

Lisa 1
RMK ja Askend Estonia OÜ
Vahelise lepingu nr 1-
18/2026/92
juurde

Mobiilirakenduste arendushange

Riigimetsa Majandamise Keskus

Taust

Riigimetsa Majandamise Keskus (RMK) kasutab matkaradade kuvamiseks, metsatööde juhtimiseks, töövõtjate tegevuste juhtimiseks ja mitmete teiste tööde andmete sisestamiseks paljusid mobiilirakendusi, millest valdav enamik on tänaseks tehnoloogiliselt vananenud.

Antud hanke eesmärk on leida RMK-le **mobiilsete rakenduste arenduspartner**.

Partneril peab olema kompetents ja kogemused mobiilsete rakenduste arendamisel.

Oluline on jätkusuutlikele lahendustele orienteeritud mõtteviis, operatiivsus ja kaasaegsetele tehnoloogiatele tuginemine.

Hanke eesmärk ja ulatus

RMK soovib hankida mobiilirakenduste arendusteenuse, mis aitab ellu viia:

- **RMK mobiilirakenduste väljavahetamise kaasaegsete ja turvaliste mobiilirakenduste vastu;**
- **Aitab valideerida meie enda uuendatud mobiilirakenduste kvaliteeti ja teha ettepanekuid parendusteks;**
- **Mobiilirakenduste komponentide hooldust ja versiooniuuendusi kokkulepitud ulatuses ja skoobis kogu raamlepingu perioodi vältel.**

Hetkeolukord

Järgnevalt kirjeldame RMK olemasolevad mobiilirakendused ja tänased tehnoloogiad, mis kõik vajavad uuendamist.

Olemasolevad rakendused

Avalikud rakendused

- RMK Loodusega koos App – Rakendus Google Plays (Cordova)
- RMK Loodusega koos App - Rakendus App Store (Cordova)

[RMK Loodusega koos – Rakendused Google Plays](#)

[RMK Loodusega koos App - App Store](#)

Sisemised mobiilsed rakendused

- ChipsDriver - Hakkpuidu veoki rakendus (Android *native*)

- Chipper - Hakkurile mõeldud rakendus (Android *native*)
- TimberDriver - Ümarpuidu veoki rakendus (Android *native*)
- Taksaator - Metsakorraldaja rakendus (Android *native*)
- Traktorist(Forwarder) - Kkokkuveo traktoristi rakendus (Android *native*)
- Saamees(Sawyer) - Raie ettevalmistuse võsatööde rakendus (Android *native*)
- MKT KVH (Kvaliteedi hindamine) - Metsakasvatavate analüütikute rakendus välitöödel (Android *native*)
- MMR - Metsamees - MKT ja TVT lisamoodul: Metsakasvatuse saemehe rakendus (Android *native*)
- DMK (Dokumendid metsa kaasa) - Rakendus, mis võimaldab salvestada mobiilseadmes offline'i kõik dokumendid, mis üleval lehel juhendid.rmk.ee (Android *native*)
- AMK (Andmed metsa kaasa) - Andmed Metsa Kaasa - Offline võimekusega mobiilne rakendus. Kvartalid, eraldised, teed ja kraavid. Andmed laetavad metuskondade kaupa. (Android *native*)

Tehnoloogiline hetkeolukord

Praeguse seisuga on ametlikes rakenduste poodides (Google Play, Apple App Store) üksnes avalik RMK rakendus Loodusega Koos. Loodusega Koos kasutab Cordova platvormi, mis võimaldab ühest veebitehnoloogiate põhisest koodibaasist genereerida *native* rakendused nii Google Play kui ka App Store-i rakenduste poe jaoks.

Teisi tööde juhtimiseks ja haldamiseks mõeldud rakendusi levitatakse läbi RMK enda loodud keskkonna(uuendused, vealogid, kasutatavus). Need on Android *native* rakendused (Java).

Rakendustel on tüüpiliselt ka java backend, suhtlus toimub läbi REST API.

Native Android rakendused on tehniliselt vananenud, sellega seoses on tekkinud mitmeid probleeme:

- Google Play Protect mehhanism märgistab rakendusi kui kahtlasi (installitud mitte-ametlikust allikast, vana Android SDK versioon)
- Rakenduste suhtlus serveriga ei vasta kaasaegsetele turvanõuetele
- Võimalikud turvavead vananenud komponentides
- Kasutajad piiratud üksnes Android platvormiga

Soovitud tehniline arhitektuur

Käesolev peatükk kirjeldab RMK poolt eelistatud arhitektuurset lähenemist ja tehnoloogilisi valikuid. Täpsemad kohustuslikud tehnilised nõuded on toodud peatükis “Tehnilised nõuded”.

PS.RMK on avatud arenduspartneri soovitudele ja arhitektuurialastele nõuannetele mobiilirakenduste tehnoloogilise arhitektuuri ülesehitamisel.

Üldine lähenemine

Avalikud ja laiale kasutajaskonnale suunatud rakendused (nt Loodusega Koos) levitatakse ametlike rakenduste poodide kaudu ning peavad olema native paketid. Eelistatult genereeritakse need ühest ühisest koodibaasist mitmele platvormile.

Sisemised, tööks mõeldud rakendused realiseeritakse esmajärjekorras Progressive Web App (PWA) tehnoloogial ning hostitakse RMK poolt. Juhtudel, kus PWA ei ole tehniliselt või funktsionaalselt sobiv, kasutatakse Androidi native-lahendust, ning levitatakse läbi Google Play Store-i.

Selline lähenemine võimaldab valida rakenduse levitus- ja kasutusviisi vastavalt sihtrühmale ja kasutusstsenaariumile, tagades kiire muudatuste tarnimise, vähendatud arendus- ja hoolduskulud ning tehnoloogilise sõltumatuse.

Alljärgnevalt on kirjeldatud RMK kontekstis eelistatud arhitektuursed realiseerimisviisid.

Native rakendused

- Native-rakendused realiseeritakse native pakettidena ning levitatakse alati ametlike rakenduste poodide kaudu (Google Play, Apple App Store).
- Native-lahendus võib olla arendatud platvormipõhiselt või genereeritud ühisest koodibaasist (nt React Native, Kotlin Multiplatform, Flutter).
- Avalikud ja laiale kasutajaskonnale suunatud rakendused peavad toetama nii Androidi kui ka iOS-i platvormi.
- Sisemised tööks mõeldud rakendused võivad olla piiratud Androidi platvormiga, kuid levitatakse siiski ametliku rakenduste poe (Google Play) kaudu.
- Native-rakendused võimaldavad täielikku ligipääsu seadme võimekustele (nt biomeetriline autentimine, turvaline võtmehoidla, offline-tugi).
- Kasutaja autentimine (nt TARA kaudu) ja turvamehhanismid on lahendatud vastavalt rakenduse kasutusotstarbele.

Puhas PWA-lahendus

- Rakendused realiseeritakse Progressive Web App (PWA) tehnoloogial ning töötavad brauseripõhiselt ilma native paketit.
- PWA-rakendused hostitakse RMK enda infrastruktuuris, ilma rakenduste poodide vahendusest.
- Paigaldamine kasutaja seadmesse toimub PWA mehhanismide kaudu.
- Autentimine, offline-tugi, seadmevõimekused ja uuendused on lahendatud PWA tehnoloogiate abil.

PWA tehnilised omadused

Järgnevalt on kirjeldatud PWA tehnoloogiaga kaasnevaid tüüpilisi võimalusi ja omadusi, mis on RMK rakenduste kontekstis asjakohased.

- Offline'i tugi - kõige olulisem eelis, kuna RMK rakendustega töötatakse ebaühtlase leviga piirkondades. PWA võimaldab rakendustel jätkata tööd ka aktiivse netiühendusest, salvestades andmed kohalikult ja sünkroniseerides need hiljem, kui ühendus taastub. See tagab töö pidevuse ka kehvema leviga metsapiirkondades;
- Geolokatsioon - võimaldab täpse asukoha määramist, mis on oluline metsatööde dokumenteerimiseks ja juhtimiseks;
- Teavitused - võimaldab kasutajatele oluliste sündmuste ja uuenduste kohta teavituste saatmist, isegi kui rakendus ei ole aktiivselt avatud;
- Uuenduste kiire levitamine - muudatused ja uuendused saab tarnida otse kasutajatele ilma rakenduste poe review protsessi läbimata, tagades kiire reageerimise vajadustele;
- Turvaarhitektuur - kasutab parimaid veebirakenduste turvapraktikaid, sh HTTPS, turvaline autentimine ja andmete krüpteerimine.

PWA eelised

- Vähendatud arendus- ja hoolduskulud - üks koodibaas töötab nii Androidi kui ka iOS platvormidel, samal ajal pakkudes native rakendustega võrreldavat kasutajakogemust;
- Laiendatav seadmete tugi - võimaldab laiendada toetatavate seadmete hulka, kuna seni on metsatööde rakendustes toetatud üksnes Androidi. PWA lahendus võimaldab nüüd toetada ka iOS seadmeid ja tulevikus võimaldab hõlpsalt lisada toetust ka teistele platvormidele;
- Kiire muudatuste tarnimine - võimaldab kiirelt muudatusi klientidele tarnida, ilma et rakenduste poe review protsessist tuleneva viivitusest;
- Vähendatud sõltuvus - vähendab sõltuvust rakenduste poe omanikfirma poliitikast ja nõudmistest.

Autentimine

Autentimise vaates on 3 erinevat stsenaariumit: RMK enda töötajad, välised kasutajad kes kasutavad isikupõhist autentimist ja seadmepõhine autentimine.

RMK töötajad autentivad kasutades Azure Entra ID-d.

RMK väliste kasutajate isikupõhiseks autentimiseks kasutame TARA/Smart-ID teenuseid.

Seadmepõhise autentimise korral kasutatakse seadme identiteedil põhinevat autentimisskeemi (nt tokeni või sertifikaadi alusel).

Tehnilised nõuded

Offline-first ja ebastabiilne võrk

- Rakendus peab töötama ka piiratud ühenduse korral (katkendlik või puuduv ühendus), sh vaated ja põhifunktsioonid peavad degradeeruma kontrollitult.
- Lokaliseeritud andmehoid (nt IndexedDB / SQLite) koos selge sünkroniseerimisloogikaga (järjekorrad, retry, backoff).
- Konfliktihaldus: määratletud reeglid (last-write-wins / merge / kasutaja valik) ning kasutajale arusaadav tagasiside.
- Failide ja suurte payload'ide käsitlemine: chunking, katkestuse jätkamine (resume), progress ja ajutine salvestus.

Andmete sünkroniseerimine ja töökindlus

- Kõik serverioperatsioonid peavad olema idempotentsed või idempotentsusega kooskõlas (võrgu korduspäringud ei tekita duplikaate).
- Taustsünkroniseerimine (PWA: Background Sync võimalusel; muidu app-level scheduler) ning selge järjekorrastamise mudel.
- Versioonitud API ja andmemudel: tagurpidi ühilduvus vähemalt N versiooni ulatuses.
- Kliendi ja serveri vaheline suhtlus peab toimuma standardiseeritud API kaudu, eelistatult REST/JSON protokollil abil, võimaldades lihtsat versioonimist, vahemällu salvestamist ja tõrketaluvust.

Turvalisus ja identiteet (mobiilikontekst)

- Autentimine/autoriseerimine standardprotokollidega (OIDC/OAuth2), tokenite turvaline hoidmine (PWA: mitte localStorage, eelistada HttpOnly cookie-t või turvalisi storage mustreid).
- Sessiooni aegumine ja taastamine: kasutaja ei kaota tööd, aga turvariskid on kontrolli all.

- Tundlikud andmed, kui neid hoitakse lokaalselt, peavad olema krüpteeritud. Rakendus peab toetama rakendusesisest lukustust ja turvalist uuesti autentimist. Biomeetria või ekraanilukustus kasutatakse juhul, kui platvorm seda standardliidest kaudu võimaldab.
- Minimaalsete õiguste põhimõte (least privilege) nii kliendis kui API-s.
- Andmeside turvalisus: kogu rakenduse andmeside kliendi ja serveri vahel peab olema krüpteeritud, kasutades kaasaegseid turvaprotokolle (nt HTTPS/TLS), ning krüpteerimata ühendused ei ole lubatud.
- Rakenduse turvalisus: rakenduse arendamisel ja konfigureerimisel tuleb arvestada OWASP-i asjakohaseid turvasoovitusi (sh OWASP Top 10), et vältida levinud turvanõrkusi.

Uuendused ja versioonihaldus (PWA eripära)

- PWA puhul peab olema selge uuenduste loogika (“update strategy”): kasutajale arusaadav teavitus uue versiooni olemasolust.
- Cache’i versioonimine ja migratsiooniplaan (service worker, asset cache, API cache).
- Mobiilirakenduse puhul tagurpidi ühilduvus serveri poolel (vanemad kliendid ei tohi kohe katkeda).

Seadmevõimekused ja platvormierinevused

- Selge abstraktsioonikiht seadme-API’dele (kaamera, GPS, failid, teavitused), et hoida äri- ja UI-loogika platvormisõltumatuna.
- Piirangute käsitlemine: rakendus peab arvestama kasutaja rolle ja õigusi, käsitlema juurdepääsupiiranguid ja tõrkeolukordi ning andma kasutajale arusaadavat tagasisidet ja juhiseid.
- Energiakasutus ja ressursid: piirata tausttööd, koondada sünkroniseerimisi, vältida liigset GPS kasutust.

Ligipääsetavus (WCAG) ja kasutatavus välitingimustes

- Siht: Vähemalt WCAG 2.1 AA või parem.
- Klaviatuur/fookus (PWA), ekraanilugejad (TalkBack/VoiceOver), kontrast, suured klikitavad sihtmärgid.
- Välitöö kontekst: suured nupud, kindaga kasutus (target size), selged veateated, “undo” kriitilistes tegevustes.
- Rakendus peab kohanema erineva suuruse ja orientatsiooniga ekraanidega (responsive design), tagades põhifunktsioonide kasutatavuse nii mobiilseadmetes, tahvlites kui ka suurematel ekraanidel.

Korduvkasutatav arhitektuur ja komponendid (mitme rakenduse portfell)

- Ühtne disainisüsteem/komponendikogu (UI kit) ja ühised UX mustrid.
- Ühiste teenuste ja klienditeekide kiht: auth, API client, error handling, logging, feature flags.
- Modulaarne arhitektuur: funktsionaalsed moodulid eraldi pakettidena, et suurendada koodi korduvkasutust ja vähendada vigu.
- Ühtsed kvaliteedireeglid: lint/format, testinõuded, sõltuvuste poliitika.

Testitavus ja kvaliteediväravad

- Automatiseeritud testid: unit, integration (API/sync), e2e kriitiliste voogude jaoks.
- Ligipääsetavuse automaattestid (PWA) + manuaalsed kontrollid.
- CI/CD: reproducible builds, PWA deploy immutable. PWA väljalasked peavad olema muutumatud (immutable), kus iga versioon kasutab eraldi versioonitud ja hashitud ressursse ning olemasolevaid väljalaskeid ei muudeta tagantjärele, võimaldades kontrollitud ja etteaimatavat rakenduse uuendamist.

Monitooring, logimine ja auditeeritavus (rakenduse tasemel)

- Kliendipoolne telemeetria ja vealogimine (crash/JS errors), korrelatsiooni-ID päringutel.
- Offline sündmuste järjekord (event queue) ja saatmine ühenduse taastumisel.
- Privaatsus: isikuandmete minimeerimine logides. Rakenduse logimise ja monitooringu konfiguratsioon peab tagama isikuandmete (PII) automaatse eemaldamise või maskeerimise enne logide talletamist ja edastamist.

Jõudlus

- Rakenduse arhitektuur peab toetama kiiret külmkäivitust ja lühikest aega kuni rakendus on kasutajale tegelikult kasutatav (time-to-interactive). Selle saavutamiseks tuleb rakenduse kood ja ressursid laadida järk-järgult, tuues alguses ainult käivituseks ja esmaseks kasutuseks vajaliku. Ülejäänud funktsionaalsus ja vaated laaditakse nõudepõhiselt (lazy-load), vähendades algset allalaaditavat mahtu. PWA puhul tuleb rakendada koodi ja ressursside jaotamist (code splitting), et vältida ühe suure monoliitse paketi laadimist.
- Andmemahutade piiramine: paging, delta sync, kompressioon.
- Piltide/failide optimeerimine ning arvestus et kasutajate seadmed ei pruugi väga võimsad olla.

Mitmekeelsus ja lokaliseerimine

- Rakendus peab toetama mitmekeelset kasutajaliidest (nt eesti ja inglise), võimaldama keele vahetamist ning kasutama lokaliseerimislahendust, mis eraldab keelelise sisu rakenduse loogikast.

Otsitav kompetents ja kvalifikatsioonid

Üldpädevus

Arendajal peab olema praktiline kogemus mobiilirakenduste arendamisel nii *native* lahenduste (Android, iOS) kui ka Progressive Web App (PWA) tüüpi rakenduste kontekstis. Ta peab suutma iseseisvalt kavandada, arendada, testida ja hooldada rakendusi kogu elutsükli vältel ning mõistma mobiilirakenduste arhitektuurilisi, ligipääsetavus-, jõudlus- ja turvanõudeid.

Androidi *native* rakenduste arendus

Arendajal peab olema:

- tugev praktiline kogemus Android rakenduste arendamisel (Android SDK);
- oskus kasutada programmeerimiskeeli Java ja/või Kotlin;
- kogemus Android Studio, Gradle'i ning Androidi build'i ja deploy protsessidega;
- teadmised Androidi arhitektuurimustritest (nt MVVM, Clean Architecture);
- kogemus UI arendamisel (XML, Jetpack Compose) ning erinevate ekraanisuuruste ja seadmete toetamisel;
- arusaam Android rakenduste jõudlusest, mäluhaldusest ja elutsüklist;
- kogemus rakenduste avaldamisel ja haldamisel Google Play poes.

iOS *native* rakenduste arendus

- tugev praktiline kogemus iOS rakenduste arendamisel (iOS SDK);
- oskus kasutada programmeerimiskeelt Swift (soovitav) ja/või Objective-C;
- kogemus XCode'i, Swift Package Manageri (või CocoaPods'i) ning iOS build'i ja deploy protsessidega;
- kogemus UI arendamisel (SwiftUI ja/või UIKit) ning erinevate seadmete ja orientatsioonide toetamisel;
- arusaam iOS rakenduste elutsüklist, mäluhaldusest ja jõudlusest;
- kogemus rakenduste avaldamisel ja haldamisel App Store's (sertifikaadid, provisioning profile'd, TestFlight).

PWA ja veebipõhised mobiilirakendused

Arendajal peab olema:

- praktiline kogemus PWA-de arendamisel;
- head teadmised veebitehnoloogiatest: HTML5, CSS3, JavaScript / TypeScript;
- kogemus kaasaegsete veebiraamistikega (nt React, Vue, Angular või sarnased);
- arusaam PWA põhikomponentidest: service worker'd, offline-toetus, caching, web manifest;
- kogemus responsive disaini ja mobiilse kasutajakogemuse loomisel;
- teadlikkus PWA piirangutest ja erinevustest võrreldes *native* rakendustega.

Integratsioon, taustasüsteemid ja API-d

Arendajal peab olema:

- kogemus REST- ja/või GraphQL API-de kasutamisel;
- arusaam autentimis- ja autoriseerimismehhanismidest (nt OAuth2, JWT);
- teadmised andmevahetusest, veahaldusest ja võrguühenduse ebastabiilsuse käsitlemisest.

Ligipääsetavus ja kasutatavus (WCAG)

Arendajal peab olema teadmised ja praktiline kogemus digiligipääsetavuse põhimõtete rakendamisel. Ta peab arendustöös arvestama WCAG 2.1 või uuema versiooni nõuetega ning suutma neid rakendada nii Androidi ja iOS *native* platvormil kui ka PWA tüüpi lahendustes.

Sealhulgas eeldatakse:

- arusaama ligipääsetavuse põhialustest (tajutavus, kasutatavus, arusaadavus, töökindlus);
- oskust rakendada korrektseid semantilisi elemente, kontrastsuse, fookuse ja navigeeritavuse nõudeid;
- teadlikkust ekraanilugerite, klaviatuuriga kasutamise ja abitehnoloogiate eripäradest;
- võimet hinnata ja parandada olemasolevate kasutajaliideste ligipääsetavust;
- valmisolekut dokumenteerida ja põhjendada tehtud ligipääsetavusalaseid otsuseid.

Kvaliteet, testimine ja turvalisus

Arendajal peab olema:

- kogemus automaat- ja manuaaltestimisega (nt unit-, integratsiooni- ja UI-testid);
- arusaam mobiilirakenduste turvariskidest ja parimatest praktikatest;
- oskus järgida versioonihalduse ja koodikvaliteedi põhimõtteid (Git, code review).

Tööstiil ja koostöö

Arendajalt eeldatakse:

- võimet töötada iseseisvalt ja meeskonnas;
- oskust dokumenteerida tehnilisi lahendusi arusaadavalt;
- kogemust agiilsete arendusmeetoditega (nt Scrum, Kanban);
- valmisolekut õppida uusi tehnoloogiaid ja kohaneda projekti vajadustega;

Töökorraldus ja nõuded

Dokumentatsioon

Arenduste teostamisel peab Täitja koostama dokumendi(d), millede eesmärgiks on anda selge ja konkreetne ülevaade süsteemi sisust ja realiseeritud arendustest. Eraldiseisvaid dokumente üle ei anta vaid Tellija keskkonda (Confluence) kirjutatakse analüüsi jooksvalt projekti käigus. Kirjeldused peavad olema kinnitatud Tellija poolt enne arendustöö algust.

Üle antavad dokumendid peavad sisaldama:

- Infosüsteemi terviku kirjeldus – olemasoleva süsteemi täiendamisel lisatud/muudetud osade kirjeldus. Tegemist on süsteemi ärisisul põhinevate baaskomponentidega, millele toetuvad kasutaja tegevused, ekraanivormid, kasutuslood/protssid, teenused jms;
- Vormide (ekraanivormide) kirjeldused;
- Kasutuslugude / vormide kasutamise protsesside / tegevuste kirjeldused;
- Infosüsteemi teenuste kirjeldused;
- Juurutuse faasis ja lõpus peab Täitja esitama arhitektuuri kirjeldus dokumendi, mille eesmärgiks on kirjeldada loodava ja/või juurutatava süsteemi tehnilist ülesehitust. Kirjeldatakse rakenduse loogilist struktuuri, näidates ära selle kihtideks jagunemise;
- Kirjeldatakse ka füüsilist arhitektuuri, antakse ülevaade kasutatavatest tehnoloogiatest ning vahenditest.

Dokumenteermise kohta täpsemalt on kirjas Arendusnõuete all.

Testimine

- Enne töö üleandmist viib Täitja töö nõuetelevastavuse kindlakstegemiseks läbi testid.
- Kui töö või mistahes selle osa ei läbi teste, viiakse pärast seda, kui Täitja on teinud vajalikud korrektuurid testide edukaks läbiviimiseks, läbi kordustestid samadel tingimustel. Tellija nõudmisel viib kordustestid läbi Täitja.

Tugiteenus

- Raamlepingu täitmise ajal tagab pakkuja tarkvara nõuetekohase toimise, sh intsidentide tekkimise korral reageerima ja probleemi lahendama vastavalt teenustasemetele;
- Pakkuma ülalhoiu teenust peale süsteemi etappide järkjärgulist tootekeskkonda paigaldamist;
- Parandama turvanõrkuseid ja uuendama tarkvara vastavalt lepingus sätestatule;
- Hoidma tehnilist ja lõppkasutaja dokumentatsiooni ajakohasena;
- Nõustama hankijat süsteemis muudatuste tegemisel ja kasutamisel.

Lisad: Lisa 1- Mobiilirakenduste arendushange – meeskonna rollid ja ülesanded