KKS arhitektuur

Sissejuhatus

Eesmärk

Dokumendi eesmärk on kirjeldada KKS rakenduse tehnilist arhitektuuri. KKS-i eesmärkideks on:

a) võimaldada operatiivset järelevalvet aktsiisilaost või tollilaost tarbimisse lubatud kütuse liikumise üle jälgides kütuse ostu-müügitehinguid;
b) võimaldada jälgida tolli järelevalve alla jäetud aktsiisikauba kasutamist, mis kasutamise eesmärgist johtuvalt on saanud maksusoodustuse, st. integreerida praegu toimiva JVIS süsteemi funktsionaalsus.
c) koguda ja edastada aktsiisisilaost või tollilaost tarbimisse lubatud biokütuse kohta spetsiifilisi andmeid, mida on vaja BioKütuse Seiresüsteem (**BKSS**).

Rakendus omab nii avalikku kliendirakendust kui sisemiseks tööks vajalikku ametnikurakendust.

Viited

React JS - https://facebook.github.io/react/
RESTful arhitektuur - 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
KKS	KKS on rakendus, mis võimaldab kütusemüüjatel sisestada kütusekoguseid ja jälgida nende liikumist, kasutades portaalrakendust
Avalik rakendus	KKS avalik rakendus pakub teenuseid kütusekäitlejatele kütuse liikumise registreerimiseks
Ametniku rakendus	KKS ametniku rakendusega on võimalik teostada kütuse liikumise järelvalvet kütusekäitlejate üleselt

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

Arhitektuur

KKS rakenduse puhul on kasutatud kihilist arhitektuuri, mis eristab erineva funktsionaalsusega kihid üksteisest. Rakendus ise on jaotatud füüsiliselt kaheks: Javascriptis realiseeritud ReactJS frontend ning Javas realiseeritud backend rakendus. Kasutajaliidese ja backend rakenduse vahel toimub suhtlus üle REST-i teenuste, kasutades JSON-i andmeobjekte.

Frontend

Frontend rakendus on realiseeritud niinimetatud üheleherakendusena (SPA - single page application). Antud lähenemise käigus luuakse rakendus, mis laetakse esimese laadimisega kasutaja brauserisse ning edasise kasutuse käigus toimub vaid kuvatavate andmete pärimine backend rakendusest. Kasutatavuse seisukohast on tegu intuitiivsema lähenemisega kuna kasutaja ei pea liikuma lehtede vahel, vaid kogu tegevus toimub ühes vaates.

Backend

Backend rakendus on realiseeritud Java8 rakendusena, kasutades Spring raamistikku. 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. Backend kihis on realiseeritud klassikalisest MVC arhitektuurist esitlus-, teenus- ja andmekiht.

Esitluskiht

Esitluskiht on realiseeritud Springi REST controlleritena, mis tarbivad ja väljastavad JSON andmeobjekte. Antud lähenemine võimaldab pakkuda rakenduse esitluskihti kui teenuseid ka teistele rakendustele. Esitluskiht kutsub äriloogika täitmiseks välja teenuskihis implementeeritud teenuseid. Esitluskiht kasutab rakenduse moodulis domain kirjeldatud objekte kasutaja poolt edastatavate andmete mappimiseks objektideks, mis siis omakorda edastatakse teenuskihti.

Teenuskiht

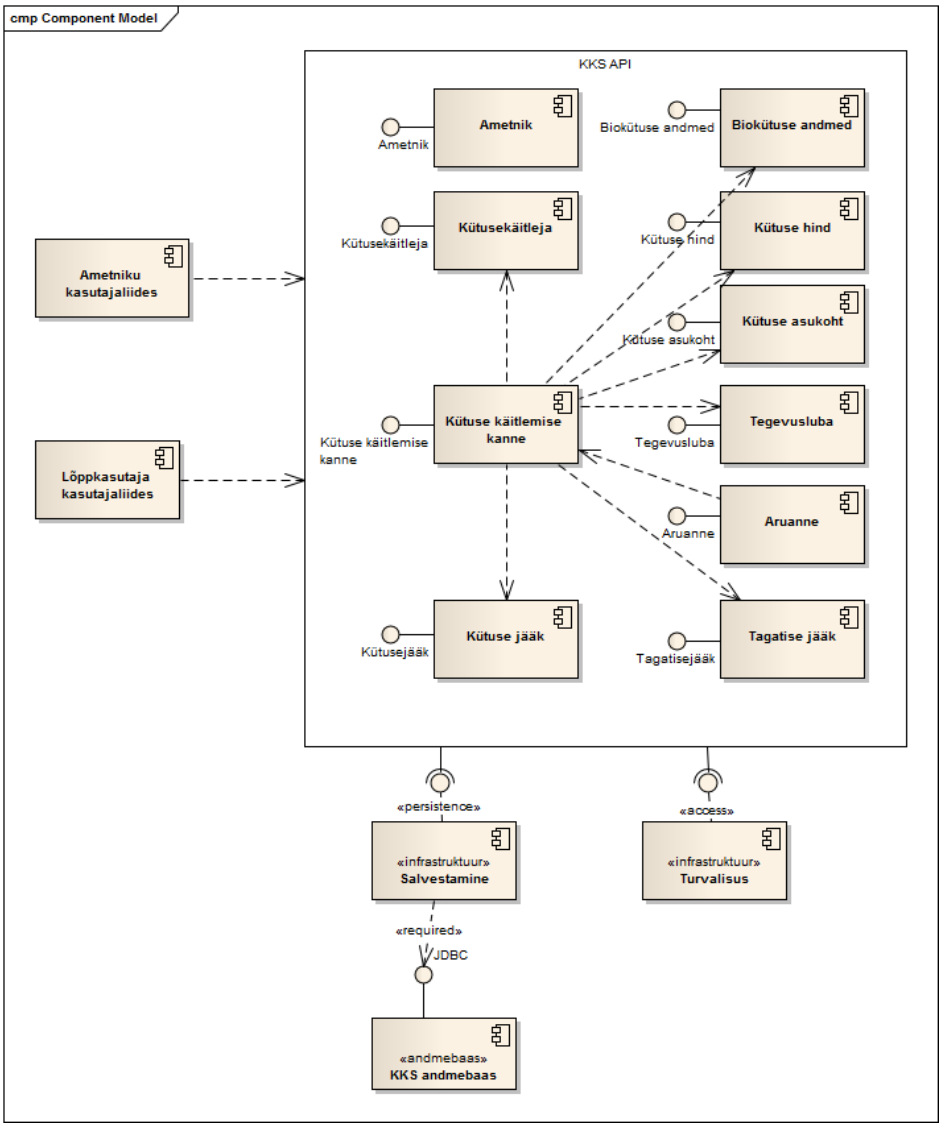
Teenuskihis paikneb kogu rakenduse äriloogika, mille poole pöördub kas esitluskiht või mõni teine teenuskihis asuv teenus. Teenuskihis on lahendatud transaktsionaalsus, auditeerimine ja autoriseerimine.

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.

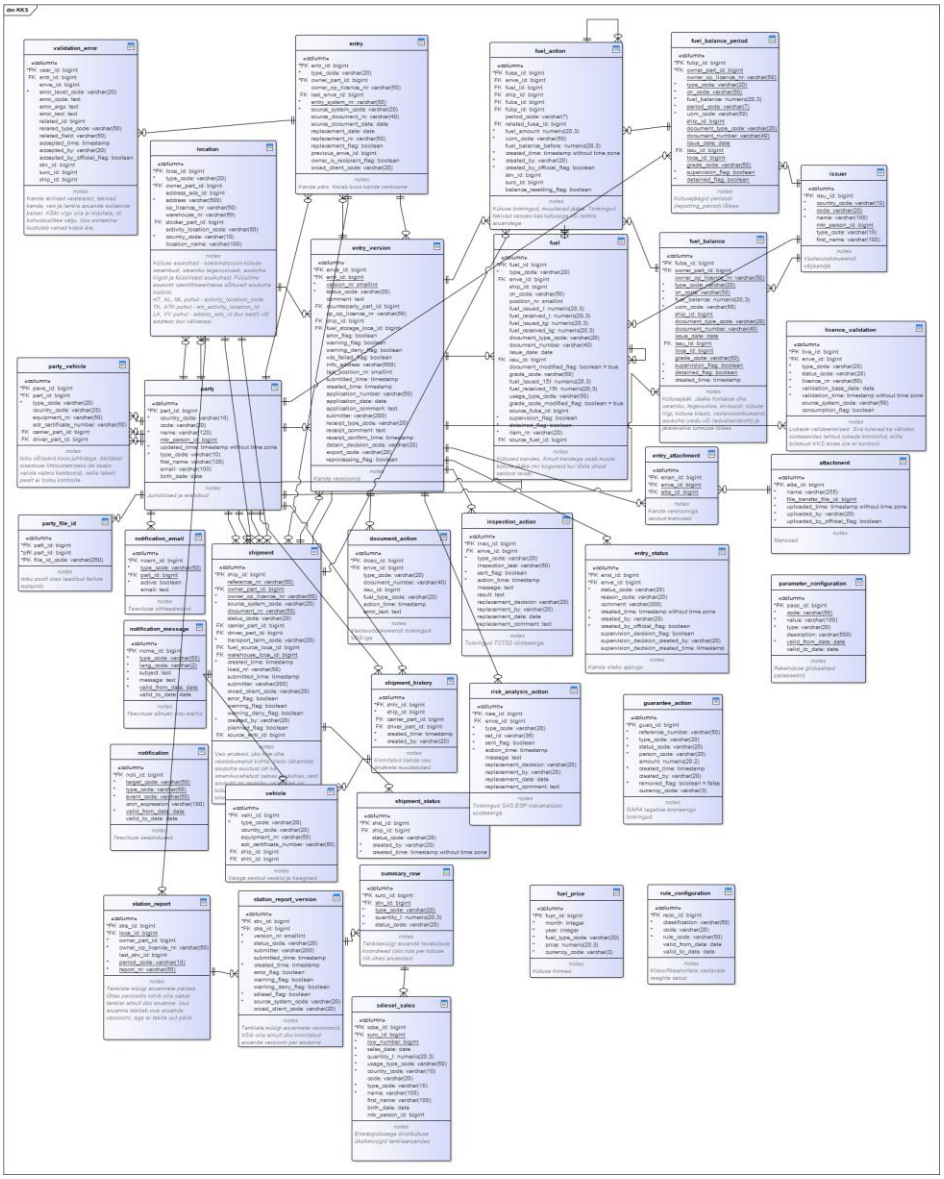
Teenuskihi väljundite defineerimisel on kasutatud DTO (*data trsansfer object*) disainimustrit. Sagedasti kasutatavatele ning harva muutuvatele andmetele realiseeritud teenuste puhul rakendatakse teenuskihi piiril Hazelcast puhverdust, mis võimaldab süsteemi ressursikasutust tuntavalt optimeerida.

Komponentdiagramm



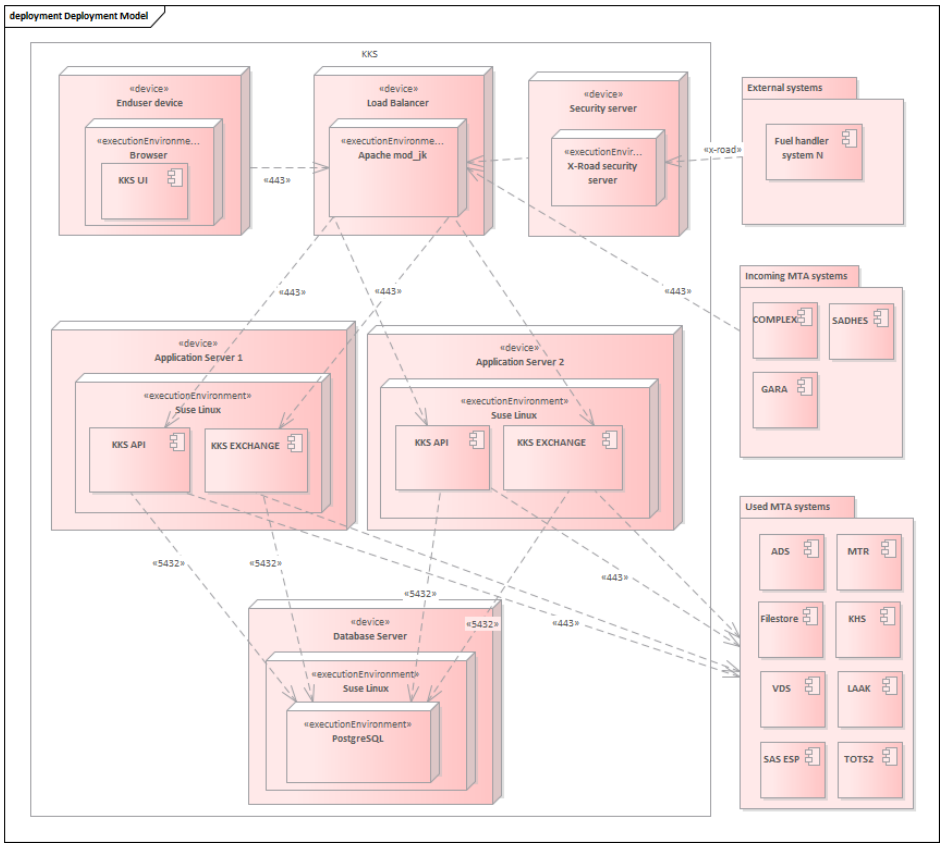
Joonis 1 KKS komponentdiagramm

Andmemudel



Joonis 2 KKS andmemudel

Paigaldusvaade



Joonis 3 KKS paigaldusvaade