Maksemooduli arhitektuur

Sissejuhatus

Eesmärk

Dokumendi eesmärk on kirjeldada maksemooduli rakenduse tehniline arhitektuur. Maksemoodul on tugisüsteem maksete tegemiseks väliste maksevahendajate kaudu. Maksemoodul omab avaliku portaali ja API-t ja sisemist ametniku rakendust ja API-t.

Viited

Angular JS - https://angular.io/
RESTful arhitektuuristiil - http://en.wikipedia.org/wiki/Representational_state_transfer
JSON andmeformaat - http://en.wikipedia.org/wiki/JSON
PostgreSQL - https://en.wikipedia.org/wiki/PostgreSQL

Mõisted / lühendid

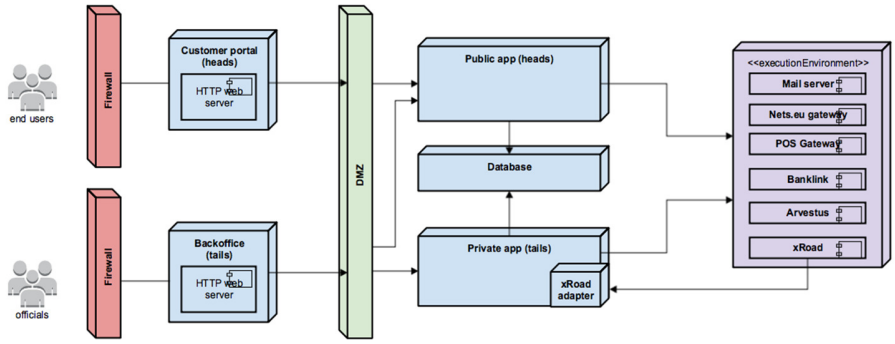
Mõiste / lühend	**Selgitus**
Maksemoodul	Maksemoodul on makserakendus, mis võimaldab vahendada maksed kaupmeeste ja maksevahendajate vahel, kasutades REST ja/või SOAP API ning portaalrakendust
Maksemooduli avalik rakendus	Maksemooduli avalik rakendus pakub teenuseid, mis ei vaja autentimist. Avalikult kättesaadav API ei sisalda delikaatset informatsiooni.
Maksemooduli ametniku rakendus	Maksemooduli ametniku rakendusega on võimalik saada ülevaadet kõikidest tellimustest ja maksetest ning konfigureerida süsteemi
Maksevahendaja	Süsteem, mis pakub maksemoodulile liidest maksete sooritamiseks, näiteks: pangalink, nets.eu
Maksemooduli klient ehk kaupmees	Maksemooduliga liidestatav süsteem, mis kasutab maksemooduli teenused
Tollikassa, srkassa, cash, sularahakassa	Erinevates kontekstides viidatakse antud nimetustega käesolevale süsteemile - tollikassale.

ORM	Object Relational Mapping - tehnoloogia, mis võimaldab abstraheerida objektorienteeritud ning relatsioonilise andmekihi ning nende sidumiseks kasutatavad teenused/operatsioonid.

Kontseptuaalne arhitektuur, kihid ning arhitektuurimustrid

Maksemooduli rakendused kasutavad klassikalist kihilist arhitektuuri. Tehnoloogilises plaanis on kihilisus realiseeritud kahe tehnoloogiliselt sõltumatu ning teineteisest eraldiseisva paigalduskomponendina:

- presentatsiooni- ehk esitluskihti realiseerivast Javascript ökosüsteemil baseeruvast Angular.io frontend rakendusest kasutaja lehitsejas
- SOA põhimõttel RESTful arhitektuuristiilis esitatud ressurss-orienteeritud teenuseid eksponeerivast klassikalisest Java rakendusest, mis koondab kõik ülejäänud klassikalised kihid (teenus, DAO, integratsioon ning ressursikihi adapterid).



Maksemooduli FE (frontend rakendused)

Esitluskiht

Maksemooduli Frontend süsteemi esitluskiht on realiseeritud Angular4 (angular.io) raamistikul realiseeritud SPA (single page application) põhimõttel töötava javascript rakendusena. Angular raamistikul baseeruvate lahenduste näol on tegu lähenemisega, kus esitluskihi realiseerimisel kasutatav klassikaline MVC arhitektuuri-/disainimuster ja selles sisalduvad abstraktsioonid (mudel, kontroller, vaade) on tervenisti realiseeritud kasutaja lehitsejas täidetava Javascripti lahendusena. Selle tulemusena on kogu esitluskihi formeerimiseks kasutatav loogika (kontrollerid), vaated (kasutaja ekraanide vaated HTML dokumendimudelina ning nende loomine ja muutmine) ning neid siduv mudel (andmed ja nende pärimine tagatoa süsteemist) implementeeritud Javascript tehnoloogias ning komponendi täitekeskkonnaks (runtime-environment) on kasutaja lehitseja ise. Sellise lähenemise peamiste eelistena võrreldes

alternatiivse serveri-poolsete Java tehnoloogial baseeruvate MVC raamistikega saab välja tuua järgmised asjaolud:

- esitluskiht on täielikult realiseeritud lehitseja poolt toetatud tehnoloogiates (javascript, html) ning rakenduse koodi täitmine toimub täielikult kliendi lehitsejas. Seeläbi on tagatud võimalus luua maksimaalselt kasutajakogemusele suunatud esitluskihi liideseid.
- kasutja lehitseja ning serveri vaheline andmevahetuse hõlmab "puhtal kujul" andmeid ilma neid ümbritseva esitlusspetsiifilise märgendita (html). See läbi on serveri ja lehitseja andmevahetus maksimaalselt efektiivne (liiguvad üksnes andmed ilma vormingut defineeriva märgendita) ning on tagatud parem vastutusjaotus erinevate komponentide/kihtide vahel (server eksponeerib andmeid/teenuseid, mille kasutus on võimalik ka väljaspool FE rakendust ennast ning nende andmete esitlus defineeritakse kasutaja lehitsejas endas täidetava FE rakenduse poolt).
- Angualr JS rakendus on põhimõttelt ühe-lehekülje-lahendus (nö. single-page-application), mis muudab kasutajakogemuse tunduvalt intuitiivsemaks.
- Lahendus on suurema tõenäosusega adaptiivsem tulevikus aset leidvatele tehnloogilistele trendidele. Järjest enam populariseeruv Javascripti ökosüsteem ja seal aset leidvad tehnoloogilised arengud garanteerivad parimal võimalikult viisil esitluskihi edasiarendatavuse (nii uute kui ka olemasolevate vajaduste näol), mis esitluskihi parima edasiarendatavuse tulevikus.

Angular4 rakendused realiseeritakse TypeScript keeles, mis on nn. superset keel Javascriptil, lisades viimasega võrreldes sellele tüübi- ning objekt-orienteeritud toe. Seeläbi on paremini tagatav koodibaasi kvaliteet ning hallatavus.

Maksemooduli BE (backend rakendused)

Maksemooduli teenuskiht (serverside backend) on klassikaline Java kihiline lahendus. Realisatsioon baseerub Java 8 versioonil (viimane stabiilne versioon arenduse teostamise hetkel). Arhitektuuri baasraamistikuna kasutatakse Spring versioon 4 raamistikku ning Spring tuumikmoodulite poolt rakenduse konteksti loomiseks kasutatavaid põhiomadusi:

- IoC (*Inverse Of Control container* ehk *Dependency Injection*) - võimaldab rakenduses esinevate tarkvaraliste komponente omavahelisi sõltuvusi paindlikult hallata ning tagab komponentide omavahelise sõltuvuste mugava hallatavuse ning testitavuse.
- AoP (Aspect Orientation Programming tugi) - AOP lähenemine võimaldab tsentraalselt kirjeldada ning väga vähese vaevaga rakendada süsteemi horisontaalselt läbivaid aspekte (näiteks transaktsioonid, logimine, *security* jms.).

Rakenduse baaslahendusena kasutatakse omakorda Spring raamistikku koos Spring tuumikmoodulitele (core moodulid) kasutatakse erinevate funktsionaalsete kihtide siseselt mitmesuguseid Spring laiendusmooduleid, millele rajanevad erinevate rakenduse arhitektuursete kihtide poolt pakutavad funktsioonid ning nendel põhinevad disainimustrid.
Maksemooduli rakenduse poolt eksponeeritud teenuskiht funktsioneerib olekuvabalt (stateless, st. serveri poolel ei säilitata ühe päring-vastus ahela üleselt kasutaja või tema tegevuse olekut, mis on eelduseks mõne järgmise antud kasutaja poolt tehtava päring-vastus ahela edukaks teostamiseks).

Teisisõnu on kogu kasutussessiooni põhine kontekst hoitud eranditult kasutaja lehitsejas ning backend's vajaminev kontekst saadakse alati väljakutsutava teenuse sisendina (vastena, et selle olemasolu eeldatakse eelnevalt initsialiseeritud Http - server-side sessioonist) või on rakendus võimeline selle puudumisel vähese kuluga seda päringu enda konteksti piires initsialiseerima.
Alljärgnevates jaotistes on välja toodud Maksemooduli rakenduses eristatavad klassikalised arhitektuursed kihid, nendes kasutatavad disainimustrid ja nende lahendamiseks kasutatavad tarkvaralised komponendid/teegid.

Esitluskiht

Tollikassa BE lahenduse esitluskiht eksponeerib RESTFul arhitektuuristiilis defineeritud teenuskihi (API), mida Maksemooduli FE lahendus (või mõni muu kolmas komponent tulevikus) oma (ekraani)komponentide loogika tarvis tarbib. Sisuliselt on Maksemooduli BE lahenduse esitluskiht Jersey JAX-RS REST kontrolleritena realiseeritud fassaad Maksemooduli teenuskihile.

Teenuskiht

Teenuskihi eesmärk on kapseldada süsteemi äriloogika. Iga teenus koosneb rangelt teenust kirjeldavast teenusesignatuuridefinitsioonist (Interface) ning seda realiseerivast POJO (*Plain Old Java Object*) stiilis realisatsiooniklassist. Teenuse signatuuridefinitsioonide range eristamine implementatsioonist tagab ühemõttelise teenuskihi definitsiooni ning loob eelduse erinevaid komponente hõlpsasti testida, võimaldades komponentide sõltuvusi testi otstarbel jäljendada/mockida (ühik- ning integratsioonitestid).

Teenuskihi piirimail rakendatakse ka sellised olulised arhitektuurilised aspektid nagu:

- transaktsioonid - erinevatele teenustele on võimalik deklaratiivselt defineerida teenuse SLA'le vastav transaktsionaalne kontekst.
- audit logi - iga teenuskihi kutse logitakse vastavalt RMIT mittefunktsionaalsetele nõuetele
- autoriseerimine - milline kasutaja millist ressursi on volitatud nägema / modifitseerima.

Teenuskihi väljundite defineerimisel on kasutatud DTO (*data trsansfer object*) disainimustrit. Sagedasti kasutatavatele ning harva muutuvatele andmetele realiseeritud teenuste puhul rakendatakse teenuskihi piiril ehCache puhverdust, mis võimaldab süsteemi ressursikasutust tuntavalt optimeerida.

Andme- ehk DAO kiht

Kõik Maksemooduli süsteemi poolt PostgreSQL andmebaasi tehtavad pöördumised on koondatud eraldiseivasse loogilisse kihti - DAO (Data Access Object) kihti. DAO kihi baastehnoloogiana kasutatakse Petit, mis omakorda kasutab Spring JDBC templatingut, kuna rakenduse andmedomeen on suhteliselt õhuke ning sellest tulenevalt on ORM (Object Relational Mapping) vahendite kasutegur võrreldes nende kasutusele võtuga suhteliselt väike. Kui süsteemi detailsem analüüs toob välja argumentatsiooni mõne ORM vahendi kasutusele võtuks, siis eelistame valikuna kergekaalulist ORM lahendust Petit (kõrvutatuna näiteks klassikalise Hibernate ORM lahendusega).

Integratsioonikiht

Integratsioonikihi moodustavad Maksemooduli süsteemi erinevad adaptermoodulid, mis pakkuvad teenused välistele süsteemidele või tarbivad teenused välistest süsteemidest.

Maksemooduli poolt tarbitavateks välisteks süsteemideks on täna:

- Maksevahendajad
  - Pangalingi iPizza protokolli toetavad pangad
  - Nets.eu kaardikeskuse lahendus kreeditkaartide jaoks
  - Swedpanga makseterminaalite lahendus kaardimaksete jaoks
- xTee andmekogud makseteavituste edastamiseks
- Officials - EMTA ametnike ning organisatsiooniüksuste andmed.
- MTA õigussüsteem
- Classificators - klassifikaatorid (codelist2)
- MTA arveldussüsteem

Integratsioon nimetatud süsteemidega on realiseeritud läbi üldkasutatavate pistikmoodulite (nn. service proxy).

Maksemoodul pakkub teenused makse registreerimiseks, teostamiseks, pärimiseks ja teavitamiseks.

Arhitektuuri olulisemad põhimõtted ning nende lahendamiseks kasutatud mehhanismid

RESTFul stiilis defineeritud teenusvaade

Maksemooduli BE poolt eksponeeritav ning FE mooduli poolt tarbitav teenusvaade on loodud RESTFul (Representional state transfer) arhitektuuristiili järgides. RESTFul teenuste poolt koostatud sõnumite andmeedastustehnoloogiana kasutatakse JSON (http://en.wikipedia.org/wiki/JSON) andmeformaati. Maksemooduli teenuskihi poolt sessioonifassaadi kihile tagastatud DTO andmeobjektide ning veebiteenustes kasutatava JSON formaati vahelised teisendused toimuvad Jackson teegi vahendusel (http://jackson.codehaus.org/).

BE logilahendus

Logilahendus on realiseeritud SLF4J (http://www.slf4j.org/) adapteri vahendusel, mis võimaldab abstraheerida rakenduse poolt konkreetse logilahenduse implementatsioonina kasutatava lahenduse (log4j1/log4j2/logback/commons-logging vms. alternatiiv). Seeläbi on logiraamitikuna kasutatav raamistik koodibaasist maksimaalselt sõltumatu ning vajadusel hõlpsasti välja vahetatav. Logilahendusena (millele SL4J fassaad logikäsklused delegeerib) kasutatakse hetkel logback (http://logback.qos.ch/) logiraamistikku, mis on edasiarendus/laiendus populaarsest log4j

logiraamistkust lahendamaks mõningaid log4j raamistiku kitsaskohti/puudujääke (http://logback.qos.ch/reasonsToSwitch.html).

Autentimine / autoriseerimine

Rakendus kasutab EMTA üldist turvaraamistikku, mille raames lahendatakse nii SSO (autentimine) kui ka privileegide kontroll (autoriseerimine). Autoriseerimine toimub nii teenus- kui esitluskihis.

Liidesed väliste süsteemidega

Vahendajate liidesed

Liidesed vahendajatega realiseeritud vahendaja poolt etteantud prokollide alusel, näiteks pangalingide puhul iPizza ja makseterminaalide puhul PosXml protokollide alusel. Andmevahetuse kord vastab vahendja ettekirjutusele.

MKR liidesed arvelduse ning isiku abstraktsioonidele

Maksemooduli süsteemi poolt pakutav funktsionaalsus hõlmab ka liideseid MKR registri ning arveldusmooduliga. Liidesed realiseeritud RESTful teenustena.

EMTA Officials

Töötajate ning organisatsiooniüksuste infot pakub EMTA infoarhitektuuris Officials komponent. Nimetatud komponendi poolt pakutavaid Maksemooduli süsteemi poolt tarbitavaid teenuseid hakatakse tarbima üle loodava proxy adapter komponendi, mille ülesanne on nimetatud süsteemide poolt pakutavaid teenuseid universaalselt klientsüsteemidele eksponeerida ning vajadusel arendus/testkeskkonna jaoks mockida.

EMTA Classificators

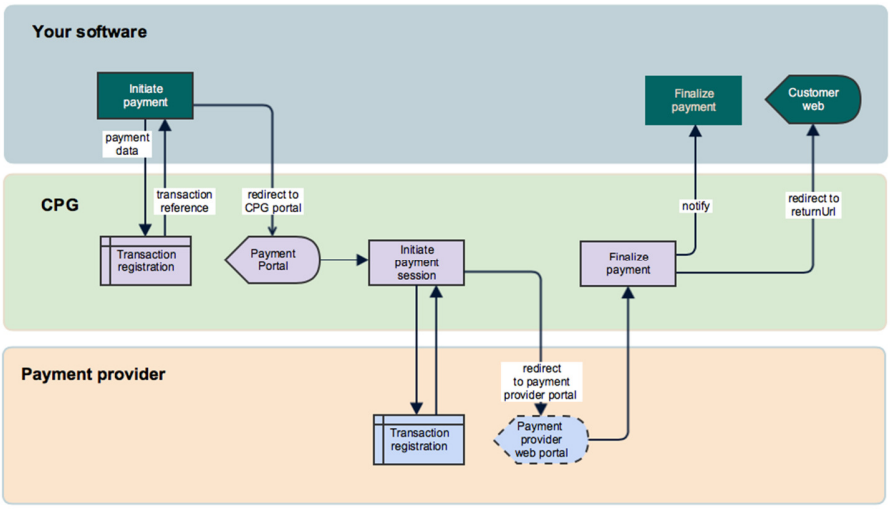
Klassifikaatorite infot pakub EMTA infoarhitektuuris Classificators komponent. Nimetatud komponendi poolt pakutavaid Maksemooduli süsteemi poolt tarbitavaid teenuseid hakatakse tarbima üle loodava proxy adapter komponendi, mille ülesanne on nimetatud süsteemide poolt pakutavaid teenuseid universaalselt klientsüsteemidele eksponeerida ning vajadusel arendus/testkeskkonna jaoks mockida.

Pakkutavad teenused

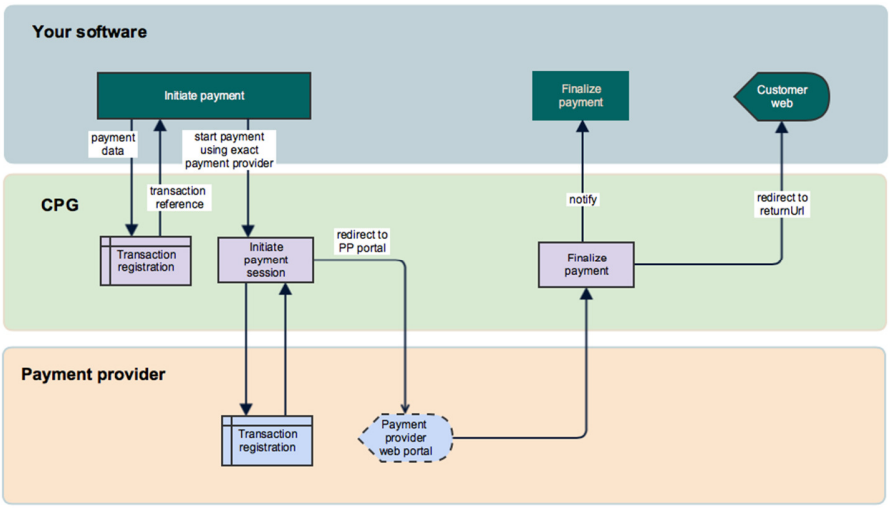
Maksemoodul pakkub makse algatamise, pärimise, tühistamise teenused nii xTee SOAP protokolli kaudu, kui ka HTTP REST kaudu.

Kaupmehe integratsioon avaliku Maksemooduli portaaliga

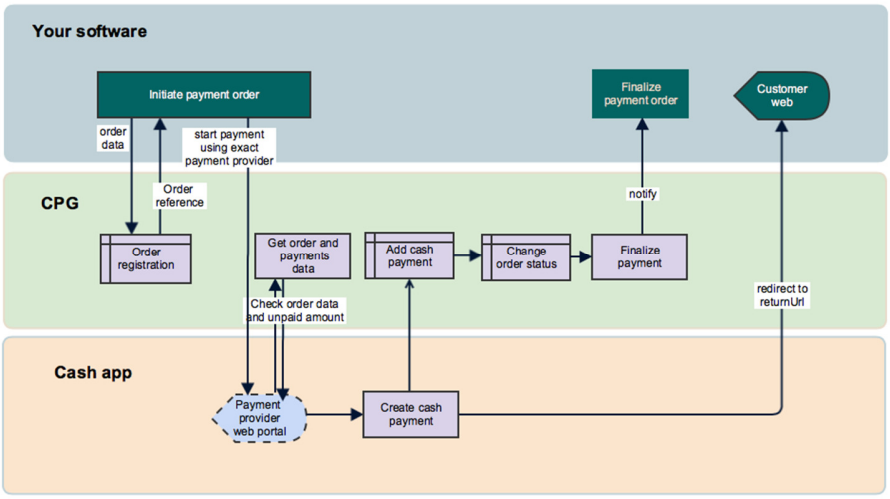
Kus maksevahendajateks võivad olla pangad (pangalink), kaardikeskus (nets.eu), makseterminaalid (POS terminal), sularaha maksed kassas.



Kaupmehe integratsioon Maksemooduli ning makseviiside integreerimine oma infosüsteemi



Kaupmehe integratsioon kassa ja maksemooduli kaudu



Maksemooduli ja Kaupmehe infosüsteemi koostoime