.5. Izgled sinkroniziranog računa

Nakon uspješno izvršene sinkronizacije razlika u događajima koji su došli s nekog drugog računa i onih koji su već bili u kalendaru ne bi trebala biti prepoznatljiva, jedino ako se nije označila opcija „*Hide value*“ za „*summary*“ atribut onda će se taj događaj lako prepoznati po imenu koji je jednak „*Unknown*“. Na slici 5.9. može se vidjeti kalendar u kojeg su se sinkronizirala druga dva kalendara, sva tri kalendara koriste različite boje događaja tako da iz jednog računa imamo prikazano pet događaja zvana „Posao“ koja su obojena plavom bojom, iz drugog računa dva događaja s imenom „*Unknown*“ što znači da je sadržaj imena skriven prilikom sinkroniziranja, ti događaji su prikazani narančastom bojom, te iz trećeg računa imamo događaj s imenom „Nogomet“ koji je označen zelenom bojom.




**Slika 5.9.** Izgled kalendara nakon sinkronizacije

# 6. ZAKLJUČAK

Tijekom izrade rješenja za problem sinkronizacije događaja koji se nalaze u kalendarima na različitim računima vidljivo je da nije jednostavno izraditi jedno takvo rješenje iz razloga što se mora implementirati algoritam koji će proći kroz sve događaje koje je pročitao iz izvorišnog i odredišnog kalendara te profiltrirati ono što mora ubaciti u odredišni kalendar. Također tu dolazi i veliki problem sigurnosti osjetljivih informacija koje se trebaju povjeriti u ruke neprovjerenih trećih strana, najviše zato što se tu radi o korisnicima koji koriste kalendare u svrhu obavljanja poslovanja. Aplikacija je izrađena radi nedostatka postojećih rješenja i potrebom za sigurnim sustavom koji će to sve odraditi.

Prednosti aplikacije su te da se mogu na jednom mjestu sinkronizirati svi željeni kalendari kojima se ima pristup, te se pri sinkronizaciji mogu anonimizirati stvari koje se ne želi prikazati u odredišnom kalendaru.

Nedostatak rješenja opisanog u ovom radu je što nije izrađena kontinuirana sinkronizacija događaja, ta funkcija bi cijelu aplikaciju podigla na puno veću razinu iz razloga što bi se za kontinuiranu sinkronizaciju bilo potrebno prijaviti jednom u sustav, podesiti sve opcije sinkroniziranja i ostaviti da to sve radi do trenutka kad se pojavi potreba za promjenom tih postavki sinkroniziranja. Još jedna stvar koja bi se jako dobro uklopila u aplikaciju je dodavanje funkcije u postavkama sinkronizacije gdje bi se osim ponuđenih „*Copy from source*“ i „*Hide Value*“ funkcija, ubacila „*Enter custom value*“ koja bi omogućavala korisniku da direktno u aplikaciji izmjeni atribute koji će se kopirati, što bi bilo dobro za označavanje događaja s nekim identifikatorom da se zna s kojeg računa se taj događaj kopirao. Opisani nedostaci su jako dobre funkcionalnosti, te će ih se pokušati dodati u sustav u nekoj od sljedećih verzija aplikacije.

# LITERATURA

[1] Britannica – Calendar development, https://www.britannica.com/science/calendar, pristupljeno: rujan, 2020.

[2] Calendar - History of the calendar, https://www.calendar.com/history-of-the-calendar/, pristupljeno: rujan, 2020.

[3] SuperSaaS – Synchronization with other calendars, https://www.supersaas.com/info/doc/integration/synchronize, pristupljeno: rujan, 2020.

[4] Calendar – Google calendar guide, https://www.calendar.com/google-calendar-guide/, pristupljeno: rujan, 2020.

[5] Developers Google – Calendar API https://developers.google.com/calendar, pristupljeno: rujan, 2020.

[6] React – A javascript library for building user interfaces, https://reactjs.org/, pristupljeno: rujan, 2020.

[7] Learning React by Alex Banks and Eve Porcello, Published by O'Reilly Media, Inc. Printed in the United States of America. 2016

[8] Flask Web Development by Miguel Grinberg, Published by O'Reilly Media, Inc. Printed in the United States of America 2018

[9] PostgreSQL: Up and Running, Second Edition by Regina O. Obe and Leo S. Hsu, Published by O'Reilly Media, Inc. Printed in the United States of America. 2015

[10] Learning PostgreSQL Create, develop, and manage relational databases in real-world applications using PostgreSQLSalahaldin Juba, Achim Vannahme, Andrey Volkov. Published by Packt Publishing Ltd. Birmingham, UK. 2015

[11] Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux, Alessandro Del Sole. Apress, Berkeley, CA. 2019

[12] Flask Migrate – A database migration, https://flask-migrate.readthedocs.io/, pristupljeno: rujan, 2020.

[13] OAuth 2 in Action by Justin Richer Antonio Sanso. Published by Manning Publications. March. Shelter Island, NY. 2017

[14] Developers Google – OAuth 2.0 web server, https://developers.google.com/identity/protocols/oauth2/web-server#python, pristupljeno: rujan, 2020.

[15] Lean UX Applying Lean Principles to Improve User Experience by Jeff Gothelf. Published by O'Reilly Media, Inc. Printed in the United States of America. 2013

[16] UX for Lean Startups Faster, Smarter User Experience Research and Design by Laura Klein. Published by O'Reilly Media, Inc. Printed in the United States of America. 2013

# SAŽETAK

**Naslov:** Web servis za sinkronizaciju događaja iz više kalendara

U današnje vrijeme kada se povećava broj ljudi koji rade od kuće važno je da se što bolje organizira vrijeme. To se postiže planiranjem radnog dana koji se mora negdje zabilježiti kako bi ti događaji imali težinu, te kako bi se mogao staviti neki podsjetnik za nadolazeće događaje. U sklopu ovog diplomskog rada predstavljeno je rješenje za osobe koje imaju više računa na kojima prate svoje vrijeme, te imaju potrebu za dodavanjem događaja koji se nalaze u jednom kalendaru u neki drugi kalendar pri čemu imaju mogućnost da odrede hoće li za određeni atribut nekog događaja prekopirati vrijednost atributa ili ga sakriti. Produkt diplomskog rada je web aplikacija koja daje korisnicima mogućnost da se prijavom u aplikaciju putem svojih Google računa sinkroniziraju njihovi kalendari koji se nalaze na drugim povezanim računima.

**Ključne riječi**: Flask, React, PostgreSQL, Google Calendar, Sinkronizacija

## ABSTRACT

**Title:** Web service for synchronization of events from multiple calendars

Today, the number of people that work from home is increasing, so it is very important to organize time schedule. To achieve that, work time schedule must be carefully planned and marked down so those events can have meaning and reminders can be set on upcoming events. In the scope of this graduate thesis solution is presented for people who have multiple accounts on which they use to follow their schedule. Usually they also have a need for adding events from one of their accounts to the other ones on which they have the option to specify setting for copying the event attribute. They can choose from whether they want to copy attribute from the source or hide it. The product of this graduate thesis is a web application that allows its users to sign up in the application using their google accounts to synchronize calendars from across their linked accounts.



**Keywords:** Flask, React, PostgreSQL, Google Calendar, Synchronization

# ŽIVOTOPIS

Tomislav Cajbert rođen 08.05.1995 u Vinkovcima. U rujnu 2010. godine upisuje Ekonomska i trgovačka školu Ivana Domca u Vinkovcima, smjer ekonomist koju završava 2014. U istoj godini upisuje Preddiplomski stručni studij elektrotehnike, smjer informatika kojeg završava 2017 godine te upisuje razlikovnu godinu kako bi pokrio razliku predmeta potrebnih za upis sveučilišnog diplomskog studija kojeg upisuje 2018. godine na kojem se odlučuje za smjer programsko inženjerstvo. Početkom 2019. godine postaje stipendist Vukovarske firme CodeConsulting gdje steče iskustvo za rad nakon završetka studiranja.

# PRILOZI

[P1] CD s programskim kodom