

Hanke eseme tehniline kirjeldus

1. Üldsätted

- 1.1. Riigihanke esemeks on Eesti majanduse tegevusalade klassifikaatori infosüsteemile (edaspidi EMTAK IS) hooldus- ja lisaarendustööde teostamine pakkuja poolt (edaspidiselt Täitja), vastavalt hankija (edaspidiselt Tellija) poolt esitatud tellimustele.
- 1.2. Hankeleping sõlmitakse ühe Täitjaga 48 kuuks või kuni hanke kuni hankemahu rahvusvahelise piirmäära täitumiseni (138 999 eurot käibemaksuta). Hanke eeldatav maksumus on 60 000 eurot (käibemaksuta), kuid Tellija jätab endale õiguse tellida hooldus- ja arendustöid vastavalt tekkivale vajadusele kuni maksumuseni 138 999 eurot.
- 1.3. EMTAK IS on Justiitsministeeriumi haldusala Registrate ja Infosüsteemide Keskuse (edaspidiselt RIK) halduses asuv infosüsteem. EMTAK IS koosneb erinevatest failidest ja andmetest, mis on välja toodud füüsilise andmemudeli dokumentatsioonis. Täpsemat tehnilist dokumentatsiooni infosüsteemi kohta on võimalik saada RIHAst aadressil <https://riha.eesti.ee> või tellija käest.
- 1.4. EMTAK IS ISKE turvaklass on K2T1S1.
- 1.5. Hankelepingu alusel tellimuste esitamise protseduur ja tööde sisu on toodud käesolevas tehnilises kirjelduses ja hankelepingu projektis.
- 1.6. Täitjal tuleb dokumenteerida tellitud hooldus- ja lisaarendustööd vastavalt hankelepingu projektis olevale dokumendiplaanile. Dokumentatsiooniga tuleb tagada administreerimise juhendi, kasutusjuhendid ja RIHA'le ehk Riigi infosüsteemi haldussüsteemile vajalikul kujul teostatud dokumentatsioon. RIHA kohta leiab lähemat teavet aadressilt www.ria.ee/riha.

2. EMTAK IS tehniline spetsifikatsioon

- 2.1. OS: CentOS Linux release 7.7 (Core) x86_64
- 2.2. Tomcat: Apache-Tomcat v9.0.31
- 2.3. Java: v1.8.0_65 64bit Oracle
- 2.4. PostgreSQL v10

3. Mittefunktsionaalsed nõuded

- 3.1. Hooldustööd ja lisaarendused peavad vastama Lisa 4-2. Nõuded arendusele v6.0 nõuetele.
- 3.2. Lisaarenduse teostamisel (sh hinnapakumuse koostamisel) tuleb lähtuda mittefunktsionaalsetest nõuetest nii palju kui olemasoleva süsteemi kontekst seda lubab, st tööde teostamisel ei tohi tekkida ebamõistlike kulusid.
- 3.3. Lisaarendus ei tohi muuta kasutuskõlbmatuks infosüsteemi teisi funktsionaalsusi või tekitada vajadust (sh lisakulusid) teiste funktsionaalsuste parendamiseks, välja arvatud juhul, kui tellija on sellega nõustunud.

4. Hooldustööd

4.1. Hooldustööde ese:

- 4.1.1. Hooldustööd jagunevad konsultatsioonitoeks ja arendustoeiks.
- 4.1.2. Konsultatsioonitugi seisneb Täitja personali poolt pakutavas kaugtoes Infosüsteemi rakendustes tekkida võivate probleemide suhtes (serveri tõrked, kiirusprobleemid, programmivead jms). Konsultatsioonitoe puhul ei tee Täitja muudatusi serveri(te) konfiguratsioonis või Infosüsteemi rakendus(t)e koodis vaid juhendab nende tegemises Tellijat.
- 4.1.3. Arendustugi seisneb Infosüsteemis ilmnevate vigade Täitja poolt lokaliseerimises. Juhul, kui esinenud viga on garantiiline ning tegemist on veaga, mille Täitja on kohustatud omal kulul kõrvaldama, asub Täitja nimetatut viivitamatult parandama, vastavalt ette antud garantiitingimustele. Juhul, kui vea näol ei ole tegemist garantiilise veaga või konkreetse vea garantii korras parandamise eest ei vastuta Täitja või tegemist ei ole veaga vaid infosüsteemi muutmise/täiendamise vajadusega vmt, esitab Täitja Tellijale e-maili vahendusel teavituse nimetatust. Tellijal on õigus esitada vea parandamiseks vajamineva arendustegevuse teostamiseks Täitjale tellimuse, vastavalt lisaarenduste tellimise korrale.

- 4.1.4. Hooldustöö alla ei kuulu Lepingu täitmiseks vajalik tavapärane suhtlus Tellija ja Täitja kontaktisikute / esindajate vmt vahel.

4.2. Tellimise protseduur

- 4.2.1. Hooldustööde teostamiseks esitab Tellija kontaktisik Täitja kontaktisikule e-posti kaudu vabas vormis kirjutatud tellimuse, milles sätestab, kas soovitakse konsultatsiooni- või arendustuge ning muud tellimuse teostamiseks vajalikud asjaolud. Täitjal peab olema valmisolek tellimuse läbirääkimiseks pidada koosolekuid RIK'i ruumides 2 tööpäevase etteteatamisega Tellija poolt vastavalt vajadusele.
- 4.2.2. Hooldustööd tuleb teostada tellimuses toodud tähtajaks või poolte vahel kokkulepitud muuks tähtajaks
- 4.2.3. Täitjal tuleb arvestada, et teatud juhtudel on võimalik, et Tellija esitab tellimusi, mille täitmist loetakse ajakriitiliseks. Nimetatud juhul eeldab Tellija tellimuste täitmist maksimaalselt 10 tööpäeva jooksul. Juhul, kui Täitja ei suuda tagada tellimuste teostamist 10 tööpäeva jooksul on Tellijal õigus tellida vajalikke hooldustöid mujalt.

4.3. Hooldustööde mahu arvestus

- 4.3.1. Hooldustööde üle arvestuse pidamine on Täitja kohustus ning toimub töötundide alusel. Tööde mahtu arvestatakse 30 minuti täpsusega, ümardades vajadusel lähema arvestusühikuni (täis- või pooltund).
- 4.3.2. Täitja kohustub esitama Tellija kontaktisikule üks kord kuus kirjalikku taasesitamist võimaldavas vormis aruande teostatud hooldustööde kohta.
- 4.3.3. Tellijal on õigus esitada Täitjale pretensioon aruande õigsuse kohta 5 tööpäeva jooksul arvates aruande esitamiseks. Pretensioon lahendatakse Poolte vahel läbirääkimiste teel. Juhul, kui pretensiooni ei esitata või on esitatud pretensioon lahendatud Poolte vahel on Täitjal õigus esitada Tellijale arve teostatud hooldustööde kohta. Arve esitamisel tuleb järgida Lepingu üldtingimustes toodut.

5. Lisaarendused

5.1. Lisaarenduste ese

- 5.1.1. Lisaarendustena käsitletakse käesoleva Lepingu raames EMTAK infosüsteemi lisaarendusi.

5.2. Tellimise protseduur

- 5.2.1. Tellijal on õigus tellida Täitjalt infosüsteemi lisaarendusi. Lisaarenduste tellimine toimub tellimuste esitamise teel. Tellimused esitatakse Tellija kontaktisiku poolt Täitja kontaktisikule. Tellimus peab sisaldab vähemalt järgnevat:
- 5.2.1.1. kõiki sisulisi andmeid, mis on vajalikud pakkumuse koostamiseks;
 - 5.2.1.2. pakkumuse eeldatavat maksumust koos käibemaksuga (eurodes);
 - 5.2.1.3. pakkumuse esitamise tähtaega;
 - 5.2.1.4. välisrahastuse puhul rahastuse allikas, märkides tellimusse järgneva teksti: „Võttes arvesse projekti „.....“, mille kohaselt on vastava projekti toetuste saaja“ ja muud põhjendatud tingimused, mis on eeltingimuseks toetuse saamisele (nt. Euroopa Komisjoni poolt rahastatud projekti puhul arvel vajalik viide projektile vmt);
- 5.2.2. Täitja esitab Tellijale hiljemalt 5 tööpäeva jooksul arvates tellimuse edastamisest pakkumuse. Pakkumus peab sisaldama:
- 5.2.2.1. kõiki vajalikke andmeid, mis tellimuses nõutud;
 - 5.2.2.2. juhul, kui tellimuses nõutud, tuleb pakkumuses esitada ka lähteülesandes kirjeldatud funktsionaalsuse detailseks analüüsiks, realiseerimiseks, testimiseks ning dokumenteerimiseks kuluvat aega arendustundides; infosüsteemi riistvaralistest komponentides vajalike muudatuste kirjeldust; Infosüsteemiga seotud infosüsteemide Tööks vajalike tarkvarakomponentide täienduste kirjeldust;
 - 5.2.2.3. ajahinnangut kalendripäevades pakkumuse aktsepteerimisest Tellijale Tööde üleandmiseni.

- 5.3. Tellijal on õigus hinnapakkumine aktsepteerida või sellest loobuda.

- 5.4. Lisaarenduste (eel)analüüsiks ja pakkumise koostamiseks kulunud aega ei käsitleta hooldustööna ja nimetatut ei maksustata.
- 5.5. Tellimuse täitja alustab Tellimuse täitmist, kui Tellija on andnud kirjalikku taasesitamist võimaldavas vormis kinnituse pakkumuse sobivuse kohta.
- 5.6. Täitjal peab olema valmisolek vajadusel töötada ilma lisatasu või kompensatsioonita Registrite ja Infosüsteemide Keskuse kontoris koos hankija spetsialistidega analüüsi tegemiseks, konsultatsioonideks või uute funktsioonide väljaõppeks.
- 5.7. Täitjapoolset Tellimuse tulemusena infosüsteemis valminud lisaarenduste üleandmiseks loetakse infosüsteemi lähtekoodi salvestamist Tellija SVN tarkvarasse ning Täitja esindaja poolt Tellija esindajale digitaallkirjastatud kontaineri edastamist, mis sisaldab Tellimuse kohta kehtivat dokumentatsiooni vastavalt hankelepingus sätestatud dokumentatsiooniplaanile.
- 5.8. Tellimuse vastuvõtmine toimub vastavalt hankelepingus sätestatud korrale.

6. Üldised nõuded testimisele

- 6.1. Lisaarenduse üleandmisel peab lisaarendus olema testitud Täitja poolt.
- 6.2. Lisaarenduse tulem peab üleandmisel vastama üldistele testimise nõuetele:
 - 6.2.1. lisaarendus on iseendaga kooskõlas (pole katkiseid mooduleid, iseendaga vastuolus olevat funktsionaalsust);
 - 6.2.2. commititava haruga kooskõlas (varasemad lisaarendused ei lähe katki, sh. mitte antud projekti lisaarendused);
 - 6.2.3. varasemat projekti muutes peab uus lisaarendus olema kooskõlas varasemaga.
- 6.3. Täitja esitab Tellijale testiplaan, testlood ja testraportid. Testlugusid kirjutatakse vastavalt vajadusele nende testide kohta, mille juures on neist kasu:
 - 6.3.1. testiplaan koostatakse Tellija poolt kinnitatud analüüsi järgi enne programmeerimise algust ja vajadusel täiendatakse testimise vältel;
 - 6.3.2. testiplaan sisaldab testlugusid, nende läbiviimise järjekorda ja omavahelisi seoseid – testiplaan peab näitama mismoodi loodavat funktsionaalsust 6 testida;
 - 6.3.3. testiplaan ja testlood peavad olema muust dokumentatsioonist iseseisvad, st et kolmandatel osapooltel peab olema võimalik selle järgi testida muusse dokumentatsiooni süvenemata;
 - 6.3.4. testilugude komplekt tuleb esitada ühes inimloetavas vormis;
 - 6.3.5. üks testilugu võib kontrollida mitut nõuet.

7. Nõuded lisaarenduste üleandmisele

Loodavad hanke tulemid antakse üle RIK-i koodirepositooriumi (SVN) kaudu. Täitja meeskonnaliikmetele luuakse RIK-i poolt ligipääs SVN-ile lähtekoodide, andmebaasimuudatuste ning muude tulemite või nende muudatuste üleandmiseks.

Lisaks tarnib Pakkuja koos iga üleantava versiooniga ka administraatori jaoks valmis pakendatud antud versiooniuuenduse paigalduspaketi. Paigalduspakette SVN-i ei laeta ja need antakse üle näiteks FTP kaudu (täpne keskkond lepatakse eraldi kokku).

RIKi SVNiga suhtlemiseks on soovitatav kasutada tortoisessvn nimelist tarkvara:

Kättesaadav: <http://code.google.com/p/tortoisessvn/downloads/list>

Manual: http://tortoisessvn.net/docs/release/TortoiseSVN_en/

7.1. Commit

Arendamisel lisada commitide (lähtekoodi depositeerimisse lisamine) juurde kindlasti töökäsu nr, mille alusel muudatus sisse viiakse. Töökäsu puudumisel sobib lühisõnaline lühikirjeldus, mida muudatused sisaldavad. Commit sõnumi alguses peab olema alati projekti nimi, mille raames commit tehakse ning kindlasti lisada info selle kohta, et kes antud commiti teeb (isiku nimi).

Vigade parandamisel tuleb alati committimisel lisada ticketi number märkusena.

Muudatusi tuleb committida vähemalt kord päevas.

Täpsemalt on võimalik lugeda: http://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-dug-commit.html

Enne commiti peab kood/teostatud töö/tulem olema:

- 1) Iseendaga kooskõlas (pole vastuolusid, katkiseid mooduleid, iseendaga vastuolus funktsionaalsust);
- 2) Üldharu/muu haru tulemiga kooskõlas (üldharus olemasolev pole peale commiti katki);
- 3) 3) Edaspidi mugavalt kasutatav ja mõistetav, ehk sellel on olemas oma enda sisseehitatud testid, mis peale järgnevaid võimalikke arendusi/commite/tegevusi näitaks, kas see konkreetne funktsionaalsuse osa on endiselt korras, või läks katki ja tuleb korrastada.

Commititav kood/teostatud töö/tulem peab olema piisavalt testitud ja Täitja poolt testidega kaetud, kasutades selleks:

- a) Unittest'e,
- b) Systemtest'e
- c) Manuaalseid teste

7.2. Tagimine

Versiooni üleandmiseks tuleb sellest koostada tag. Tagi number peab ühtima versiooni numbriga, mida üle antakse, kuid tagi nimi peab alati algama projekti lühendiga. Tagi alla tuleb siduda kõikide arhitektuuriliste komponentide (rakendus, andmebaasiscriptid, x-tee adapter jne) ning muudetud või loodud dokumentatsiooni commitid.

Versiooni numbrid peavad olema vähemalt kolmekohalised (X.Y.Z, milles X- põhiversioon, Y - üleantav versioon, Z paranduse number). Vajadusel võib kokkuleppida teistsuguses numeratsioonis.

http://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-dug-branchtag.html#tsvn-dug-branch-1

Iga tagiga (uue versiooniga) peab kaasnema versiooni kirjeldus (mis taskid, mis ticketid, mis baasiscriptid, mis juhendeid täiendati jne), mis tuleb paigutada dokumentatsiooni alla.

Dokumentatsioon

Dokumentatsiooni tuleb hoida trunk/doc/ kaustas. Alampuud lepitakse eraldi kokku.

Baasiscriptid

Andmebaas ning selle muudatused tuleb alati üle anda loovate, muutvate, täiendavate jne skriptidena ning neid tuleb alati hoida trunk/baas/ kaustas.

Andmebaasiscriptide failinimed peavad olema kujul:

YYYYMMDDHHMM_prefiks+Järjekorranr.sql ehk 201105131430_KR0003.sql kus prefiks on süsteemi lühend.

7.3. Täitjapoolne testimine

Täitja peab lisaks kasutuslugude funktsionaalsele testimisele teostatava arenduse/töö/tulemi commitimisel ja üleandmisel ise tõestatavalt veendunud olema, et nende arendus/töö/tulem teeb seda, mis on nõutud ning samaaegselt ei riku/lõhu juba olemasolevat muud arendust/tööd/tulemit.

Iga kirjutatud koodiüksuse kohta peaks olemas olema ka vastav test, mis kindlustab, et loodu toimib ja seda ka eriolukordades. Samuti on testid abiks juba olemasolevate komponentide muutmisel, indikeerides koheselt võimaliku probleemolukorra.

Arendatav funktsionaalsus peab olema kaetud:

- 1) Unittestidega (kõik unittestid kooskõlastada tellijaga)
- 2) Süsteemitestidega (kõik süsteemitestid kooskõlastada tellijaga)
- 3) Manuaalsete testidega

Igal commitimisel ning üleandmisega peab edukalt:

- 1) käivitama kõik Unittestid ning selle dokumenteerima.
- 2) käivitama kõik süsteemitestid ning selle dokumenteerima
- 3) käivitama kõik manuaalsed testid ning selle dokumenteerima
- 4) dokumenteerima, mis test data't kasutati, mis keskkondi kasutati, jne.

Unit-testid e. moodultestid

Moodultestid testivad ÜHTE klassi isoleeritult teistest klassidest ja ressurssidest (andmebaas).

Idealis peaks igal iseseival klassil olema oma moodultest, mis kataks selle funktsionaalsust täielikult.

Kuna klassid kasutavad teineteist, moodustuvad niiöelda kobarad, siis moodultesti ideoloogiaga see hästi kokku ei sobi. Selleks, et saaks testida tõesti vaid konkreetse ühe klassi funktsionaalsust, on kasutusel mock-objektid(väline komponent Rhino Mocks).

Süsteemitestid

Süsteemitestid testivad süsteemi kasutaja vaatenurgast, eesmärgiks on kontrollida süsteemi vastavust kasutaja nõudmistele.

Süsteemitestid jagunevad omakorda kaheks:

- Funktsionaalsuse testid - üldjuhul koostatakse konkreetse teenuse klassi süsteemitestid sellises mahus, et kaetud oleksid selle kõik erinevad meetodid ning lisaks ärioloogiliselt olulisemad stsenaariumid.
- Mittefunktsionaalsete aspektide testid - näiteks jõudluse testimiseks.

Üldised nõuded testide kirjutamisel

Nagu eelnevalt sai kirjeldatud, on testid liigendatud oma olemuse järgi. Testi sisu ja iseloom varieerub olenevalt selle kategooriale (nt. [Test, Category("Bug")] – testis kirjeldatakse kindel veaolukord ning kontrollitakse selle põhjal vea olemasolu ET-s, st. testi andmed on antud veaolukorra järgi koostatud ja spetsiifilised). Küll aga on kujunenud kindlad head tavad, mida tuleb kõigi testide kirjutamisel järgida – koodieetikat, testide liigendamist, testide automatiseeritust (st. teste peab saama käivitada skriptidega projektiväliselt nt. kategooria alusel iga-öiselt) ning testide autonoomsust (testid peavad võimalikult hästi taluma ET muudatusi).

Koodieetika ja struktuur testide kirjutamisel

Testkood peaks olema liigendatud ja selgelt loetav:

- Defineerida testis projekti erinevate osade kasutus (nt. using System;);
- Iga testklass tuleb deklareerida [TestFixture]-raamistikus ning olenevalt testi iseloomust nimetada kujul Bug1234Test (kindlat viga #1234 kirjeldav Bug-test), Ticket1234Test (uuendust #1234 kontrolliv test) või TestiKirjutajaEesnimi_TeenuseNimiTest (teenuse spetsiifiline süvatest);
- Testis olevad privaatsed klassimuutujad on tava kohaselt kujul _muutujaNimi, näiteks private Menetlus _menetlus;
- Kaitstud ja avalikud muutujad on kujul muutujaNimi või MuutujaNimi;
- Vajalik uute klassiobjektide initsialiseerimine (nt. liidesed) toimub meetodis SetUp(). Näiteks:
[SetUp] public void SetUp()
{
 _liides = new Liides();
}
- Iga uus testcase tuleb kirjutada raamistiku sisse kujul
[Test, Category("Kategooria")]
ning testi enda nimetus tuleneb testi tüübist:

- Bug-testide ja uuendusi kontrollivate testide korral nimetus, mis kirjeldab võimalikult hästi ja konkreetselt kontrollitavat objekti/olukorda (nt. teenusel VäärteoMenetluseAlustamine ei salvestu süüteod ET-baasi: testi pealkiri SyyteoSalvestumineTest());
- Teenusepõhiste süvatestide puhul esiteks test, mis kontrollib, et teenust saaks nii minimaalsete kui ka maksimaalsete andmetega kinnitada, nimetusega A_SetUp() ning edasi kujul B_Äridokumendi_esimene_punkt() jne.
- Testandmed võiksid olla võimalikult reaalse elu lähedased ja loogilised, st. ei tohiks tekitada uut objekti klassist Helikopter kujul _a, vaid kujul _helikopter.
- Iga äridokumendis olev nõue või tingimus tuleb testida eraldi testcase'i all, olenemata sellest kas kontrolliks on üks või rohkem ridu koodi. Kui teha ärioloogiline kontroll vaid ühe testi all, siis ei kajasta see teenuse tervist – st. kui esimene kontroll annab veateate, siis ülejäänuteni test ei jõua. Selle vältimiseks on testid ärinõuete kaupa eraldi kirjutatud.

Autonoomsus

Testid peavad taluma pisemaid koodi muudatusi (neid, millega ei kaasne DTO või andmemudeli muutused).

Automatiseeritus

Kõiki teste peab saama käivitada näiteks projektiväliselt skripti abil. Selleks peavad testid olema õigesti kategoriseeritud.

Kujundus

Funktsionaalsuse arendamisel tuleb lähtuda olemasolevast kujundus- ja navigeerimisloogikast, mis on kirjeldatud <http://uig.rik.ee> veebilehel.

Uute komponentide lisandumisel tuleb need eelnevalt Tellijaga kooskõlastada.

/allkirjastatud digitaalselt/

Mehis Sihvart

direktor