

# Koncentrator plaćanja

---

- *Sistem kojim upravlja odvojeno preduzeće čija svrha je da posreduje između različitih korisnika i servisa za plaćanje*
- *Pruža usluge servisa za plaćanje koji obuhvataju : banku, PayPal, Bitcoin*
- *Ima API koji je prilagođen raznim sistemima, bez obzira na veličinu sistema*
- **Plagabilan**, što predstavlja mogućnost dodavanja novih načina plaćanja i pretplaćivanje novih sistema tako da se sistem ne gasi

## 1. Plagabilnost

Koncentrator plaćanja kao zaseban sistem treba da ima svoju web stranicu. Sistem podržava dve vrste korisnika : Administratore i obične korisnike. Obični korisnici mogu predstavljati prodavce ili druge sisteme koji se mogu pretplatiti na KP tako što će na sajtu kroz formu uneti sve potrebne podatke koji su potrebni sistemu uključujući i podatke o tome koje sve načine plaćanja pretplatnik želi da mu se omogući. Administrator proverava svaki zahtev za pretplatom i odobrava ga ili odbija u zavisnosti od ispravnosti podataka. Ukoliko administrator odobri pretplatnika, on će dobiti nalog na sistemu, moći će da se uloguje i vidi detalje svoje pretplate, kao i da zatraži novi način plaćanja da mu se omogući u sistemu ili da otkáže neki postojeći.

Takođe, administratoru je dozvoljeno dodavanje novih načina plaćanja. Ovom akcijom unutar sistema KP će se pozvati funkcija koja dinamički dodaje klase koje nam omogućavaju interakciju sa novonastalim načinom plaćanja. Funkcija će da kreira novi API koji će servisi moći da konzumiraju i novi *SERVIS* koji će nam pružiti akcije plaćanja.

## 2. Plaćanja

Kao što je pomenuto KP ima servise koje omogućuju različite načine plaćanja što sigurno obuhvata *Banku, PayPal* i *Bitcoin*.

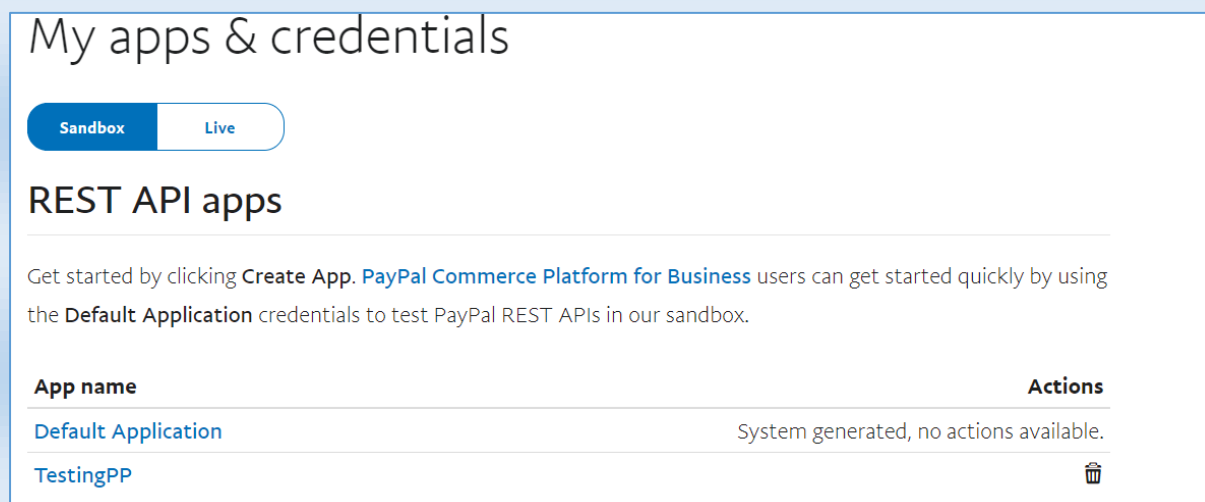
### 2.1 Plaćanje karticom

Kako bismo omogućili plaćanje preko banke potrebno je da kreiramo mikroservise za banku prodavca, banku kupca i centar za platne kartice. Plaćanje karticom podrazumeva komunikaciju između različitih učesnika.


Potrebno je da i kupac i prodavac imaju otvorene račune u banci. Ukoliko dva učesnika nemaju račune u istoj banci kao posrednika imamo centar za platne kartice koji učestvuje kao posrednik u komunikaciji između dveju banaka. Kupac kada odabere banku kao način plaćanja u obavezi je da unese sve potrebne podatke kako bi se inicirala transakcija. Kada se transakcija izvrši, mogući ishodi jesu uspešno, neuspešno i greška nakon čega se kupcu daje na znanje do kog ishoda je došlo.

## 2.2 Plaćanje PayPal-om

Kako bismo omogućili plaćanje putem *PayPal*-a imamo nekoliko preduslova. Prvi preduslov jeste da napravimo nalog na *PayPal*-u. Zatim kada se logujemo imamo opciju da napravimo aplikaciju koja će nam simulirati plaćanja i korisnike i takođe nam omogućava da koristimo *PayPal Api* preko kojeg ćemo moći da simuliramo akcije.



The screenshot shows the 'My apps & credentials' interface. At the top, there are two tabs: 'Sandbox' (selected) and 'Live'. Below the tabs, the heading 'REST API apps' is displayed. A text block explains that users can get started quickly by using the 'Default Application' credentials in the sandbox. Below this is a table with two columns: 'App name' and 'Actions'.

App name	Actions
Default Application	System generated, no actions available.
TestingPP	

Listing 1 - Prikaz kreiranih aplikacija

Na prikazanoj slici možemo da vidimo kreirane aplikacije. U našem slučaju *TestingPP* je aplikacija koju ćemo koristiti. Ukoliko je odabran **Sandbox** aplikacija neće izvršavati prava plaćanja nego samo simulirati, sve dok ne predjemo na **Live** opciju. Svaka aplikacija ima svoje kredencijale pomoću kojih možemo dobiti token koji nam omogućava konzumiranje *PayPal Api*-ja.

SANDBOX API CREDENTIALS

Sandbox account

sb-m2elv3870617@business.example.com

Client ID

AUoKNW\_Sj0z8cYq6N\_P33gzaxwUscgYfvIVovP\_35MvZdX3JPsqeczYH96m3Lh1ZslUGs7gfBOISjZfs

Secret

[Hide](#)

**Note:** When you generate a new secret, you still maintain the original secret. The maximum number of client secrets is two. A client secret is either in enabled or disabled state.

Created	Secret	Status	Action
Nov 25, 2020	EM_OZBI_Bg--4EYY11aZnuFGcfBp8dm4n2xV-1pJtsEA9pPw5wTTPSMfRmt2tCpwqrs2IVUxsNnDd4d	Enabled	...

Ilustracija 2 - Kredencijali kreirane aplikacije

Poslednji preduslov jeste da napravimo lažne naloge pomoć kojih ćemo da vršimo plaćanja. Takvi nalozi se zovu **Sandbox test accounts**. Moguće je imati personal ili bussiness naloge, gde se najčešće bussiness koriste u slučaju prodavca, a personal u slučaju kupca. Na svakom nalogu se nalazi određena suma novca koja nam je dostupna.

Sandbox Accounts:

Create bulk accounts

Create account

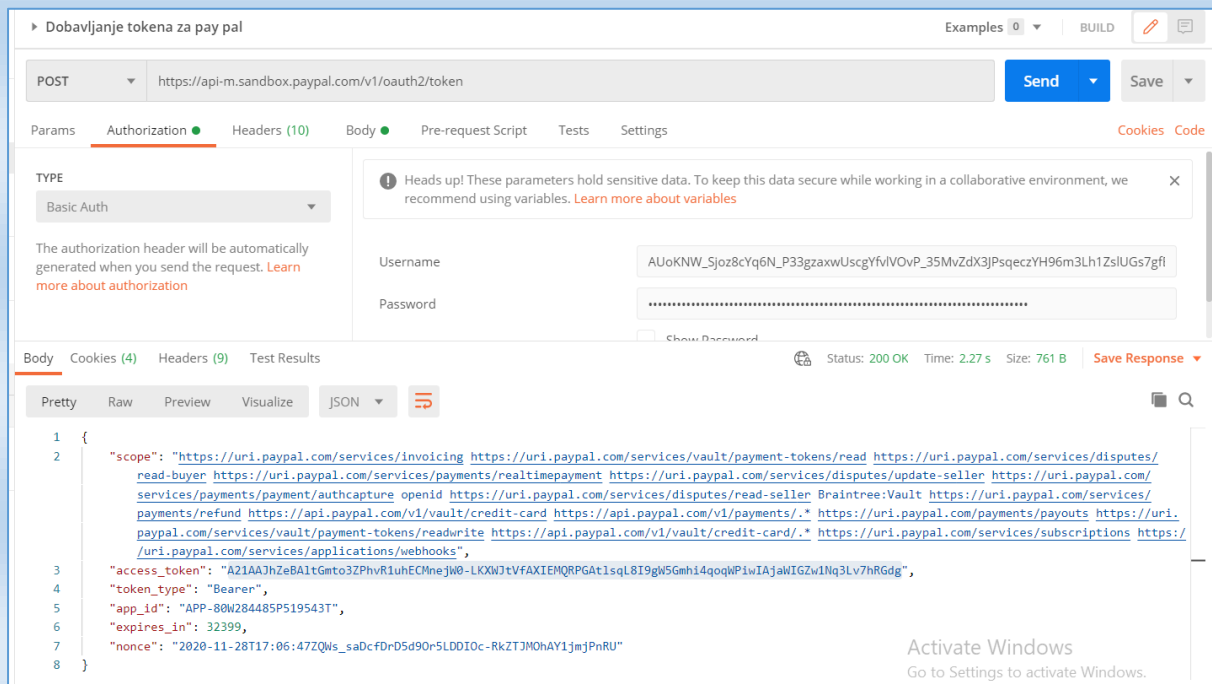
Total Accounts: 2

<input type="checkbox"/> Account name	Type	Country	Date created	Manage accounts
<input type="checkbox"/> sb-43youk3866166@personal.example.com <div>DEFAULT</div>	Personal	US	25 Nov 2020	...
<input type="checkbox"/> sb-m2elv3870617@business.example.com <div>DEFAULT</div>	Business	US	25 Nov 2020	...

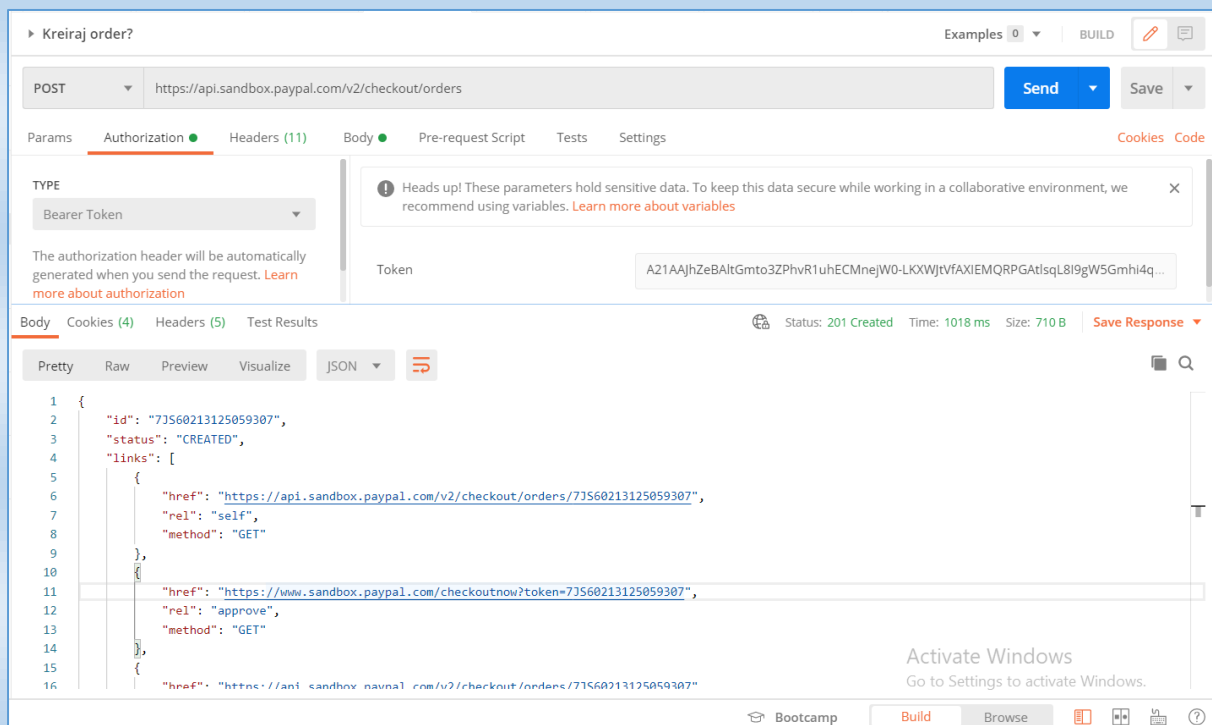
Ilustracija 3 - Prikaz Sandbox naloga

Nakon svih ovih podešavanja moguće je konzumirati *Api* kako bismo radili konkretne narudžbine. U nastavku će biti prikazane akcije dobavljanja tokena

koji je potreban za komunikaciju sa *PayPal* Api-jem i kreiranje jedne porudžbine korišćenjem programa **Postman**. Prilikom dobavljanja tokena potrebno je uneti kredencijale aplikacije (u našem slučaju *TestingPP*), a u slučaju nardžbine dobavljeni token i kredencijale *Sandbox* naloga.



Ilustracija 4 - Dobavljanje tokena



Ilustracija 5 - Kreiranje narudžbine

## 2.3 Plaćanje Bitcoin-om

Za potrebe plaćanja putem *Bitcoina*-a koristićemo **bitcoinj**, koji predstavlja java implementaciju bitcoin protokola. Pomoću njega moguće je održavati *wallet*, slati i primati transakcije.

*Wallet* predstavlja program u kome se čuvaju bitcoini. Za svakog korisnika koji ima određena sredstva u svom *wallet*-u postoji privatni ključ koji se odnosi na adresu *wallet*-a. *Wallet*-i pružaju mogućnost slanja i primanja bitcoina i daju vlasništvo korisniku.

Kako bismo iskoristili bitcoinj unutar projekta potrebno je ubaciti *dependency*-je koji nam to omogućavaju.

```
<dependency>
  <groupId>org.bitcoinj</groupId>
  <artifactId>bitcoinj-core</artifactId>
  <version>0.15</version>
  <scope>compile</scope>
</dependency>

<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava</artifactId>
  <version>27.1-jre</version>
</dependency>
```

Ilustracija 6 – Dependencies

Zatim je potrebno da konfiguriramo određene *Bean*-ove koje će *wallet* da koristi. Jedan od njih je *NetworkParametar* koji nam definiše u kom režimu ćemo raditi. U našem slučaju to je testing. Kako bismo pojednostavili konfigurisanje različitih instanci potrebnih za *wallet* možemo koristiti *WalletAppKit* koji nam sve to spakuje odjendom.

@Configuration

```
public class Config {
    @Value("${bitcoin.network}")
    private String network;
    @Value("${bitcoin.file-prefix}")
    private String filePrefix;
    @Value("${bitcoin.file-location}")
    private String btcFileLocation;
    private NetworkParameters networkParameters;
    public Config() {
        BriefLogFormatter.init();
    }
}
```

```

@Bean
public NetworkParameters networkParameters() {
    if(network.equals("testnet")) {
        return TestNet3Params.get();
    } else if(network.equals("regtest")) {
        return RegTestParams.get();
    } else {
        return MainNetParams.get();
    }
}

@Bean
public WalletAppKit walletAppKit(@Autowired NetworkParameters networkParameters) {
    return new WalletAppKit(networkParameters, new File(btcFileLocation), filePrefix) {
        @Override
        protected void onSetupCompleted() {
            if (wallet().getKeyChainGroupSize() < 1) {
                wallet().importKey(new ECKey());
            }
        }
    };
}
}

```

### Ilustracija 7 – Konfiguracioni fajl

Nakon toga kreiramo *wallet*. Nakon pokretanja aplikacije i izvršene metode za kreiranje *wallet*-a dobićemo njegovu adresu. Kako bismo primali novac potrebno je registrovati *listener* koji će pratiti događaje i primiti sadržaj. Ukoliko želimo da šaljemo novac potrebno je da definišemo sumu i kome šaljemo, takođe uslov je da imamo dovoljno sredstava na našem računu kako bismo uspešno izvršili plaćanje.

```

@Component
public class MyWallet {
    @Autowired
    private WalletAppKit walletAppKit;
    @Autowired
    private NetworkParameters networkParameters;
    @PostConstruct
    public void start() {
        walletAppKit.startAsync();
        walletAppKit.awaitRunning();
        walletAppKit.wallet().addCoinsReceivedEventListener(
            (wallet, tx, prevBalance, newBalance) -> {
                Coin value = tx.getValueSentToMe(wallet);
            }
        );
    }
}

```

```

        System.out.println("Received tx for " + value.toFriendlyString());
        Futures.addCallback(tx.getConfidence().getDepthFuture(1),
            new FutureCallback<TransactionConfidence>() {
                @Override
                public void onSuccess(TransactionConfidence result) {
                    System.out.println("Received tx " +
                        value.toFriendlyString() + " is confirmed. ");
                }
                @Override
                public void onFailure(Throwable t) {}
            }, MoreExecutors.directExecutor());
    });
    Address sendToAddress = LegacyAddress.fromKey(networkParameters,
walletAppKit.wallet().currentReceiveKey());
    System.out.println("Send coins to: " + sendToAddress);
}
public void send(String value, String to) {
    try {
        Address toAddress = LegacyAddress.fromBase58(networkParameters, to);
        SendRequest sendRequest = SendRequest.to(toAddress, Coin.parseCoin(value));
        sendRequest.feePerKb = Coin.parseCoin("0.0005");
        Wallet.SendResult sendResult =
walletAppKit.wallet().sendCoins(walletAppKit.peerGroup(), sendRequest);
        sendResult.broadcastComplete.addListener(() ->
            System.out.println("Sent coins onwards! Transaction hash is " +
sendResult.tx.getTxId()),
            MoreExecutors.directExecutor());
    } catch (InsufficientMoneyException e) {
        throw new RuntimeException(e);
    }
}
}
}

```

### *Ilustracija 8 – Implementacija Wallet-a*