LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

HTML5 UN ADOBE AIR HIBRĪDO MOBILO LIETOTŅU IZSTRĀDES SALĪDZINĀJUMS

BAKALAURA DARBS

Autors: **Armands Baurovskis**
Studenta apliecības Nr.: ab11196
Darba vadītājs: asociētais profesors, Dr. dat. Uldis Straujums

RĪGA 2015

ANOTĀCIJA

Mobilo lietotņu tirgus ir plašs un ar katru nākamo gadu lietotņu skaits strauji pieaug. 2014. gadā 80% pasaules iedzīvotāju bija vismaz viens viedtālrunis. Bakalaura darbā ir veikts *HTML5* un *Adobe AIR* hibrīdo mobilo lietotņu izstrādes platformu salīdzinājums. *HTML5* izstrādes pieeja ir lietotņu izstrāde, izmantojot tīmekļa pārlūku, bet *Adobe AIR* izstrādes pieeja veic kompilāciju no augsta līmeņa programmēšanas valodas uz dzimtās platformas zema līmeņa valodu – bitkodu. Abas mobilo lietotņu platformas tika salīdzinātas pēc *Henning Heitkötter* vairākplatformu salīdzinājuma kritērijiem. Gala rezultātā tika izveidota mobilā lietotne priekš *HTML5* un *Adobe AIR* izstrādes platformām un veikti platformu novērtējumi. Novērtējumi tika apkopoti tabulās un diagrammās, kas apraksta abu izstrādes platformu kopīgās/atšķirīgās īpašības un priekšrocības/trūkumus.

Atslēgvārdi: *HTML5, Adobe AIR,* hibrīdo mobilo lietotņu izstrād*e, Phaser, Apache Cordova, Starling.*

ANNOTATION

Comparison of HTML5 and Adobe AIR hybrid mobile applications development

The mobile application market is wide, and with each passing year application, count rapidly increases. In 2014, 80% of world's population has at least one smartphone. In the work, author compare two hybrid mobile application platforms – *HTML5* and *Adobe AIR*. *HTML5* development approach is mobile application development using WEB browser, while *Adobe AIR* development approach compiles high-level programming language to native platform low-level language like byte code. Both development platforms were compared by *Henning Heitkötter* cross-platform development approach evaluation criteria's. The result is mobile application development in *HTML5* and Adobe AIR and comparison of platforms. The assessments were summarized in tables and charts, who describes approaches common/distinctive features and advantages /disadvantages.

Keywords: *HTML5*, *Adobe AIR*, hybrid mobile application development, *Phaser*, *Apache Cordova*, *Starling*.

SATURS

APZĪMĒJUMU SARAKSTS UN DEFINĪCIJAS

Adobe AIR – *Adobe* Integrētā izpildlaika sistēma ir daudz platformu izpildlaika sistēma, kuru izstrādāja uzņēmums *Adobe Systems.*

API – iepriekš definētu klašu, procedūru, funkciju, struktūru un konstanšu kopums, kas tiek pasniegts kā pielikums (bibliotēkas, servisi), kuru iespējams izmantot ārējiem programmatūras produktiem. Izmanto programmētāji, lai rakstītu dažādus programmu pielikumus.

Bitkaršu spraits – Vizuāla entitāte, kas, pārvietojoties pa fona priekšu, rada secīgu animāciju.

Distribūcija – process, kas apraksta preču vai pakalpojumu plūsmu no piedāvātāja līdz patērētājam.

Dzimtās platformas lietotne – lietotnes kura ir izstrādāta, izmantojot platformas nodrošinātos izstrādātāju rīkkopas.

Hibrīdā mobilā lietotne – lietotnes paveids, kas apvieno divas tehnoloģijas – vairākplatformu tehnoloģiju bāzi kā tīmekļa implementāciju, un dzimto mobilo lietotņu tehnoloģijas

HTML5 – Valoda, kas, izmantojot speciālus kodus, nosaka hiperteksta dokumenta atveidojumu displeja ekrānā gadījumā, ja tiek lietotas interneta globālā tīmekļa lappuses.

Lietojamība – Sistēmas īpašība, kas raksturo, cik viegli lietotājs var apgūt tās izmantošanu, sagatavot tai ieejas datus un interpretēt tās izejas datus.

Mērogojamība – Sistēmas programmatūras vai aparatūras spēja normāli funkcionēt gadījumā, ja tās funkcionēšanas apjoms vai konfigurācija tiek mainīta, piem., lietotāju skaita palielināšana nesamazina sniegto pakalpojumu apjomu, pāreja uz citu operētājsistēmu netraucē normālu sistēmas funkcionēšanu.

SDK – Palīgprogrammu (rutīnu) kopa, kas lietojumprogrammu izstrādātājam atvieglo programmu veidošanu, ievērojot konkrētās to darbības vides īpatnības (piem., grafisko lietotāja saskarni, operētājsistēmu, datu bāzu pārvaldības sistēmu u.c.).

Uzturamība – Objekta spēja saglabāt vai atjaunot stāvokli, kādā tas var veikt prasīto funkciju, ja uzturēšanu veic, ievērojot noteiktus nosacījumus un izmanto noteiktas procedūras un resursus.

Veiktspēja – Sistēmas vai tās komponentu spēja izpildīt paredzētās funkcijas; par kvantitatīviem kritērijiem parasti izmanto atbildes laiku, caurlaidspēju un izmantojamību.

Xcode – „Apple'' izstrādāta programmatūras izstrādes vide, kura ļauj veidot, kompilēt un testēt programmatūru Mac OS X un iOS operētājsistēmām.

IEVADS

Mobilo lietotņu tirgus ir plašs un ar katru nākamo gadu lietotņu skaits strauji pieaug. 2014. gadā 80% pasaules iedzīvotājiem bija vismaz viens viedtālrunis un 2015. gadā ir plānots pārdot vienu miljardu viedtālruņu, kas ir divas reizes vairāk kā plānotais personālo datoru pārdošanu skaits. Viedtālruņu lietotājs vidēji 30 stundas mēnesī lieto viedtālruni, izmantojot vidēji 20 lietotnes mēnesī [Clifford2014].

Populārākais hibrīdo mobilo lietotņu implementācijas veids ir tīmekļa implementācija. Tīmekļa implementācija nodrošina lietotnes bāzes veidošanu, izmantojot tīmekļa tehnoloģijas kā *HTML5*, *JavaScript*, *CSS* u.c., kā arī nodrošina pieeju ierīču aparatūras saskarnei. Implementācija sastāv no divām pamatkomponentēm – tīmekļa pārlūka un "tilta" starp pārlūku un mobilo operētājsistēmu [GeoSpatial2012].

Tīmekļa izstrādes pieeja nav vienīgā izstrādes pieeja hibrīdām mobilām lietotnēm. Bitkoda pieeja ir pieeja, kurā kompilators augsta līmeņa programmēšanas valodu pārtulko uz dzimtās platformas zema līmeņa valodu – bitkodu. Bitkoda pieejā izmanto divas tulkošanas metodes – *JIT* (*Just-In-Time*) kompilāciju vai *AOT* (*Ahead of time*) kompilāciju.

Šī bakalaura darba mērķis ir veikt *HTML5* un *Adobe AIR* hibrīdo mobilo lietotņu izstrādes platformu salīdzinājumu, kurā tiktu vērtētas abu platformu kopīgās/atšķirīgās īpašības, priekšrocības/trūkumi, kā arī, veikti praktiski lietotņu salīdzinājumu testi.

Darba uzdevumi:

- Veikt literatūras apskatu par hibrīdo mobilo lietotņu izstrādes īpatnībām un pieejām,
- Izvēlēties *HTML5* un *Adobe AIR* platformu ietvarus,
- Izstrādāt mobilo lietotni *HTML5* un *Adobe AIR* platformām,
- Veikt individuālo izstrāžu platformu novērtēšanu,
- Apkopot rezultātus un veikt secinājumus,
- Publiskot mobilās lietotnes.

Lai detalizēti salīdzinātu abas izstrādes platformas – *HTML5* un *Adobe AIR*, tika izstrādāta neliela mobilā lietotne abās izstrādes platformās. Tas ļāva veiksmīgāk izpētīt abu platformu izstrādes iespējas, veikt veiktspējas testēšanu un biznesa aspekta salīdzinājumu. Tika iegūti secinājumi par platformu priekšrocībām un trūkumiem. Mobilās lietotnes ir publicētas *Google Play* lietotņu veikalā un ir atrodamas pēc atslēgvārda "Prātnieks".

Katra izstrādes platforma tika novērtēta pēc *Henning Heitkötter* vairākplatformu salīdzinājuma kritērijiem [Heitkötter2013], kas veic platformu vērtēšanu no izstrādes un infrastruktūras perspektīvas. Pamatojoties uz individuālo izstrādes platformu novērtējumu, tika veikts abu platformu vērtējuma salīdzinājums un iegūti secinājumi. Gala rezultāti atspoguļo abu platformu salīdzinājumu, priekšrocības/trūkumus un kopīgās/atšķirīgās īpašības.

Dokuments ir sadalīts vairākās nodaļās: mobilās lietotnes un to attīstība, hibrīdo mobilo lietotnes, hibrīdo un dzimtās mobilo lietotņu izstrādes pieejas salīdzinājums, mobilās lietotnes izstrāde HTML5 un Adobe AIR vidēs, HTML5 un Adobe AIR platformas novērtēšana. Pielikums satur HTML5 un Adobe AIR mobilo lietotņu koda fragmentus, licences.

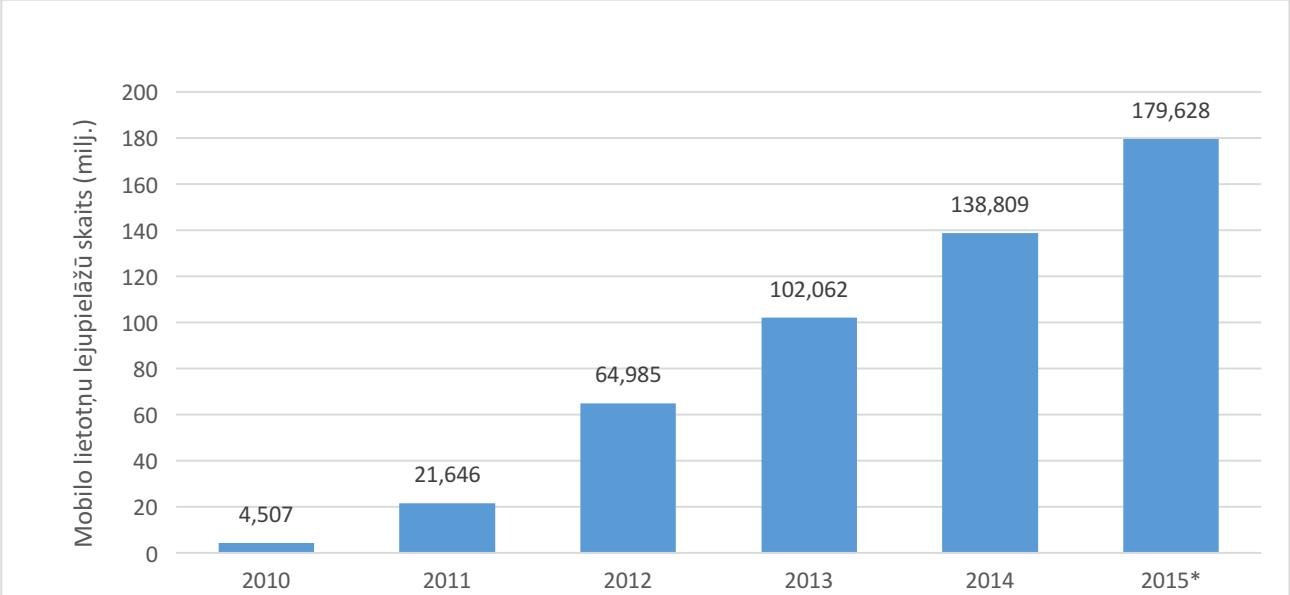
1. MOBILĀS LIETOTNES UN TO ATTĪSTĪBA

Mobilās lietotnes ir lietotnes, kuras ir projektētas mazām, rokas izmēra iekārtām kā viedtālruņiem, planšetdatoriem, plaukstdatoriem, e-lasītājiem u.c. iekārtām. Mobilās lietotnes mērķis ir izklaidēt, izglītot vai atvieglot lietotāja ikdienas dzīvi [Viswanathan2014]. Tās ir iespējams uzstādīt no personālā datora, izmantojot USB vai bezvadu savienojumu, vai lejupielādēt no lietotņu tirgus kā *Appstore, Google Play store*, u.c. tirgiem.

1.1. Mobilo lietotņu attīstība

Mobilo lietotņu tirgus ir plašs un ar katru nākamo gadu lietotņu skaits strauji palielinās. 2014. gadā 80% pasaules iedzīvotājiem bija vismaz viens viedtālrunis un 2015. gadā ir plānots pārdot vienu miljardu viedtālruņu, kas ir divas reizes vairāk kā plānotais personālo datoru pārdošanas skaits. Viedtālruņu lietotājs vidēji 30 stundas mēnesī lieto viedtālruni, izmantojot vidēji 20 lietotnes mēnesī [Clifford2014].

Zemāk ir aplūkojams lietotņu lejupielāžu skaits laika posmā no 2010. līdz 2015.gadam.

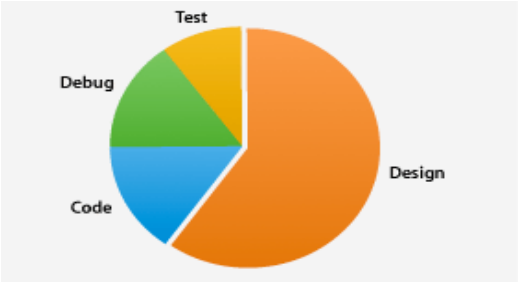


1.1.1. att. **Mobilo lietotņu lejupielāžu skaits laika posmā no 2010. līdz 2015.gadam**

Pēc statiskajiem datiem var secināt, ka mobilo lietotņu tirgus strauji attīstās un palielinās, piemēram, no 2010. līdz 2014. gadam mobilo lietotņu skaits ir palielinājis 30 reizes [Statista2015a]. Mobilo lietotņu ienākumi 2010. gadā bija 8.32 miljardi ASV dolāru un 2014. gadā plānotie ienākumi ir 35 miljardi ASV dolāru, kas ir četras reizes vairāk nekā 2010. gadā [Statista2015b].

1.2. Mobilo lietotņu izstrādes cikls

Programmatūras izstrāde ir sarežģīta, tādēļ, lai atvieglotu izstrādes procesu, to var sadalīt vairākos mazākos posmos. Zemāk ir aprakstīti soļi, kurus nepieciešams veikt, lai veiksmīgi izstrādātu jebkuru mobilo lietotni.



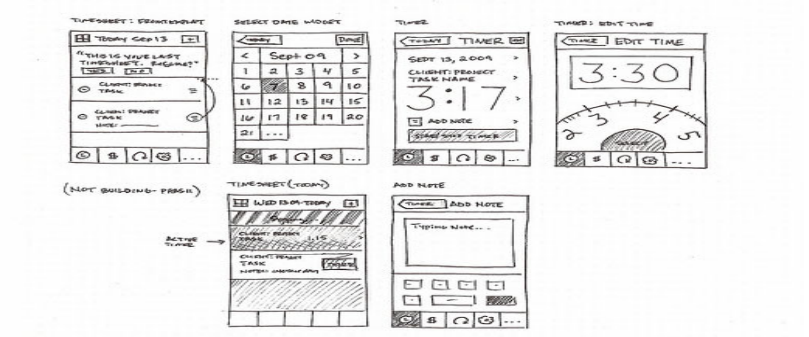
1.2.1. att. **Mobilās lietotnes izstrādes cikls**

1.2.1. Izstrādāt lietotnes koncepciju

Lai izstrādātu lietotnes koncepciju, ir nepieciešams apsvērt uzdevumu, kādu risinās lietotne. Jebkura lietotne risina konkrēti definētu uzdevumu, piemēram, iestatījuma lietotne ļauj lietotājam rediģēt viedtālruņa iestatījumus [Apple2014a] .

Jautājumi, kurus ir nepieciešams apsvērt koncepcijas izstrādē [Apple2014a] :

- Auditorija – lietotnes saturs un iespējas būs atkarīgas no mērķauditorijas;
- Lietotnes mērķis – ir nepieciešams definēt lietotnes mērķi, kas motivētu lietotājus lietot lietotni;
- Kādu uzdevumu risina lietotne – lietotnei vajadzētu risināt vienu, konkrētu uzdevumu, nevis daudzus, dažādus uzdevumus;
- Lietotnes saturs – ir nepieciešams apsvērt, kāds būs lietotnes saturs un kā lietotājs ar to savstarpēji mijiedarbosies.

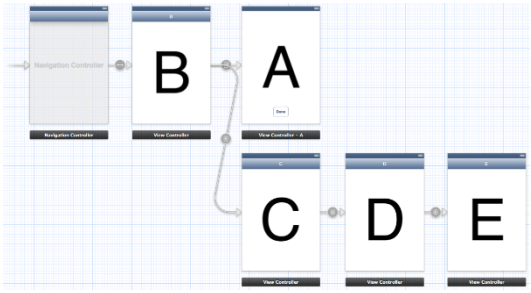


1.2.1.1. att. ***iOS*** **koncepcijas piemērs**

Izstrādes koncepcijai nav nepieciešams būt sarežģītai un pabeigtai izstrādes sākumā, bet tas palīdz saprast turpmāko izstrādes gaitu [Apple2014a].

1.2.2. Izstrādāt lietotāja interfeisu

Pēc koncepcijas izstrādes, ir nepieciešams izstrādāt lietotāja interfeisu. Lietotājam ir jāspēj veikt darbības pēc iespējas vienkāršākā veidā. Darbībām ir jābūt vienkāršām, nepārprotamām un efektīvām. Lai palīdzētu izprast koncepcijas un dizaina mijiedarbības procesu, nepieciešams izveidot grafisku lietotnes skatu diagrammu. Skatu diagrammas ļauj izprast izstrādes gaitu un saņemt tūlītēju atgriezenisko saiti par izstrādes nepilnībām, kas ļauj tās izlabot [Apple2014a].



1.2.2.1. att. ***iOS*** **skata diagrammas piemērs**

1.2.3. Definēt lietotāja mijiedarbību ar lietotni

Pēc lietotāja interfeisa izstrādes ir nepieciešams definēt kā lietotājs mijiedarbojas ar lietotnes saturu. Jebkura mobilā lietotne ir bāzēta uz notikumu programmēšanu. Notikumu programmēšana ir programmēšana, kad lietotne saņem notikumu paziņojumus un tie tiek apstrādāti, piemēram, lietotājs nospiež pogu. Kad lietotājs veic kādu darbību ar interfeisu, lietotne saņem notikuma paziņojumu. Šo notikumu pamatā tiek veidota lietotnes loģika un manipulācija ar datiem. Lietotne atbild uz lietotāja darbībām caur lietotāja interfeisu. Visus notikumus nepieciešams apstrādāt skata vadīklā [Apple2014a].

1.4. Viedtālruņu operētājsistēmas

Mobilā operētājsistēma ir operētājsistēma, kura ir izstrādāta mobilajām iekārtām kā viedtālruņiem, planšetdatoriem, plaukstas datoriem un citām iekārtām [Beal2014a]. Pēc statistikas datiem par 2014. gada populārākajām mobilajām operētājsistēmām var secināt, ka pašlaik pasaulē dominē četras mobilās operētājsistēmas – *Android*, *iOS*, *Windows Phone* un *BlackBerry*. Izstrādājot mobilās lietotnes, ir nepieciešams pievērst uzmanību tieši šīm operētājsistēmām, lai nodrošinātu pēc iespējas lielāku lietotāju bāzu pārklājumu. Zemāk ir iespējams aplūkot mobilo operētājsistēmu sadalījumu pēc to izmantošanas biežuma [IDC2014].

1.4.1. tabula

Viedtālruņu operētājsistēmu sadalījums

Gads	*Android*	*iOS*	*Windows Phone*	*BlackBerry OS*	Citi
2014	76.6%	19.7%	2.8%	0.4%	0.5%
2013	78.2%	17.5%	3.0%	0.6%	0.8%
2012	70.4%	20.9%	2.6%	3.2%	2.9%
2011	52.8%	23.0%	1.5%	8.1%	14.6%

Pēc viedtālruņu operētājsistēmu sadalījuma var secināt, ka vairākus gadus vadošā loma ir *Android* operētājsistēmai, kam seko *iOS* un *Windows Phone*, savukārt, *BlackBerry OS* un citas viedtālruņu operētājsistēmas ir piedzīvojušas straujumu kritumu pasaules tirgū.

1.4.1. *Android* operētājsistēma

Android ir atvērtā pirmkodu operētājsistēma, kura ir veidota uz *Linux* kodola bāzes un ir paredzēta skārienjutīgajām mobilajām tehnoloģijām, piemēram, viedtālruņiem, planšetdatoriem, izklaižu sistēmām (*Android TV*) u.c. iekārtām. To izstrādāja uzņēmums *Google* 2008. gada 23. septembrī [EngineersGarage2014, Android2015].

Android ir plaši izmantojama, jo ir viegli pielāgojama operētājsistēma klientu vajadzībām un ir atvērtā pirmkoda operētājsistēma, kas padara izstrādes procesu lētāku. Pamata izstrādātāju rīkkopa, kuru izmanto *Android* lietotņu izstrādē, ir *Android SDK*, kura ir pieejama izstrādes vidēs kā *Eclipse IDE* vai *Android Studio*.

Lielākie mobilo tehnoloģiju ražotāji, kuri izmanto *Android* operētājsistēmu – *Asus*, *Acer*, *Dell*, *HTC*, *LG*, *Motorolla*, *Samsung*. 2014. gada populārākais *Android* viedtālrunis bija *Samsung Galaxy S5* [Valdoz2014].

1.4.2. *iOS*

iOS ir operētājsistēma uzņēmuma *Apple* izstrādātajām mobilajām tehnoloģijām – *iPhone*, *iPod*, *iPad*, *Apple TV* un citām līdzīgam iekārtām. Oriģināli *iOS* tika dēvēts par *iPhone OS*, bet tika pārsaukts 2010. gadā, lai atspoguļotu operētājsistēmas attīstību un atbalstītu citas *Apple* iekārtas. *iOS* ir veidota uz citas *Apple* operētājsistēmas bāzes – *OS X*, izmantojot kodolu no *OS X* operētājsistēmas, bet nodrošinot savu lietotāja interfeisa struktūru [Beal2014b]. Šī operētājsistēma nav atvērtā pirmkoda operētājsistēma kā *Android*, tādēļ *iOS* operētājsistēma ir sastopama tikai un vienīgi *Apple* iekārtās.

2014. populārākais viedtālrunis bija *iPhone 5S*, kas atbalsta *iOS* operētājsitēmu [Valdez2014]. Šī brīža pēdējā *iOS* versija ir 8.0, kas piedāvā iespēju komunicēt starp citām *iOS8* vai *Mac OSX* iekārtām (pārsūtīt informāciju, ļaut atbildēt uz telefona zvaniem/ paziņojumiem).

1.4.3. *Windows Phone* operētājsistēma

Windows Phone operētājsistēma ir mobilo iekārtu operētājsistēma, kuru izstrādāja uzņēmums *Microsoft*. *Windows Phone* piedāvā jaunu lietotāja interfeisa veidu – *Metro* lietotāja interfeiss (*Metro UI*), kuru vēlāk pārdēvēja par moderno lietotāja interfeisu (*Modern UI*), kas ir sastopams arī *Windows 8* operētājsistēmā. *Metro* lietotāja dizains ietver lielāku sākumekrānu, kas sastāv no dinamiskiem lietotņu rajoniem, kurus pielāgo attiecīgā lietotne.

Zemāk ir aplūkojams *Metro* lietotāja interfeisa sākuma skats.

1.4.3.1. att. **Metro lietotāja interfeisa sākuma skats**

Lielākie mobilo tehnoloģiju ražotāji, kuri izmanto *Windows Phone* operētājsistēmu – *Microsoft Phone* (*Nokia*), *Samsung*, *HTC*, *LG*.

2. HIBRĪDU MOBILĀS LIETOTNES

Mobilo lietotņu tirgus strauji izplešas un lielā mobilo lietotņu pieprasījuma dēļ daudzi uzņēmumi izvēlas mobilo lietotni kā reklāmu, kas veicinātu uzņēmuma izaugsmi. Pašlaik tirgū dominē trīs mobilās operētājsistēmas – *Android*, *iOS* un *Windows Phone*. Tas nozīmē, ka katrai mobilajai operētājsistēmai nepieciešama sava lietotne – kopumā trīs dažādas lietotnes, kas izstrādes procesu padara dārgāku trīs reizes.

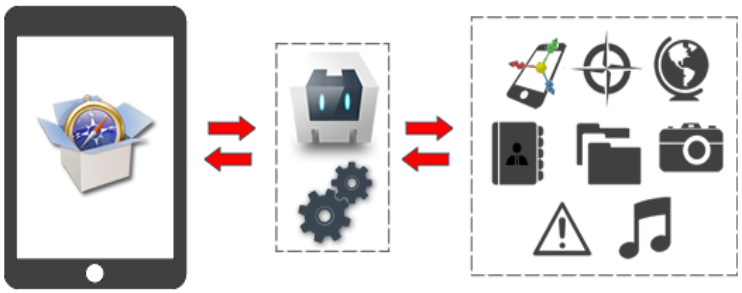
Hibrīdā mobilā lietotne ir lietotnes paveids, kas apvieno divas tehnoloģijas – vairākplatformu tehnoloģiju bāzi kā tīmekļa implementāciju, un dzimto mobilo lietotņu tehnoloģijas [Rouse2014]. Tas nozīmē, ka visām mobilajām operētājsistēmām tiek nodrošināta viena implementācija (lietotnes izstrādes kods) ar iespēju izmantot dzimto mobilo lietotņu saskarni (API). Hibrīdās mobilās lietotnes ir pieprasītas vairākplatformu iespēju un zemo izmaksu dēļ [Chell2015].

2.1. Hibrīdo mobilo lietotņu izstrāžu pieejas

Hibrīdajām mobilajām lietotnēm pasaulē ir dažādas izstrādes pieejas. Populārākās hibrīdo mobilo lietotņu izstrāžu pieejas ir tīmekļa izstrādes pieeja un bitkodu izstrādes pieeja [Raj2014]. Tīmekļa izstrādes pieeja balstās uz tīmekļa tehnoloģijām un implementācijām (tiek izmantots tīmekļa pārlūks), savukārt, bitkoda izstrādes pieeja veic augsta līmeņa programmēšanas valodu (piemēram, *Adobe ActionScript* vai *C#*) kompilāciju uz operētājsistēmai saprotamu mašīnkodu.

2.1.1. Tīmekļa pieeja

Populārākais hibrīdu implementācijas veids ir tīmekļa implementācija. Tīmekļa implementācija nodrošina, ka lietotnes bāze tiek veidota, izmantojot tīmekļa tehnoloģijas kā *HTML5*, *Java Script*, *CSS* u.c., kā arī nodrošina pieeju ierīču aparatūras saskarnei. Implementācija sastāv no divām pamatkomponentēm – tīmekļa pārlūka un "tilta" starp pārlūku un mobilo operētājsistēmu [GeoSpatial2012].



2.1.1.1. att. **Tīmekļa implementācijas pieeja** [PhoneGap2012]

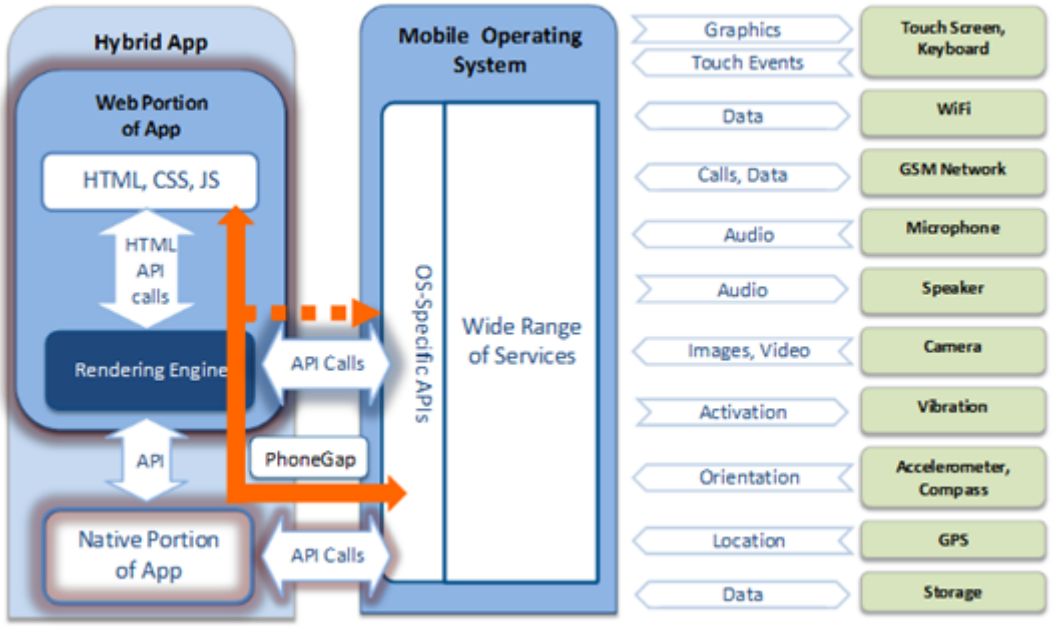
Parasti mobilā lietotne sastāv no vairākiem skatiem, veicot navigāciju pa sadaļām, bet tīmekļa implementācijai tiek izveidots tikai viens pamatskats, kas satur tikai skata komponenti – tīmekļa pārlūku, kas ir standarta skata komponente visās mobilajās operētājsistēmās.

Tīmekļa pārlūks ir galvenais lietotāja interfeiss, ar kura palīdzību notiek komunikācija starp lietotāju un lietotni [GeoSpatial2012, Chell2015].

Tīmekļa pārlūks nespēj tieši komunicēt ar mobilo operētājsistēmu, jo katrai mobilajai operētājsistēmai ir savs lietotnes interfeiss. Starp tīmekļa pārlūku un operētājsistēmu tiek izveidots "tilts" (piemēram, *PhoneGap* ietvars), kas nodrošina komunikāciju starp pārlūku un operētājsistēmu. Ietvara loma ir *JavaScript* komandas pārveidot uz dzimto kompilatora valodu un otrādi jeb notiek *JavaScript* komandu tulkošana uz kompilatoram saprotamu valodu. Ietvars strādā arī pretējā virzienā, kas nozīmē, ka operētājsistēmas spēj nosūtīt ziņu pārlūkam, piemērām, pēc dzimtās platformas ziņojuma saņemšanas [PhoneGap2012, Chell2015].

Gala produkta izstrādē tomēr nepieciešami dzimtās platformas lietotnes kompilatori, kas hibrīdo mobilo lietotni pārveido uz bināru failu, kuru pēc tam var publicēt lietotņu veikalos vai uzstādīt uz viedtālruņiem kā dzimto platformu mobilo lietotni.

Zemāk ir aplūkojams tīmekļa implementācijas izstrādes princips.



2.1.1.2. att. **Tīmekļa pieejas detalizēts izstrādes** princips [GeoSpatial2012]

Hibrīdā mobilā lietotne iekļauj tīmekļa pārlūku un ietvaru, kas nodrošina komunikācijas iespējas starp tīmekļa pārlūku un operētājsistēmu kā viedtālruņu kameras vai GPS izmantošana.

2.1.1.1 *HTML5* platforma

HTML jeb hiperteksta iezīmēšanas valoda ir vissvarīgākais tīmekļa elements. Šī valoda apraksta kā tīmekļa lapai ir jāizskatās. *HTML* vājā puse ir tās statiskā pieeja – lai nodrošinātu papildus tīmekļa iespējas, nepieciešams izveidot tīmekļa spraudņus kā *CSS*, *Flash*, *Java*, *Silverlight* u.c. Piemēram, daudzi video atskaņotāji tiek veidoti *Flash* platformā vai tīmekļa lietotnes *Java* platformā. Tā kā tīmekļa lapa sastāv no daudziem šādiem spraudņiem, tad izstrādātājiem ir grūti nodrošināt tīmekļa lapas standartus kā rezultātā dažādi tīmekļa pārlūki uzrāda informāciju dažādi. Būtisks ir aspekts , ka spraudņi patērē procesora un atmiņas resursus, kuri mobilās iekārtās ir stipri ierobežoti [Lee2013, Graham2012, Bunyan2015].

Tā kā *HTML* pamatdarbības ir pietiekošas, bet ar dažādiem ierobežojumiem, ir izstrādātā jauna *HTML* versija – *HTML5*. *HTML5* apvieno četras pamattīmekļa komponentes – *HTML*, *CSS*, *DOM* un *JavasScript*. *HTML5* mērķis ir samazināt atkarību no spraudņiem, aizstāt izpildes skriptus ar birkām, mazināt iekārtu atkarību (visām iekārtām ir nepieciešams uzrādīt vienādu tīmekļa lapas saturu) un pievienot citus būtiskus uzlabojumus [Lūse2013, Lee2013]. *HTML5* piedāvā šādas jaunas birkas (*tags*):

- **<header>** un **<footer>** birkas, kas palīdz nodalīt galvenes un kājenes satura blokus no kopēja satura, un kuras var lietot vairākas reizes vienā tīmekļa lapā,
- **<article>** birka, kas atdala satura bloku kā komentāru vai memuāru ierakstu,
- **<audio>** un **<video>** birka, kas ļauj iekļaut audio un video saturu,
- **<canvas>** birka, kas ļauj zīmēt grafisku saturu,
- u.c. birkas.

2.2.1.1.1. att. **HTML5 platformas logo**

Tīmekļa lapas, kuras izmanto *HTML5*, nodrošina labāku veiktspēju, jo efektīvāk nodrošina tīmekļa standartus. *HTML5* piedāvā plašas multimediju iespējas, aizstājot tādu platformu kā *Flash* [Lūse2013]. Sīkāk ar *HTML5* iespējam ir iespējams iepazīties oficiālajā tīmekļa lapā **w3schools.com.**

2.1.1.2 *Google Dart* platforma

Dart ir pielāgojama platforma, kas ļauj izstrādāt tīmekļa lietotnes *Dart* programmēšanas valodā. *Dart* programmēšanas valoda ir objektorientēta programmēšanas valoda, kuru izstrādāja uzņēmums *Google* 2011. gadā. *Dart* mērķis ir aizstāt *JavaScript* programmēšanas valodu ar modernu, objektorientētu programmēšanas valodu. *JavaScript* programmēšanas galvenā problēma ir tās struktūrā, dokumentācijā un standartos, kas ierobežo izstrādātāju iespējas [Clark2011, Dart2015].

Dart piedāvā plašākas iespējas nekā *JavaScript* programmēšanas valoda, kas uzlabo tīmekļa lapu veiktspēju un kvalitāti.

Zemāk ir uzskaitītas dažas no *Dart* piedāvātām iespējām [Clark2011]:

- Klases – *Dart* klases un mantošanas iespējas ļauj izstrādātājiem strukturizēt programmatūras kodu, metožu atkal izmantošu u.c. iespējas,
- Iespējamās (*optional*) vērtības – atšķirībā no *JavaScript*, *Dart* ļauj izstrādātājiem izveidot statiskus tipus programmatūrā,
- Bibliotēkas – bibliotēkas, kuras izmanto *Dart* nemainās izpildes brīdī,
- Rīki – *Dart* ir iespējams izstrādāt izstrādes vidē *Dart Editor*, kas ļauj izstrādāt un testēt *Dart* lietotnes.

Zemāk ir aplūkojams *Dart* programmatūras koda piemērs [Dart2015].

```
import 'dart:math' show Random;        // Import a class from a library.
void main() {                          // The app starts executing here.
  print(new Die(n: 12).roll());  // Print a new object's value. Chain method calls
}
class Die {                            // Define a class.
  static Random shaker = new Random(); // Define a class variable.
  int sides, value;                    // Define instance variables.
  String toString() => '$value ';      // Define a method using shorthand syntax.
  Die({int n: 6}){                   // Define a constructor.
    if (4 <= n && n <= 20) {
      sides = n;
    } else { throw new ArgumentError(/* */);}  // Support for errors and exceptions.
}
```

2.2.1.2.1. att. **Google Dart programmatūras koda piemērs**

Dart programmatūras kods, kas tiek izmantots tīmekļa lapās, tiek pārkompilēts *JavaScript* programmēšanas valodā, izmantojot *dart2js* kompilatoru, kas ļauj to izmantot daudzos tīmekļa pārlūkos [Dart2015].

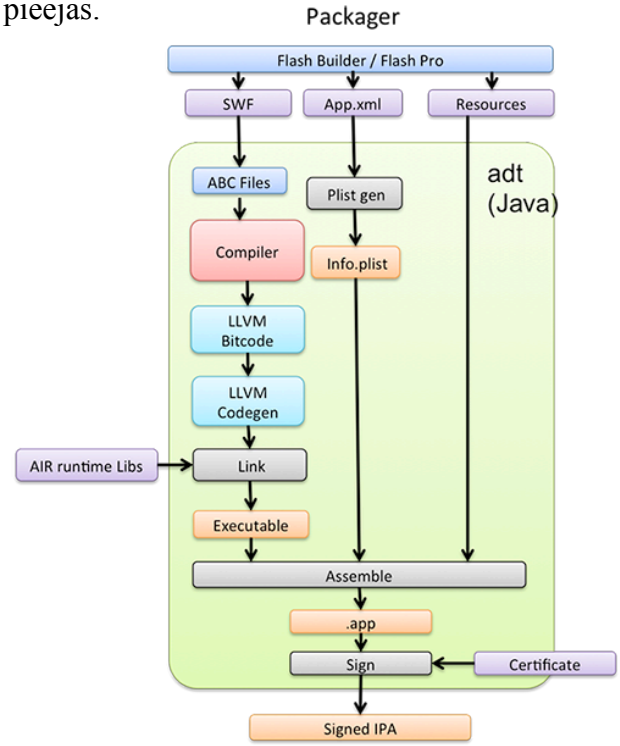
2.1.2. Bitkoda pieeja

Tīmekļa izstrādes pieeja nav vienīgā izstrādes pieeja hibrīdu mobilām lietotnēm. Bitkoda pieeja ir pieeja, kurā kompilators augsta līmeņa programmēšanas valodu pārtulko uz dzimto platformu zema līmeņa valodu – bitkodu. Bitkoda pieejā izmanto divas tulkošanas metodes – *JIT*(*Just-In-Time*) kompilāciju vai *AOT* (*Ahead of time*) kompilāciju.

JIT (*Just-In-Time*) kompilācija nozīmē, ka augsta līmeņa izpildes kods tiek pārveidots uz zema līmeņa mašīnkodu. Pārveidošanu ir iespējams veikt pirms programmatūras izpildes vai programmatūras izpildlaika brīdī [Kshitiz2012, Bansod2011]. Šo pieeju izmanto *Android* un *Windows Phone* operētājsistēmas, kuras atbalsta izpildlaika sistēmu kodu ģenerēšanu.

iOS mobilās lietotnes neatbalsta izpildlaiku sistēmu kodu ģenerēšanu, tādēļ tiek izmantota *AOT* (*Ahead of Time*) kompilācija, kas pārveido augsta līmeņa valodu uz zema līmeņa valodu (piemēram, *ARM* mašīnkodu) produkcijas izstrādes brīdī un izpildlaika sistēma tiek pievienota pārveidošanas brīdī, kas palielina lietotnes fizisko izmēru [Kshitiz2012, Philomena2014, Bansod2011, Adobe2011].

Zemāk ir iespējams aplūkot *Adobe AIR* platformas *iOS* mobilās lietotnes izveides ilustrāciju pēc bitkoda pieejas.

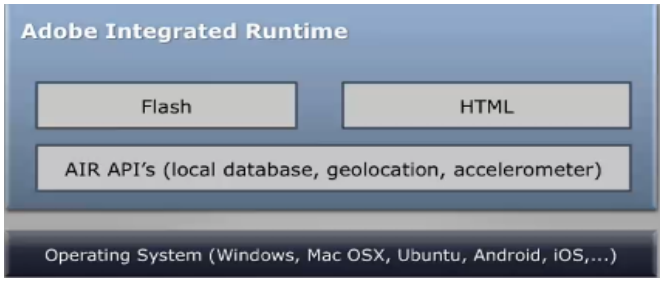


2.1.2.1. att. ***Adobe AIR iOS*** **lietotnes izstrādes vizualizācija pēc bitkoda pieejas** [Philomena2014]

Lietotnes izstrādē tiek veikti vairāki procesi kā bitkodu (*bytecode*) izveide un *iOS* mobilās lietotnes izveide no *ARM* mašīnkoda. Tā kā procesa posmu ir samērā daudz, tad gala lietotnes izveide notiek lēni.

2.1.2.1. *Adobe AIR* platforma

Adobe AIR jeb *Adobe* Integrētā izpildlaika sistēma ir daudz platformu izpildlaika sistēma, kuru izstrādāja uzņēmums *Adobe Systems* 2008. gadā. Sistēma ļauj izstrādāt darbvirsmu vai mobilās lietotnes, izmantojot *Adobe Flash* dzini un *Action*, *Flex* programmēšanas valodas. *Adobe AIR* lietotnes satur *Flash* atskaņotāja instanci, kura darbojas ārpus tīmekļa pārlūka. Tomēr izpildlaika sistēma nenodrošina dzimtās platformas grafiskā lietotāja interfeisa (*GUI*) izmantošanu kā navigāciju vai dažādas kontroles. Izstrādājot *Flash* lietotnes, *Adobe AIR* ļauj izmantot funkcijas kā datņu sistēmas integrēšanu dzimto sistēmu spraudņiem, iekārtu interfeisa izmantošanu (piemēram, akselerometra datu vai GPS izmantošanu), datu bāžu izveidi u.c. iespējas. Zemāk ir aplūkojama *Adobe AIR* darbības principa vizualizācija [Adobe2015a, Graham2012, Adobe2011].



2.1.2.1.1. att. **Adobe AIR darbības principa vizualizācija**

Adobe AIR lietotņu pieejamība

Adobe AIR pašlaik nodrošina lietotnes šādām operētājsistēmām – *Windows*, *Mac OS X*, *Android*, *iOS* un *BlackBerry OS*. Lietotņu izveidē ir pieejami divi režīmi – lietotne iekļauj sevī izpildlaika sistēmu (*AIR*) vai neiekļauj (izmanto ārēji). Operētājsistēmas kā *iOS* nenodrošina iespēju piekļūt ārēji *Adobe AIR* sistēmai, kas nozīmē, ka pati sistēma ir jāiekļauj lietotnē, bet, savukārt, *Android* operētājsistēma nodrošina iespēju piekļūt ārēji *Adobe AIR* sistēmai. Iekļaujot sistēmu lietotnē, tiek palielināts kopējais lietotnes atmiņas daudzums.

Zemāk ir aplūkojama *Adobe AIR* operētājsistēmu pieejamības tabula [Adobe2015a].

2.1.2.1.1. tabula

***Adobe AIR* operētājsistēmu pieejamības** [Adobe2015a]

Platforma	Datņu paplašinājumi	Pieejami lietotņu veikali
Windows	.air, .exe, .msi	-
Mac OS X	.air , .dmg	AppStore
Android	.apk	Google Play
iOS	.ipa	AppStore
BlackBerry OS	.bar	BlackBerry App World

Pēc pieejamības tabulas var secināt, ka *Adobe AIR* sistēma ļauj izveidot lietotnes operētājsistēmu dzimto lietotņu formātā.

2.1.2.2. *Xamarin* platforma

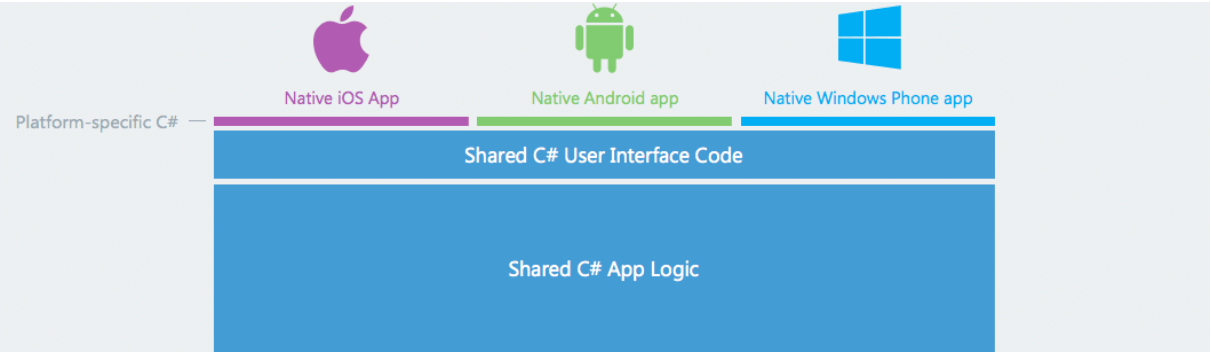
Xamarin 2.0 platformu izstrādāja uzņēmums *Xamarin* 2013. gada februārī. Tika apvienoti vairāki *Xamarin* izstrādes produkti – *iOS*, *Mac OS X* un *Android* izstrādes rīki, tehnoloģijas. *Xamarin.iOS* un *Xamarin.Android* ļauj izstrādāt hibrīdās mobilās lietotnes, izmantojot programmēšanas valodu C# *Xamarin Studio* vai *Visual Studio* izstrādes vidēs. Izstrādāto lietotnes kodu ir iespējams izmantot atkārtoti visās piedāvātajās operētājsistēmās [Xamarin2015].

Xamarin Studio, kas ir *Xamarin* mobilo lietotņu izstrādes vide, piedāvā ērtu lietotņu izstrādi, nodrošinot atkļūdošanas iespējas, hibrīdo mobilo lietotņu kompilēšanu un lietotņu skatu izveidi, neizmantojot skatu *XML* kodu ģenerēšanu [Xamarin2015].

Xamarin sastāv no dažādiem izstrādes elementiem [Xamarin2015]:

- *C#* programmēšanas valodas,
- *Mono. NET* ietvara (*framework*) – nodrošina daudz platformu implementāciju, izmantojot *Microsoft's .NET* ietvaru,
- Kompilatoru – *iOS* un *Android* lietotņu kompilēšana notiek dažādi, izmantojot *AOT* (*Ahead of time*) vai *JIT* (*Justi-In-Time*) kompilāciju,
- Izstrādes vides (*IDE*) rīku – *Xamarin Studio*.

Zemāk ir aplūkojama *Xamarin* platformas izstrādes pieeja.



2.1.2.2.1. att. **Xamarin platformas izstrādes pieeja** [Xamarin2015]

Xamarin platformas priekšrocība ir izstrādes koda atkal izmantošana dažādās platformās, neatkarīgi no tā, ka implementācija katrai sistēmai ir dažāda.

Kompilācija

Xamarin platforma hibrīdās mobilās lietotnes izstrādā pēc bitkoda pieejas – augsta līmeņa programmēšanas valoda (C#) tiek kompilēta uz zema līmeņa valodu – mašīnkodu. Tomēr implementācija katrai operētājsistēmai ir dažāda.

Lai izveidotu *iOS* lietotni, C# programmēšanas valoda tiek kompilēta, izmantojot *AOT* (*Ahead-of-time*) kompilāciju uz *ARM* mašīnkodu. *.NET* ietvars tiek pievienots kompilēšanas brīdī un tiek izņemtas neizmantotās klases, kas samazina lietotnes fizisko izmēru. *.NET* ietvars tiek pievienots kompilācijas brīdī, jo *iOS* operētājsistēma neatbalsta izpildlaika (*runtime*) kodu ģenerēšanu [Xamarin2015].

Lai izveidotu *Android* lietotni, C# programmēšanas valoda tiek kompilēta uz *IL* (*Intermediate Language*) programmēšanas valodu, izmantojot *JIT* (*Just-in-time*) kompilāciju. Neizmantotās klases tiek izņemtas ārā, kas samazina lietotnes fizisko izmēru kompilācijas brīdī. Kompilēto kodu pēc tam izmanto *Dalvik* (*Android* sistēmas virtuālā mašīnā) kā dzimto *Android* sistēmas lietotni [Xamarin2015].

Tā kā *Windows Phone* dzimtā lietotņu valoda ir tieši C#, tad C# programmēšanas valoda tiek nokompilēta uz *IL* (*Intermediate Language*) programmēšanas valodu, kuru pēc tam izmanto iebūvētā izpildlaika sistēma (*runtime*) [Xamarin2015].

3. HIBRĪDO UN DZIMTO MOBILO LIETOŅU IZSTRĀDES PIEEJU SALĪDZINĀJUMS

Mobilo lietotņu izstrāde ir aktuāls jautājums – vai mobilo programmatūru izstrādāt, izmantojot dzimtās platformas vai tīmekļa (hibrīda) tehnoloģijas. Izstrādājot mobilo lietotni ar dzimtās platformas pieeju, tiek nodrošināta augstāka lietotnes veiktspēja, bet hibrīdo mobilo lietotņu izmaksas ir zemākas, salīdzinot ar dzimto platformu lietotnēm. Izmantojot "*DZone's mobile research portal*" datus, zemāk ir aprakstīti pamatfaktori, pēc kuriem salīdzina konkrētās pieejas priekšrocības un trūkumus [DZone2014, Brown2014].

3.1. Izmaksas

Dzimtās platformas lietotnes parasti rada augstākas izmaksas nekā hibrīdās mobilās lietotnes. Ja ir nepieciešams nodrošināt vairākplatformu lietotni, tad ir nepieciešami programmētāji ar zināšanām dažādās izstrādes tehnoloģijās kā *iOS*, *Android*, *Windows Phone* u.c. Tomēr izmaksas ir atkarīgas arī no citiem faktoriem, tāpēc izmaksas faktors nav noteicošais faktors platformas izvēlē, jo abos gadījumos nepieciešamas augsti kvalificēti izstrādātāji [Pronschinske2014].

3.2. Testējamība

Dzimtā platforma ar profilēšanas rīkiem nodrošina vieglāku testēšanu nekā ietvari, kuri izstrādā hibrīdās mobilās lietotnes. Hibrīdo mobilo lietotņu testēšana var būt apgrūtināta, ja izstrādātājs nepārzina *JavaScript* izstrādes ekosistēmu, bet dzimtās platformas lietotnes testēšanas grūtības pakāpe pieaug atkarībā no bāzes kodu skaita un atbalstāmo iekārtu skaita [Pronschinske2014].

3.3. Koda atkalizmantojamība/Pārnesamība

Iespējams lielākais dzimtās platformas trūkums ir izstrādes koda pārnesamība uz citām platformām. Hibrīdās mobilās lietotnes priekšrocība ir tā, ka tiek nodrošināta pamatizstrādes koda bāze visām platformām. Tomēr ne visas hibrīdās mobilās lietotnes 100% nodrošina koda pārnesamību – jaunākie tīmekļa standarti netiek atbalstīti visu platformu tīmekļos visās iekārtās, tāpēc hibrīdās mobilās lietotnes izstrādātājiem arī ir jāpievērš uzmanība platformu saderībai [Pronschinske2014].

3.4. Saskarne ar iekārtas komponentēm

Dzimtā platforma nodrošina pilnīgu saskarni ar iekārtas komponentēm (*GPS*, akselerometra rādītājiem u.c. komponentēm). Hibrīdās mobilās lietotnes nodrošina pamatiespējas ar iekārtas saskarni, piemēram, akselerometra radītājus [Pronschinske2014].

3.5. Lietotāja saskarne

Mobilo tīmekļu lietotāju interfeisu ietvari nodrošina iespēju hibrīdās mobilajām lietotnēm izmantot dzimtās platformas lietotāja interfeisa komponentes, kuras nav identiskas dzimtajai platformai. Ietvariem nepieciešams vairākkārtēja atjaunošana, lai nodrošinātu svarīgākās dzimtās platformas lietotāja interfeisa izmaiņas, piemēram, *iOS 7* lietotāja interfeisa stils. Turklāt attēlu objektu zīmēšana hibrīdajās mobilajās lietotnēs tiek nodrošināta citādāk nekā dzimtajās platformās, kas būtiski samazina hibrīdo mobilo lietotņu attēla kvalitāti, jo, piemēram, *iOS* sarežģītās animācijas nav iespējams pilnvērtīgi pārveidot *JavaScript* programmēšanas valodā [Pronschinske2014].

3.6. Distribūcija

Abas platformas nodrošina vienādu lietotņu publicēšanu lietotņu portālos/veikalos. Lietotnēm ir noteikti ierobežojumi, kuri ir jāievēro veiksmīgu lietotņu publicēšanas gadījumā, pretēji, lietotne tiks noraidīta publiskai publicēšanai [Pronschinske2014].

3.7. Veiktspēja

Dzimtā platforma nodrošina tiešo piekļuvi operētājsistēmas platformas funkcionalitātēm, kas nodrošina augstāku dzimtās mobilās lietotnes veiktspēju nekā hibrīdajām mobilajām lietotnēm, kurām tiek nodrošināta atsevišķa saskarne starp operētājsistēmu un lietotni. Hibrīdās mobilās lietotnes veiktspēja ir atkarīga no dizaina sarežģītības un tehnoloģijām, piemēram, tiek atjaunoti un uzlaboti mobilo iekārtu tīmekļu pārlūki, kas palielina veiktspēju. Veiktspēja abām platformām ir atkarīga no izstrādātāja zināšanām [Pronschinske2014, Brown2014].

3.8. Lietotnes atjauninājumi

Dzimtās lietotnes atjaunošana tiek veikta ar lietotnes veikalu palīdzību. Atkarībā no lietotņu veikala, dzimtās platformas lietotnes atjaunošana prasa laiku, kas var būt no dažām minūtēm līdz vairākām dienām (*Android* – 1-2 stundu laikā, *iOS* – piecas dienas). Hibrīdās mobilās lietotnes atjaunināšanu var nodrošināt divos veidos – caur lietotņu veikalu vai veicot atjauninājumu no tīmekļa servera. Tā kā hibrīdās mobilās lietotnes nodrošina tīmekļa pārlūka tehnoloģiju izmantošanu, tad pastāv iespēja lietotnes saturu atjaunot attālināti [Pronschinske2014, Brown2014].

3.9. Kopsavilkums

Tika secināts, ka hibrīdajām mobilajām lietotnēm ir vairāk kopīgu īpašību ar dzimtās platformas lietotnēm nekā atšķirību. Zemāk ir aplūkojams dzimtās platformas un hibrīdās mobilās platformas salīdzinājuma kopsavilkums.

3.9.1. tabula

Dzimtās platformas un hibrīdās mobilās platformas salīdzinājums

Skaidrojumi: priekšrocība neitrāli trūkums

Faktors	Dzimtā platforma	Hibrīdā mobilā platforma
Izmaksas	Samērā augstas, atkarīgas no platformu skaita	Izmaksas būtiski samazina koda pārnesamības nodrošinājums
Pārnesamība	Kods strādā tikai attiecīgajā platformā	Nodrošināta koda pārnesamība
Saskarne ar iekārtas komponentēm	Nodrošina 100% saskarni ar iekārtas komponentēm	Nodrošina pamatiespējas ar iekārtas komponentēm
Lietotāja interfeiss	Nodrošina pamatinterfeisa elementus ar augstu lietojamību	Lietotāja interfeisa ietvari nodrošina līdzīgus interfeisa elementus kā dzimtajā platformā
Distribūcija	Nepieciešama publicēšana lietotņu veikalā ar noteiktiem ierobežojumiem	
Veiktspēja	Nodrošina tiešu pieeju operētājsistēmas funkcionalitātei	Ir atkarīga no izmantotajām tehnoloģijām un izstrādes metodēm
Satura atjaunināšana	Nepieciešamas atjaunot, izmantojot lietotņu veikalu	Nodrošina attālinātu atjaunināšanu, bez lietotnes iesūtīšanas lietotņu veikalā

Dzimtās platformas priekšrocības ir saskarne un veiktspēja, bet hibrīdās mobilās platformas priekšrocības ir zemās izmaksas un koda pārnesamība. Mobilo lietotņu izstrāde strauji attīstās un dzimto platformu ietvari tiek papildināti ar jaunām funkcionalitātēm, kuras pēc tam var integrēt hibrīdās mobilās lietotnes ietvaros, kas nozīmē, ka hibrīdā mobilā lietotne spēs izmantot dzimtās platformas jaunās funkcionalitātes ātrāk un efektīvāk.

Zemāk ir aplūkojams dzimtās platformas lietotnes un hibrīdās mobilās lietotnes salīdzinājums.

3.9.2. tabula

Dzimtās platformas lietotnes un hibrīdās mobilās lietotnes salīdzinājums [Korf2012]

	Dzimtās platformas lietotne	Hibrīdā mobilā lietotne
Ierīces saskarne		
Kamera	+	+
Paziņojumi (*notifications*)	+	+
Kontakti	+	+
Kalendārs	+	+
Bezastes glabātuve	Droša datņu glabātuve	Koplietojama *SQL Lite* datubāzes
Geolokācija	+	+
Žesti		
Pavilkšana (*Swipe*)	+	+
Savilkšana (*Pinch*)	+	+
Izplešana (*Spread*)	+	+
Pieskāriens (*Tap*)	+	+
Savienojums	Tiešsaistes un bezsaistes	

Pēc salīdzinājuma var secināt, ka abām tehnoloģijām ir savas priekšrocības un trūkumi, kuras ir jāvērtē konkrētos gadījumos. Var secināt arī to ka, hibrīdās mobilās lietotnes būtiski neatpaliek no dzimtās platformas lietotnēm iespēju ziņā un spēj konkurēt ar tām.

Zemāk ir uzskaitīti faktori, kuri palīdz izvēlēties izstrādes pieeju atkarībā no projekta specifikas.

Faktori, kas palīdz izvēlēties dzimtās platformas izstrādes pieeju:

- Lietotne tiks izstrādāta konkrētai platformai,
- Lietotnes prasības pārsniedz hibrīdo lietotņu iespējas,
- Lietotnes veiktspēja ir būtisks faktors,
- Grafiski sarežģītas lietotnes izstrāde.

Faktori, kas palīdz izvēlēties hibrīdās mobilās lietotnes izstrādes pieeju:

- Tiek nodrošināti izstrādātāji ar tīmekļa tehnoloģiju pieredzi,
- Lietotne ir jāizstrādā vairākas platformās,
- Lietotne ir vienkārša pēc grafiskās uzbūves,
- Lietotnes izstrādei ir ierobežoti resursi.

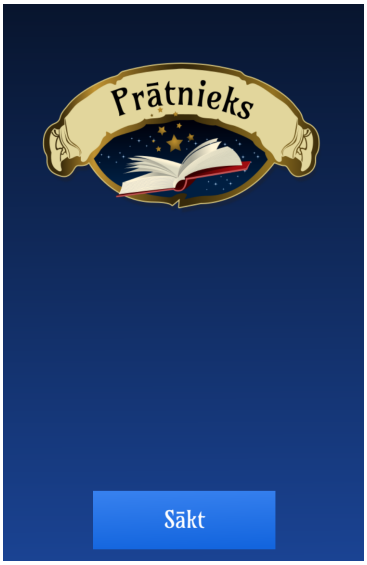
4. MOBILĀS LIETOTNES IZSTRĀDE

Lai detalizēti salīdzinātu abas izstrādes platformas – *HTML5* un *Adobe AIR* izstrādes platformas, tika izstrādāta neliela mobilā lietotne abās izstrādes platformās. Tas ļāva veiksmīgāk izpētīt abu platformu izstrādes iespējas, veikt veiktspējas testēšanu un biznesa aspekta salīdzinājumu, iegūt secinājumus par platformu priekšrocībām un trūkumiem. Mobilās lietotnes programmatūras fragmentus ir iespējams aplūkot 1. un 2. pielikumā. Mobilās lietotnes resursu licences ir iespējams aplūkot 3. pielikumā.

4.1. Mobilās lietotnes neformāls apraksts

Testēšanas nolūkos uz abām platformām tika izstrādāta divu dimensiju spēle "Prātnieks", kura ir veidota uz erudīcijas spēles "Kurš vēlas kļūt par miljonāru" bāzes. Spēles uzdevums ir atbildēt uz 15 jautājumiem. Katram jautājumam ir četri atbilžu variantiem. Par katru pareizu atbildi, lietotājs saņem "prātnieka naudiņas". Ja lietotājs atbild nepareizi, spēle ir beigusies un lietotājs saņem "nedegošā posma naudiņu" summu. Kopumā spēlē ir trīs "nedegošo naudiņu posmi" – pirmais jautājums, piektais jautājums un desmitais jautājums. Pavarot šos posmus, zaudējumu gadījumā tiek iegūta pēdējā "nedegošā posma" summa. Lietotājam ir iespējams spēles gaitā izmantot trīs papildiespējas:

- 50/50 – atbilžu skaits tiek samazināts uz pusi,
- Zvans mammai – lietotājam tiek piedāvāta provizoriski pareizā atbilde,
- Draugu palīdzība – lietotājam tiks uzrādīts procentuāls iespējamo atbilžu variantu sadalījums.



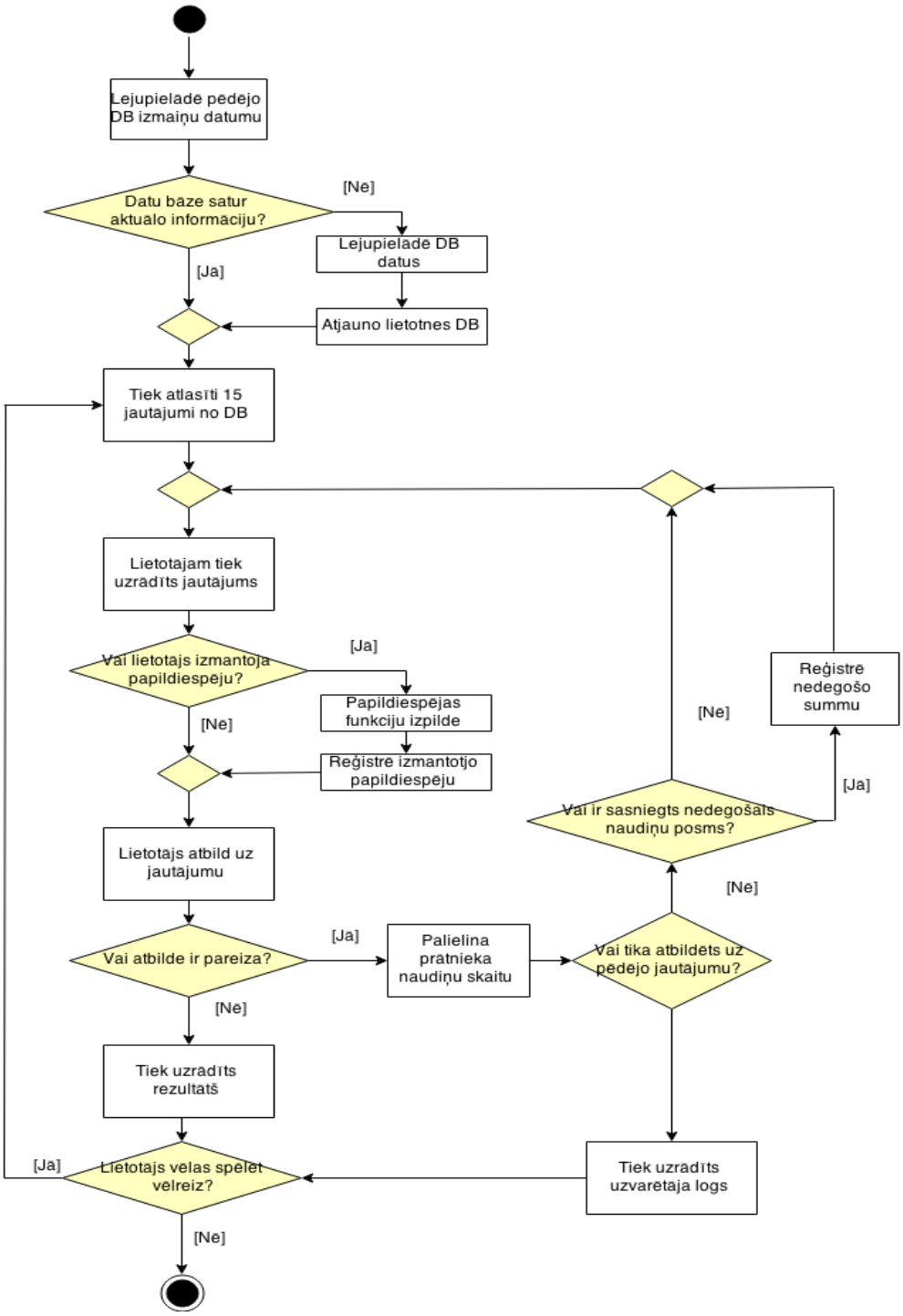
4.1.1. att. **Mobilās lietotnes sākuma skats**

4.1.2. att. **Lietotnes "Prātnieka" jautājuma skats**

Spēles mērķis ir iegūt pēc iespējas vairāk "prātnieka naudiņas".

4.2. Mobilās lietotnes aktivitāšu diagramma

Lai izprastu detalizētāku izstrādātās sistēmas operāciju algoritmisko un loģisko realizāciju, tika izveidota mobilās lietotnes aktivitāšu diagramma, kas attēlo sistēmas vadības plūsmas starp funkcijām. Zemāk ir aplūkojama mobilas lietotnes aktivitāšu diagramma.



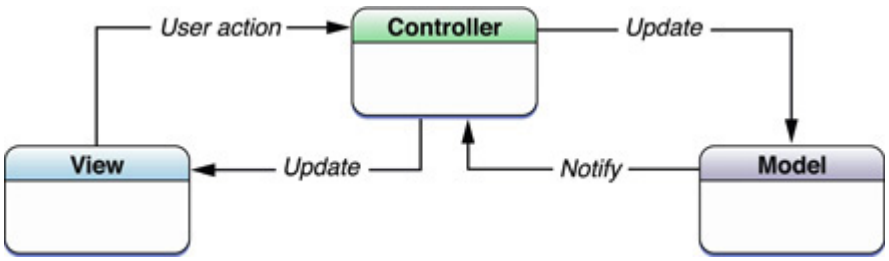
4.2.1. att. **Mobilās lietotnes "Prātnieks" aktivitāšu diagramma**

Aktivitāšu diagramma palīdzēja veikt operāciju algoritmisko realizāciju abās lietotnēs, kas uzlaboja tās kvalitāti un organizāciju, nodrošinot efektīvāku datu plūsmu.

4.3. Programmatūras arhitektūras izstrādes modelis

Mobilās lietotne tika izstrādāta pēc *MVC* arhitektūras modeļa. *MVC* (*Model–View–Controller*) ir lietotņu arhitektūras struktūra, kura norāda kā būtu nepieciešams nodalīt lietotnes darbības organizētā veidā. *Model* var definēt kā lietotnes „smadzenes" un tas tiek izmantots, lai veiktu dažādas apstrādes darbības kā datu lejupielādēšanu un parsēšanu. *View* ir lietotāja saskarne ar lietotni. Tajā tiek definēti visi skata elementi un *View* reģistrē lietotāja darbības, piemēram, nospiesta poga. *Controller* veic komunikāciju starp *View* un *Model*. Ja lietotājs nospiež pogu, *View* to reģistrē un nosūta informāciju *Controller*. Pēc tam, *Controller* to nosūta *Model*, kas veic datu apstrādi.

Kad apstrāde ir pabeigta, *Model* sūta informāciju *Controller*, kas pēc tam atjaunina *View* skata elementus. Lietotnei ir iespējami vairāki šo elementu pāri, jo *Controller* spēj komunicēt arī ar citiem *Controller*.



4.3.1. att. **MVC darbības principi** [Apple2013b]

4.4. Mobilo platformu ietvaru un rīku izvēle

Programmatūras izstrādē tika izmantoti konkrētās platformas mobilie ietvari, kas ļāva izstrādāt lietotni tieši viedtālruņiem. Tā kā izstrādātā programmatūra bija spēle, tad tika izvēlēti tieši spēļu izstrādes dziņi (*game engines*), kas būtiski atviegloja zema līmeņa komunikāciju ar aparatūru.

Spēļu dziņiem bija nepieciešams iekļaut šādus izvēles kritērijus:

- Bezmaksas atvērtā pirmkoda programmatūra,
- Nodrošina spēļu vairākplatformu izstrādi (konkrēti mobilo platformu),
- Ir plaši izmantota (popularitāte) [Adobe2015b, Clay2015],
- Nodrošina izstrādes dokumentāciju.

Tika izvēlēti šādi spēļu dziņi – ***Phaser*** un ***Apache Cordova*** ietvari [Phaser2015, Cordova2015] *HTML5* platformai un ***Starling*** ietvars [Gamua2015] *Adobe AIR* platformai.

Phaser ietvars izmanto *pixi.js* renderēšanas dzini, kas ir viens no ātrākajiem *JavaScript* renderēšanas dziņiem, kas izmanto *GPU*. Turklāt ietvars ir populārs un nodrošina daudzus paplašinājumus un detalizētu dokumentāciju. *Starling* ir *Adobe* sponsorēts ietvars, kas nodrošina atbalstu sistēmu izstrādātājiem, ļauj izstrādāt spēles uz dažādām platformām, ir populārs un nodrošina daudzus paplašinājumus, detalizētu dokumentāciju. Abi ietvari atbilst iepriekš definētajiem kritērijiem.

Programmatūra tika izstrādāta ***NetBeans IDE 8.0.2*** [NetBeans2015] *HTML5* platformas un ***FlashDevelop 4.7.2.*** [FlashDelevop2015] *Adobe AIR* platformas izstrādes vidēs.

4.4.1. *Phaser* ietvars

Programmatūra *HTML5* platformai tika izstrādāta, izmantojot *Phaser* ietvaru. *Phaser* ir atvērtā pirmkoda *HTML5* ietvars, kuru izstrādāja uzņēmums *Photon storm*. Tā mērķis ir izstrādāt spēles vairākplatformu sistēmām – gan darbvirsmu, gan mobilo iekārtu tīmekļa pārlūkiem, gan nodrošināt augstu veiktspēju mobilo tīmekļu pārlūkos. Ietvars spēļu renderēšanai izmanto *WebGL* tīmekļa tehnoloģiju, ja tādu iespēju nodrošina aparatūra, pretējā gadījumā, izmanto *Canvas* tīmekļa tehnoloģiju [Davey2013 [Bunyan2015]].

Phaser nodrošina lietotnes izstrādi šādām platformām [Phaser2015]:

- Darbvirsmas tīmekļa pārlūkiem – *IE 9* vai augstākai versijai, *Mozilla Firefox*, *Google Chrome* un *Safari*,

- Mobilo tīmekļu pārlūkiem – *iOS 5* vai augstākai versijai, *Android 2.2* vai augstākai versijai.

Phaser ietvara galvenās iezīmes [Davey2013]:

- Vienkārša lietotāja interfeisa datņu ielāde,
- Renderēšana: *WebGL* vai *Canvas* tīmekļa tehnoloģijas,
- Audio izvads: *Web Audio* vai *Legacy Audio* tīmekļa iespējas,
- Iespēja izmantot spraudņus.

4.4.2. *Apache Cordova* ietvars

Apache Cordova ir bezmaksas, atvērta pirmkoda dažādu iekārtu interfeisu kopums, kas ļauj mobilo lietotņu izstrādātājiem izmantot dzimto iekārtu funkcionalitāti kā kameru vai *GPS* tīmekļa vidē. Apvienojot iekārtu interfeisu kopu ar lietotāju interfeisu ietvariem kā *jQuery Mobile* vai *Dojo Mobile*, šis ietvars ļauj izstrādāt mobilās lietotnes *HTML*, *CSS*, *JavaScript* programmēšanas valodās dažādām platformām [Cordova2015].

Izmantojot *Apache Cordova* interfeisu, izstrādātājiem nav nepieciešamas zināšanas dzimto platformu programmēšanas valodās (*Java, Objective-C* u.c.). Mobilās lietotnes tiek izstrādātas, izmantojot tīmekļa tehnoloģijas, kuras pēc tam tiek izmantotas tīmekļa pārlūkā. Tīmekļa tehnoloģijām tiek nodrošināti standarti dažādās platformās, kas ļauj veikt lietotņu izstrādi vairākām platformām uz vienas programmatūras koda bāzes [Cordova2015].

Mobilās lietotnes, kuras tiek izstrādātas ar *Apache Cordov*a ietvaru un ir veidotas, izmantojot tīmekļa tehnoloģijas, tiek kompilētas kā dzimtās platformas lietotnes, kas nozīmē, ka lietotnes ir iespējams publicēt platformu lietotņu veikalos.

Apache Cordova nodrošina mobilo lietotņu izstrādi šādām platformām: *iOS*, *Android*, *Blackberry*, *Windows Phone*, *Palm WebOS*, *Bada* un *Symbian*. [Cordova2015].

4.4.3. *Starling* ietvars

Programmatūra priekš *Adobe AIR* platformas tika izstrādāta, izmantojot *Starling* ietvaru. *Starling* ir bezmaksas, atvērtā pirmkoda ietvars *ActionScript 3.0* programmēšanas valodā, kura izstrādi atbalsta uzņēmums *Adobe*. Tā mērķis ir izstrādāt spēles *Adobe Flash*/*Adobe AIR* platformai. Ietvars ir izstrādāts kā augšējais slānis *Stage3D* interfeisam, kas ļauj izstrādātājiem ērtāk un efektīvāk izmantot Stage3D iespējas. *Stage3D* interfeiss ir *Adobe Flash*/*AIR* paātrinātas aparatūras arhitektūra priekš 2D un 3D modelēšanas [Imbert2013].

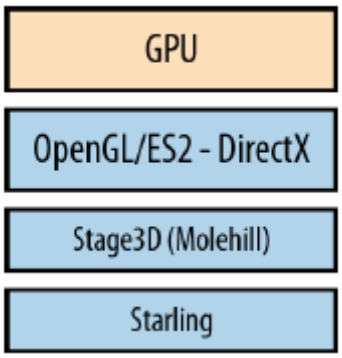
Starling nodrošina izstrādi šādām platformām:

- Mobilā platforma – *iOS* un *Android*,
- Tīmekļa pārlūki – *Mozilla Firefox*, *Google Chrome*, *Opera*, *IE*.

Starling ietvara galvenās iezīmes [Imbert2013,1-2]:

- Intuitīva izstrāde,
- Neliela izmēra – aizņem tikai 80KB,
- Bezmaksas un aktīvs ietvars,
- Nodrošina fizikas dziņus,
- Nodrošina ērtu un efektīvu piekļuvi aparatūrai 2D un 3D modeļu attēlošanai.

Zemāk ir aplūkojama grafiskās aparatūras piekļuves kārtu diagramma.



4.4.3.1. att. **Grafiskās aparatūras piekļuves kārtas** [Imbert2013]

Starling ietvars izmanto *Stage3D* interfeisus, kuri ir zema līmeņa grafiskās aparatūras interfeisi, kuri darbojas, izmantojot *OpenGL* un *DirectX* darbvirsmu ierīces un *OpenGL ES3* mobilās iekārtas, nodrošinot iespēju izstrādātājiem izmantot zema līmeņa interfeisus caur augsta līmeņa interfeisiem [Imbert2013].

5. *HTML5* UN *ADOBE AIR* PLATFORMU NOVĒRTĒŠANA

Lai veiktu izstrāžu pieeju salīdzinājumu, bija nepieciešams veikt individuālu izstrāžu pieeju novērtējumu. Katra izstrādes pieeja tika novērtēta pēc *Henning Heitkötter* vairākplatformu salīdzinājuma kritērijiem [Heitkötter2013], kas veic pieeju vērtēšanu no izstrādes un infrastruktūras perspektīvas. Pamatojoties uz individuālo izstrāžu pieeju novērtējumu, tika veikts abu pieeju salīdzinājums un iegūti secinājumi.

5.1. Novērtēšanas kritēriji

Izstrāžu pieeju novērtējumā tika izmantoti 13 novērtēšanas kritēriji, kuri sadalīti divās kategorijās – infrastruktūras kritēriji un izstrādes kritēriji. Zemāk ir iespējams aplūkot infrastruktūras kategorijas novērtējuma kritērijus.

5.1.1. tabula

Infrastruktūras kategorijas novērtējuma kritēriji [Heitkötter2013]

I1. Licences un izmaksas Kritērijs apraksta ietvara un pieejas izmaksas – vai izmantotie rīki ir bezmaksas, licences nosacījumus, izstrādājamo projektu publicēšanas izmaksas u.c. nosacījumus, kas saistīti ar licencēm un izmaksām.
I2. Atbalstītās platformas Apraksta atbalstītās platformas, pievēršot uzmanību vai pieeja nodrošina programmatūras izstrādi vienādi labi visām platformām.
I3. Piekļuve platformu specifiskajām funkcijām Apraksta iespējas piekļūt iekārtu aparatūrai, piemēram, kamerai vai GPS, un platformu funkcionalitātei kā kontaktiem un paziņojumiem.
I4. Ilgtermiņa iespējamība Šis kritērijs novērtē vai pieejas ietvari tiek regulāri atjaunoti, vai tiek veikti uzlabojumi, vai tiek atbalstītas jaunākas operētājsistēmas, vai tiek nodrošināta atbalsta sistēma (*support*).
I5. Lietojamība Lietojamība ir svarīgs aspekts, jo lietotājiem ir svarīga atgriezeniskā saite ar lietotni. Kritērijs apraksta pieejas iespēju nodrošināt veiksmīgu lietojamību, kas tuvināta dzimto platformu lietojamībai.

I6. Lietotnes ātrums Kritērijs neapraksta lietotnes veiktspējas mērījumus, bet gan lietotnes atsaucību uz lietotāja veiktajām darbībām. Uzmanība tiek vērsta uz lietotāja pieredzi (*user experience*).
I7. Distribūcija Novērtē pieejas mobilo lietotņu distribūcijas sarežģītību un alternatīvas lietotņu atjaunošanā.

Infrastruktūras kategorija apraksta kritērijus saistībā ar lietotnes dzīves ciklu, izmantošanu, operāciju un funkcionalitāšu iespējām. Zemāk ir iespējams aplūkot izstrādes kategorijas novērtējuma kritērijus.

5.1.2. tabula

Izstrādes kategorijas novērtējuma kritēriji [Heitkötter2013]

D1. Izstrādes vide Novērtē pieejas nodrošinātos izstrādes rīkus (izstrādes vidi, atkļūdošanas iespējas, simulāciju) un funkcionālās iespējas (automatizētos testus). Tiek vērtēta arī sistēmas uzstādīšanas sarežģītība.
D2. Vizuālais interfeiss Vērtē vizuālā interfeisa izstrādes sarežģītību, iespēju izstrādāt un testēt vizuālo interfeisu bez lietotnes emulācijas.
D3. Izstrādes sarežģītība Kritērijs vērtē dokumentācijas un mācību materiālu kopumu. Tiek vērtētā interfeisa un dokumentācijas kvalitāte. Piemēram, vai dokumentācija ir skaidri aprakstīta un programmatūras fragmentu piemēri ir viegli saprotami. Mācību materiālu novērtēšanā tiek vērtētas pieejas mācību iespējas, mācību literatūras pieejamība, ietvaru sarežģītība.
D4. Uzturamība Kritērijs vērtē lietotnes uzturamību pēc programmatūras izstrādes koda rindiņu skaita. Nodrošinot mazāku programmatūras koda rindiņu skaitu, tiek nodrošināta vieglāka lasāmība, nepieciešamas ieguldīt mazāk resursu programmatūras uzlabošanā un jaunu izstrādātāju apmācībā.
D5. Mērogojamība Kritērijs vērtē sistēmas programmatūras vai aparatūras spēju normāli funkcionēt gadījumā, ja tās funkcionēšanas apjoms vai konfigurācija tiek mainīta, piemēram, lietotāju skaita palielināšana nesamazina sniegto pakalpojumu apjomu, pāreja uz citu operētājsistēmu netraucē normālu sistēmas funkcionēšanu.

D6. Veiktspēja
Veiktspējas testēšana sastāv no divām daļām – algoritmiskā veiktspējas testa un grafiskā veiktspējas testa. Algoritmiskais tests sastāv no piecu miljonu nejauši sajauktu skaitļu kārtošanas, izmantojot *QuickSort* kārtošanas algoritmu. Grafiskais veiktspējas tests sastāv no *BunnyMark* [CreateJS2014] veiktspējas testa, kas vērtē maksimālo bitkaršu spraitu (*bitmap sprites*) skaita renderēšanu, nodrošinot 60 kadrus sekundē (*FPS*).

Izstrādes kategorijas kritēriji novērtē pieejas izstrādes procesus kā lietotnes izstrādes rīkus, testēšanu, atkļūdošanas rīkus, veiktspējas testus, izstrādes sarežģītību u.c. procesus.

5.2. *HTML5* platformas novērtējums

Programmatūra *HTML5* platformai tika izstrādāta, izmantojot *Phaser* un *Apache Cordova* ietvarus. *Phaser* un *Apache Cordova* ietvari ir bezmaksas, atvērtā pirmkoda ietvari *HTML*, *CSS*, *JavaScript* programmēšanas valodā. Novērtējums tika veikts uz ***iPhone 6 iOS 8.3*** un ***Samsung Galaxy S2 Android 4.4.4*** ierīcēm. Novērtējums balstās uz *Heitkötter Henning Apache Cordova* novērtējuma [Heitkötter2013, 11-12] un uz autora individuālā novērtējuma.

Zemāk ir iespējams aplūkot *HTML5* platformas un *Phaser*, *Apache Cordova* ietvaru infrastruktūras kategorijas novērtējumu.

5.2.1. tabula

***HTML5* platformas un *Phaser, Apache Cordova* ietvaru infrastruktūras kategorijas novērtējums**

I1. Licences un izmaksas
Abi ietvari ir bezmaksas, atvērtā pirmkoda ietvari, kuri ir izplatīti zem *Apache License 2.0* licences. Arī ietvara izstrādes vide *NetBeans* ir bezmaksas, atvērtā pirmkoda izstrādes vide. Komerciālu lietotņu izstrādi var veikt bezmaksas.
I2. Atbalstītās platformas
Apache Cordova nodrošina mobilo lietotņu izstrādi šādām platformām: *iOS*, *Android*, *Blackberry*, *Windows Phone*, *Palm WebOS*, *Bada* un *Symbian.* Lai izmantotu *Phaser* ietvaru, nepieciešams izmantot šādus darbvirsmas tīmekļa pārlūkus – *IE 9* vai augstāku versiju, *Mozilla Firefox*, *Google Chrome* un *Safari*. Ja ir nepieciešams izmantot mobilos pārlūkus, tad ir jānodrošina šādas operētājsistēmas versijas – *iOS 5* vai augstāka versija, *Android 2.2* vai augstāka versija.

I3. Piekļuve platformu specifiskajām funkcijām

Apache Cordova nodrošina vienkāršu pieeju galvenajām viedtālruņu aparatūras iespējām, piemēram, skārienjūtīgā ekrāna žestiem, iespēju izmantot kameru un mikrofonu, akselerometra datu saņemšanu vai GPS u.c. iespējas. *Apache Cordo*va piedāvā iespēju izstrādāt operētājsistēmu paplašinājumus, kas ļauj izmantot citas viedierīces iespējas, kuras *Apache Cordova* nenodrošina (piemēram, operētājsistēmu spēļu servisiem – *GameCenter*, *Google Play*). Pašlaik *Apache Cordova* datubāzē ir **pieeja 987 paplašinājumiem**.

I4. Ilgtermiņa iespējamība

Apache Cordova tika prezentēts 2011. gadā, kas nozīmē, ka *Apache Cordova* ir samērā jauns un ilgtspējīgs projekts. *Apache Cordova* ietvara atbalstu nodrošina uzņēmums *Adobe*, kas regulāri veic ietvara atjaunošanu un pielāgošanu jaunajām OS iespējām. *Apache Cordova* nodrošina plašu paplašinājumu datubāzi.

I5. Lietojamība

Apache Cordova ietvars nenodrošina dzimto platformu interfeisa elementus, bet tie tiek attēloti manuāli, kas nozīmē, ka katru elementu – pogu, teksta lauku nepieciešams izveidot un izkārtot pašam izstrādātājam. Lietotāja interfeisa izstrāde, izmantojot *CSS* ir sarežģīta un laikietilpīga. Vairums *JavaScript* lietotāju interfeisu ietvari kā *jQuery Mobile* nodrošina līdzīgu interfeisu elementu izmantošanu kā dzimtajās platformās, tomēr atšķirības ir viegli pamanāmas. *Apache Cordova* ievēro lietotnes dzīves cikla vadlīnijas – tiek ievēroti lietotnes statusa maiņas pieprasījumi, piemēram, lietotnes aizvēršana un atvēršana.

I6. Lietotnes ātrums

Apache Cordova iOS mobilā lietotne atveras ātri un lietotnes atgriezeniskās saites darbības ir plūstošas un ātras, tomēr lietotāja interfeisa animācijas nav tik plūstošas kā tas ir dzimto platformu lietotnēs, kas liek secināt, ka lietotne nav izstrādāta dzimto platformu vidē. *Apache Cordova* nodrošina pieeju grafiskajai kartei (*GPU*) tikai sākot no *iOS 8* versijas un *Android 5* versijas operētājsistēmām. Tas ir svarīgi, ja nepieciešams nodrošināt atbalstu vecākām iekārtām. Mobilā lietotne *Android* operētājsistēmas vidē bija lēna un grūti lietojama, jo *WebGL* tehnoloģija tiek atbalstīta sākot no *Android 5* versijas.

I7. Distribūcija
HTML5 platforma ļauj veikt lietotņu distribūciju uz jebkuru no atbalstītajām platformām. Lai veiktu distribūciju uz *Mac OS X* vai *iOS* lietotnēm, parasti nepieciešams to darīt *Mac OS X* vidē, jo gan *iOS SDK*, gan *Mac OS X SDK* ir pieejams tikai *Mac OS X* vidē. Lietotnes tiek atjaunotas caur lietotņu veikaliem (*AppStore*, *Google Play*) vai arī lietotnes datnes atrodas uz ārējā servera, kas nozīmē, ka, lai veiktu lietotnes satura atjaunošanu, nav nepieciešama lietotnes atjaunošana caur veikalu.

Novērtējot infrastruktūru, var secināt, ka *HTML5* platforma nodrošina bezmaksas, atvērtā pirmkoda ietvarus, kas ļauj publicēt komerciālas lietotnes bezmaksas. *Apache Cordova* nodrošina mobilo lietotņu izstrādi šādām platformām: *iOS*, *Android*, *Blackberry*, *Windows Phone*, *Palm WebOS*, *Bada* un *Symbian*. *Apache Cordova* nodrošina piekļuvi galvenajām ierīces aparatūras iespējām. Kopumā *Apache Cordova* datubāzē ir 987 paplašinājumi. Platforma ir dzīvot ilgtspējīga, jo regulāri tiek atjaunota un nodrošina izstrādātāju atbalstu. *HTML5* izstrādes metodes priekšrocība ir lietotņu atjaunošana, neizmantojot platformu lietotņu veikalus.

Zemāk ir iespējams aplūkot *HTML5* platformas un *Phaser*, *Apache Cordova* ietvaru izstrādātāju kategorijas novērtējumu.

5.2.2. tabula

***HTML5* platformas un *Phaser*, *Apache Cordova* ietvaru izstrādātāju kategorijas novērtējums**

D1. Izstrādes vide
Izstrādātājiem ir brīva izvēle izstrādes vides izvēlē, bet ne visas izstrādes vides nodrošina automātisku *Apache Cordova* funkciju aizpildīšanu. Ja nepieciešams izstrādāt mobilo lietotni, tad ir jāizmanto izstrādes vidi, kas atbalsta konkrētās platformas lietotņu izstrādi, piemēram, *iOS* gadījumā tā ir *Xcode* izstrādes vide. Alternatīva pieeja ir izmantot *Adobe* maksas servisu – *PhoneGap Build,* kas ļauj augšupielādēt programmatūras kodu un veikt lietotņu izveidi uz visām izvēlētajām platformām. Sīkāk par servisu skatīt **https://build.phonegap.com**. Programmatūras testēšana un rīki ir atkarīgi no izstrādes vides.

D2. Vizuālais interfeiss

Kā jebkuru tīmekļa lietotni, arī *Apache Cordova* lietotni ir iespējams izstrādāt ar dažādiem vizuāliem rīkiem, piemēram, *WYSIWYG* rīku vai tīmekļa pārlūku, definējot visu programmatūras kodā.

D3. Izstrādes sarežģītība

Apache Cordova vidē lietotnes tiek izveidotas standarta tīmekļa programmēšanas valodās – *HTML*, *CSS* un *JS*, kuras ir ļoti populāras. *Apache Cordova* nodrošina plašu un sīki detalizētu dokumentāciju ar programmatūras koda fragmentiem. Pašlaik *Apache Cordova* nodrošina datubāzi 987 paplašinājumiem, kas atvieglo pieeju iekārtu interfeisam. Izstrādātājam ir iespēja izveidot arī savus paplašinājums. To izstrāde nav sarežģīta un ir sīki aprakstīta *Apache Cordova* dokumentācijā, tomēr jauna projekta uzsākšana ir ilgs process. Katra izstrādes vide prasa citādāku projektu izveidi, konfigurāciju, kas aizņem daudz laika. *NetBeans IDE* nenodrošina *code-hinting*, kas palēnina programmatūras koda rakstīšanu. Būtisku problēmu autoram sagādāja mainīgo tipu neesamība, kas apgrūtināja testēšanu, veicot tipu pārbaudi un neuzrādot kļūdas kompilēšanas brīdī. *Apache Cordova* un to paplašinājumi tiek veikti no komandrindas, kas nozīmē, ka ir nepieciešamas papildus zināšanas komandrindu izmantošanā.

D4. Uzturamība

Mobilās lietotnes, kura izstrādātas *HTML5* vidē, galvenokārt ir programmētas *JavaScript* programmēšanas valodā. Vairums tīmekļa ietvaru nodrošina īsu un elegantu programmatūras kodu. Grūtības sagādā faila un objektu organizēšana, jo *JavaScript* nenodrošina klašu izstrādi – tās nepieciešamas izstrādāt mākslīgi, kas, savukārt, apgrūtina MVC arhitektūras modeļa ievērošanu. Ietvara trūkums ir tas, ka daudzi paplašinājumi ir atkarīgi viens no otra, kas palielina datņu apjomu un apgrūtina uzturamību. Programmatūra "Prātnieks" tika izstrādāta, izveidojot **26** datnes ar **1892** programmatūras koda rindiņām, papildus izmantotas arī **57** paplašinājuma datnes, neskaitot *Phaser* ietvaru, ar **15594** programmatūras koda rindiņām. "Prātnieka" kompilētā mobilā datne ir **2.5MB** liela.

D5. Mērogojamība

Apache Cordova nav platformas atkarīgs rīks, kas nozīmē, ka jebkuras platformas izmaiņas (piemēram, jauna operētājsistēmas versija) neveicina ietvara atjaunošanu. Tā kā *Apache Cordova* ir atvērtā pirmkoda ietvars, tad tas ir brīvi labojams kļūdu gadījumā, kuras rodas uz lielākām platformas izmaiņām. Ietvara trūkums ir atšķirīga vienas funkcijas funkcionalitāte dažādās operētājsistēmās, kas apgrūtina vairākplatformu lietotņu izstrādi, kā arī ietvara paplašinājumi ir savstarpēji atkarīgi.

D6. Veiktspēja

Kārtošanas algoritma vidējie rezultāti: pieci miljoni skaitļu tika sakārtoti 11,2 sekunžu laikā, izmantojot 400 KB lielu atmiņu.

***BunnyMark* veiktspējas vidējie rezultāti**:

60 FPS robeža tika noturēta līdz 2000 bitkaršu spraitiem (*sprite*), bet 30 FPS, kas ir akceptējošs rezultāts, tika noturēts līdz 4000 bitkaršu spraitiem. Tika secināts, ka FPS bija stabils līdz 2000 bitkaršu spraitiem, pēc kuriem FPS veiktspēja sāka strauji kristies līdz sasniedza 4 FPS robežu pie 30 000 bitkaršu spraitiem. Pie 4 FPS rezultāta lietotne nav akceptējama un lietojama.

Lai sasniegtu 100 000 bitkaršu spraitus, tika izmantoti 60MB iekārtas atmiņas, bet aizņēma tikai 2MB no iekārtas atmiņas.

Pēc izstrādes kategorijas novērtējama var secināt, ka *HTML5* piedāvā daudzus bezmaksas, atvērtā pirmkoda rīkus, kas nodrošina mobilo lietotņu izstrādi *HTML5* vidē. Tomēr, ja ir nepieciešams izstrādāt mobilo lietotni, tad jāizmanto izstrādes vidi, kas atbalsta konkrētās platformas lietotņu izstrādi, piemēram, *iOS* gadījumā tā ir *Xcode* izstrādes vide. Alternatīva pieeja ir izmantot *Adobe* maksas servisu – *PhoneGap Build. Apache Cordova* vidē, lietotnes tiek izveidotas standarta tīmekļa programmēšanas valodās – *HTML*, *CSS* un *JS*, kuras ir ļoti populāras. Pašlaik *Apache Cordova* nodrošina datubāzi ar 987 paplašinājumiem, kas atvieglo pieeju iekārtu interfeisam. *Apache Cordova* nav platformas atkarīgs rīks, kas nozīmē, ka jebkuras platformas izmaiņas (piemēram, jauna operētājsistēmas versija) neveicina ietvara atjaunošanu, bet paplašinājumi ir savstarpēji atkarīgi.

5.3. *Adobe AIR* platformas novērtējums

Programmatūra *Adobe AIR* platformai tika izstrādāta, izmantojot *Starling* ietvaru. *Starling* ir bezmaksas, atvērtā pirmkoda ietvars *ActionScript 3.0* programmēšanas valodā, kura izstrādi atbalsta uzņēmums *Adobe*. Novērtējums tika veikts uz ***iPhone 6 iOS 8.3*** un ***Samsung Galaxy S2 Android 4.4.4*** ierīcēm.

Zemāk ir iespējams aplūkot *Adobe AIR* un *Starling* ietvara infrastruktūras kategorijas novērtējumu.

5.3.1. tabula

***Adobe AIR* un *Starling* ietvara infrastruktūras kategorijas novērtējums**

I1. Licences un izmaksas *Starling* ietvars ir bezmaksas, atvērtā pirmkoda ietvars, kas ir izplatīts zem *Simplified BSD* licences. Arī ietvara izstrādes vide *FlashDevelop* ir bezmaksas, atvērtā pirmkoda izstrādes vide. Komerciālu lietotņu izstrādi var veikt bezmaksas.
I2. Atbalstītās platformas *Starling* ietvars nodrošina atbalstu divām mobilo lietotņu operētājsistēmām (*iOS, Android*), jebkuram tīmekļu pārlūkam, kas atbalsta *Adobe Flash player* un divām darbvirsmas operētājsistēmām (*Mac OS X, Windows*). Lai izmantotu *Starling* ietvaru, nepieciešams izmantot vismaz *Adobe AIR* 3.4. versiju vai *Flash Player* 11.4 versiju.
I3. Piekļuve platformu specifiskajām funkcijām *Adobe AIR* nodrošina vienkāršu pieeju galvenajām viedtālruņu aparatūras iespējām, piemēram, skārienjūtīgā ekrāna žestiem, iespēju izmantot kameru un mikrofonu, akselerometra datu saņemšanu vai GPS u.c. iespējas. *Adobe AIR* piedāvā iespēju izstrādāt operētājsistēmu paplašinājumus, kas ļauj izmantot citas viedierīces iespējas, kuras *Adobe AIR* nenodrošina (piemēram, operētājsistēmu spēļu servisiem – *GameCenter*, *Google Play*). Vairums no funkcijām tiek nodrošinātas caur standarta izstrādātājrīkkopu.
I4. Ilgtermiņa iespējamība *Adobe AIR* tika prezentēts 2008. gada. 25. februārī un *Starling* ietvars tika prezentēts 2011.gada 21. septembrī, kas nozīmē, ka abi projekti ir samērā jauni un ilgtspējīgi. *Starling* ietvara atbalstu nodrošina uzņēmums *Adobe*, kas regulāri veic ietvara atjaunošanu un pielāgošanu jaunajām OS iespējām un *Adobe AIR* versijām, kā arī nodrošina 24/7 klientu atbalsta servisu.

I5. Lietojamība *Starling* ietvars nenodrošina dzimto platformu interfeisa elementus, bet tiek attēloti manuāli, kas nozīmē, ka katru elementu – pogu, teksta lauku nepieciešams izveidot un izkārtot pašam izstrādātājam. Veicot grafisko elementu pielāgošanu dažādiem ekrāna izmēriem, var secināt, ka elementi izskatās dažādi un atrodas dažādās vietās, kas ļauj redzēt atšķirības no dzimto platformu lietotnēm, kaut arī animācijas un atgriezeniskā lietotnes saite ir skaidra un saprotama. Lietotnes dzīves cikls *Adobe AIR* platformā tiek ievērots pēc lietotnes dzīvescikla vadlīnijām – tiek ievēroti lietotnes statusa maiņas pieprasījumi, piemēram, lietotnes aizvēršana un atvēršana.
I6. Lietotnes ātrums Atverot lietotni, kas izstrādāta *Adobe AIR* vidē, tā tiek atvērta ātri, datu ielāde un interfeisu elementu attēlošana notiek ātri. Diemžēl animācijas brīžiem ir pārāk ātras, pārslēdzot tās starp skatiem, bet brīžiem pārāk lēnas pie skatu bīdīšanas. Tā kā interfeisa elementi tiek manuāli izveidoti, tad ir iespējams saskatīt atšķirības no dzimto platformu ierastajiem interfeisa elementiem. Kopumā lietotne ir ātra un lietotnes atbildes reakcija arī ir ātra. *Adobe AIR* atbalsta pieeju grafiskajai kartei (*GPU*).
I7. Distribūcija *Adobe AIR* platforma ļauj veikt lietotņu distribūciju uz jebkuru no atbalstītajām platformām. Lai veiktu distribūciju uz *Mac OS X* vai *iOS* lietotnēm, parasti nepieciešams to darīt *Mac OS X* vidē, jo gan *iOS SDK*, gan *Mac OS X SDK* ir pieejams tikai *Mac OS X* vidē. *Adobe AIR* versijās tiek iebūvēti jaunākie *iOS* un *Mac OS X SDK*, kas ļauj izstrādāt šīs operētājsistēmas lietotnes arī no citām operētājsistēmām kā *Windows*, kas ir viena no galvenajām *Adobe AIR* priekšrocībām. Lietotnes tiek atjaunotas caur lietotņu veikaliem (*AppStore*, *Google Play*).

Pēc infrastruktūras novērtējuma, var secināt, ka *Adobe AIR* platforma nodrošina bezmaksas, atvērtā pirmkoda ietvarus, kas ļauj publicēt komerciālās lietotnes bezmaksas. *Adobe AIR* atbalsta vadošās darbvirsmas un mobilo lietotņu operētājsistēmas – *Mac OS X*, *Windows*, *iOS*, *Android*, kā arī tīmekļa pārlūkus, kuri atbalsta *Flash* paplašinājumu. *Adobe AIR* nodrošina piekļuvi galvenajām ierīces aparatūras iespējām un ļauj izveidot atsevišķus paplašinājumus funkcijām, kas netiek nodrošinātas. Platforma ir dzīvot ilgtspējīga, jo regulāri tiek atjaunota un nodrošina izstrādātāju atbalstu. Tomēr lietojamība atšķiras no dzimto platformu lietotnēm, kas var negatīvi ietekmēt lietotnes kvalitāti. *Adobe AIR* nodrošina plašu lietotņu distribūciju arī *Mac OS X* un *iOS* operētājsistēmas lietotnes *Windows* vidē.

Zemāk ir iespējams aplūkot *Adobe AIR* un *Starling* ietvara izstrādātāju kategorijas novērtējumu.

5.3.2. tabula

***Adobe AIR* un *Starling* ietvara izstrādātāju kategorijas novērtējums**

D1. Izstrādes vide Programmatūra izstrāde uz *Adobe AIR* platformas tika veikta *FlashDevelop* izstrādes vidē, kas ir bezmaksas, atvērtā pirmkoda izstrādes vide. Izstrādes vide nodrošina vairāku platformu lietotņu kompilēšanu uz attiecīgas platformas iekārtas vai simulatorā. Tiek nodrošināti divu veidu simulatori – iebūvētais vai ārējais. Iebūvētais simulators nodrošina *Adobe AIR* vides simulāciju ar mainīgu ekrānu izmēru, kas pielāgojas ierīces ekrāna izmēram. Iebūvētais simulators nenodrošina platformas specifiskās funkcijas kā *GameCenter* (*iOS* spēļu serviss). Ārējie simulatori ir mobilo platformu izstrādes simulatori, kurus var piesaistīt *FlashDevelop* izstrādes videi (*iOS* simulators un *Android* emulators). Iekārtas un simulatori atkļūdošanas režīmā ir sasaistīti ar izstrādes vidi un saņem atkļūdošanas paziņojumus, piemēram, pārtraukumpunktus. Izstrādes vide nodrošina lietotņu kompilāciju gan caur komandrindu, gan no izstrādes vides. Ir iespēja izstrādāt automatizētus testus.
D2. Vizuālais interfeiss Vizuālo interfeisa izstrādi ir iespējams veikt divos veidos – izmantojot *Adobe* oficiālo izstrādes vidi *Adobe Flash Professional*, kas ļauj izmantot atsevišķu vizuālo interfeisu izstrādes vidi, vai vizuālo interfeisa izstrādi no programmatūras koda. Vizuālā interfeisa izstrādes vide ļauj izkārtot, definēt interfeisa elementus, piešķirt elementiem izpildāmo programmatūras kodu, definēt animācijas, mainīt interfeisa elementa slāņus u.c. ar vizuālo interfeisu saistītas lietas. Tomēr, ja izstrādātājs izmanto *FlashDevelop* izstrādes vidi, tad vizuālo interfeisu ir nepieciešams izstrādāt caur programmatūras kodu un testēšanai jāizmanto iekārta vai simulators.
D3. Izstrādes sarežģītība *Adobe AIR* programmēšanas valoda – *ActionScript 3* ir standarta objektorientēta programmēšanas valoda, kas ir līdzīga programmēšanas valodai *Java* vai *JavaScript*. *Adobe* nodrošina plašu un sīki detalizētu dokumentāciju ar programmatūras koda fragmentiem. *Adobe* atbalsta serviss ir ļoti plašs un vairumā problēmu atbildes ir atrodamas atbalsta lapā vai interneta forumos. Tomēr *Adobe SDK* nodrošina sarežģītu paplašinājumu izstrādi specifiskām platformām kā *Android* vai *iOS*. Vairums iespējas ir maksas pakalpojumi, piemēram, izstrādes vides.

D4. Uzturamība Tā kā *Adobe AIR* izmanto objektorientētu programmēšanas valodu, tad projekti tiek veidoti pēc objektorientētas pieejas vadlīnijām – klašu un objektu izveide, mantošana u.c. iespējas. Kods ir strukturēts pēc klasēm un objektiem. Objektiem ir definētas metodes un parametri. Izmantojot *Starling* ietvaru, programmatūra "Prātnieks" tika izstrādāta, izmantojot **39** datnes ar **2800** programmatūras koda rindiņām. "Prātnieka" kompilētā mobilā datne ir **10MB** liela.
D5. Mērogojamība *Adobe AIR* izstrādātāja rīkkopa (SDK) nav atvērtā pirmkoda rīkkopa, kas nozīmē, ka pie jebkurām platformas izmaiņām (piemēram, jaunu operētājsistēmas versiju), nepieciešams gaidīt uz jaunas izstrādātāja rīkkopas izveidi, kuru izstrādā pati *Adobe*. Tomēr, *Adobe AIR* izstrādātāja rīkkopa nodrošina atbalstu vairākām operētājsistēmas versijām, kas pielāgojas konkrētās operētājsistēmas versijas īpatnībām. Slikta iezīme ir, ka vienas funkcijas funkcionalitāte dažādās operētājsistēmas strādā atšķirīgi, kas apgrūtina vairākplatformu lietotņu izstrādi.
D6. Veiktspēja **Kārtošanas algoritma vidējie rezultāti**: pieci miljoni skaitļu tika sakārtoti 22,3 sekunžu laikā, izmantojot 40 KB lielu atmiņu. ***BunnyMark* veiktspējas vidējie rezultāti**: 60 FPS robeža tika sasniegta ar 3000 bitkaršu spraitiem, bet 30 FPS, kas ir akceptējošs rezultāts, tika sasniegts pie 15 500 bitkaršu spraitiem. Tika secināts, ka FPS bija stabils līdz 3000 bitkaršu spraitiem, pēc kuriem tas sāka strauji kristies līdz sasniedza 4 FPS robeža pie 150 000 bitkaršu spraitiem. Pie 4 FPS rezultāta lietotne nav akceptējama un lietojama. Sasniedzot 100 000 bitkaršu spraitus, tika izmantoti 7GB iekārtas atmiņas, bet aizņēma tikai 10MB no iekārtas atmiņas.

Pēc izstrādes kategorijas novērtēšanas var secināt, ka *Adobe AIR* piedāvā izcilas bezmaksas, atvērtā pirmkoda izstrādes vides, ar kurām ir iespējams veikt visas nepieciešamās darbības vairākplatformu mobilo lietotņu izstrādē. *Adobe AIR* platformas programmēšanas valoda ir *OOP ActionScrip3*, kas ir līdzīga programmēšanas valodām kā *Java* vai *JavaScript*. *Adobe* nodrošina detalizētu dokumentāciju ar programmatūras kodu fragmentiem, kas ļauj efektīvāk apgūt *Adobe AIR* platformu. Tomēr *Adobe AIR* izstrādātāja rīkkopa (SDK) nav atvērtā pirmkoda rīkkopa, kas nozīmē, ka pie jebkurām platformas izmaiņām (piemēram, jaunu operētājsistēmas versiju), nepieciešams gaidīt līdz jaunas izstrādātāja rīkkopas izveidei.

REZULTĀTI

Lai detalizēti salīdzinātu abas izstrādes platformas – *HTML5* un *Adobe AIR* izstrādes platformas, tika izstrādāta neliela mobilā lietotne abās izstrādes platformās. Tas ļāva veiksmīgāk izpētīt abu platformu izstrādes iespējas, veikt veiktspējas testēšanu un biznesa aspekta salīdzinājumu, kā rezultātā, tika iegūti secinājumi par platformu priekšrocībām un trūkumiem.

Katra izstrādes pieeja tika novērtēta pēc *Henning Heitkötter* vairākplatformu salīdzinājuma kritērijiem [Heitkötter2013,5-7], kas veic pieeju vērtēšanu no izstrādes un infrastruktūras perspektīvas. Izstrāžu pieeju novērtējumā tika izmantoti 13 novērtēšanas kritēriji, kuri sadalīti divās kategorijās – infrastruktūras kritēriji un izstrādes kritēriji.

Pēc individuālā novērtējuma, rezultāti tika apkopoti un veikts abu pieeju novērtējumu salīdzinājums. Zemāk ir aplūkojams *HTML5* un *Adobe AIR* infrastruktūras kategorijas novērtējumu salīdzinājums.

1. tabula

***Adobe AIR* un *HTML5* infrastruktūras kategorijas novērtējumu salīdzinājums**

I1. Licences un izmaksas *Starling* ietvars ir bezmaksas, atvērtā pirmkoda ietvars, kas ir izplatīts zem *Simplified BSD* licences. *Apache Cordova* ir atvērtā pirmkoda ietvari, kuri ir izplatīti zem *Apache License 2.0* licences. Abu izstrāžu pieeju ietvari un rīki ir bezmaksas, kā arī komerciālu lietotņu izstrāde arī ir bezmaksas.
I2. Atbalstītās platformas *Starling* ietvars nodrošina atbalstu divām mobilo operētājsistēmām (*iOS, Android*), kamēr *Apache Cordova* nodrošina atbalstu sekojošajām mobilajām operētājsistēmām: *iOS*, *Android*, *Blackberry*, *Windows Phone*, *Palm WebOS*, *Bada* un *Symbian. HTML5* izstrādes pieejai ir par piecām mobilajām operētājsistēmām vairāk nekā *Adobe AIR* izstrādes pieejai, tomēr *Adobe AIR* nodrošina atbalstu divām populārākajām mobilajām operētājsistēmām.
I3. Piekļuve platformu specifiskajām funkcijām Abas izstrādes pieejas nodrošina piekļuvi galvenajām iekārtas interfeisa funkcijām, piedāvājot iespēju izstrādāt paplašinājumus. Adobe AIR vairums no funkcijām tiek nodrošinātas caur standarta izstrādātājrīkkopu. Kamēr vairums *Adobe AIR* paplašinājumu ir maksas, tikmēr *Apache Cordova* datubāze nodrošina **987** bezmaksas paplašinājumus.

I4. Ilgtermiņa iespējamība

Adobe AIR tika prezentēts 2008. gadā, kamēr *Apache Cordova* tika prezentēts 2011. gadā, kas nozīmē, ka abi projekti ir samērā jauni un ilgtspējīgi. *Adobe AIR* un *Apache Cordova* atbalstu nodrošina uzņēmums *Adobe*, kas regulāri veic ietvaru atjaunošanu un pielāgošanu jaunajām OS iespējām un *Adobe AIR* versijām, kā arī nodrošina 24/7 klientu atbalsta servisu. Abi projekti ir ilgtspējīgi ar regulāriem atjauninājumiem un kļūdu labojumiem.

I5. Lietojamība

Adobe AIR un *HTML5* ietvari nenodrošina dzimto platformu interfeisa elementus, bet tie tiek attēloti manuāli, kas nozīmē, ka katru elementu – pogu, teksta lauku nepieciešams izveidot un izkārtot pašam izstrādātājam. Veicot grafisko elementu pielāgošanu dažādiem ekrāna izmēriem, var secināt, ka elementi izskatās dažādi un atrodas dažādās vietās, kas ļauj redzēt atšķirību no dzimto platformu lietotnēm. Abu pieeju mobilās lietotnes dzīves cikli tiek ievēroti pēc lietotnes dzīvescikla vadlīnijām – tiek ievēroti lietotnes statusa maiņas pieprasījumi, piemēram, lietotnes aizvēršana un atvēršana. Lietojamība ir atkarīga no izstrādātāja programmēšanas prasmēm, kaut gan *Adobe AIR* vizuālā veiktspēja ir labāka nekā *HTML5* izstrādes pieejai.

I6. Lietotnes ātrums

Abu izstrāžu pieeju *iOS* mobilās lietotnes bija ātras un atgriezenisko saišu atbildes bija pietiekami ātras. Tomēr lietotāja interfeisa animācijas nav plūstošas kā tas ir dzimto platformu lietotnēs, kas liek secināt, ka lietotne nav izstrādāta dzimto platformu vidē. Kopumā var uzskatīt, ka lietotnes strādā ātri, kaut gan atšķirība no dzimto platformu lietotnēm ir redzama. Starp abām lietotnēm netika saskatītas būtiskas atšķirības. *Adobe AIR* atbalsta pieeju grafiskajai kartei (*GPU*) vecāku mobilo operētājsistēmu versijām, bet *Apache Cordova* to nodrošina tikai sākot no *iOS 8* un *Android 5* operētājsistēmas versijām. Tas ir svarīgi, ja nepieciešams nodrošināt atbalstu vecākām iekārtām. *Adobe AIR* platformā *Android* mobilā lietotne strādāja tāpat kā *iOS* lietotne, savukārt *HTML5* platformā *Android* mobilā lietotne bija lēna un grūti lietojama.

I7. Distribūcija

Adobe AIR un *HTML5* platforma ļauj veikt lietotņu distribūciju uz jebkuru no atbalstītajām platformām. Lai veiktu distribūciju *Mac OS X* vai *iOS* lietotnēm, parasti nepieciešams to darīt *Mac OS X* vidē, jo gan *iOS SDK*, gan *Mac OS X SDK* ir pieejams tikai *Mac OS X* vidē. *Adobe AIR* versijās tiek iebūvēti jaunākie *iOS* un *Mac OS X SDK*, kas ļauj izstrādāt šo operētājsistēmu lietotnes arī no citām operētājsistēmām kā *Windows*, kas ir viena no galvenajām *Adobe AIR* priekšrocībām, kamēr *Apache Cordova* gadījumā ir jānodrošina

dators, kas atbalsta *Mac OS X* operētājsistēmu. *Adobe AIR* lietotnes tiek atjaunotas caur lietotņu veikaliem (*AppStore*, *Google Play*), kamēr *HTML5* gadījumā lietotnes saturu iespējams atjaunot attālināti, neveicot pašas lietotnes atjauninājumu caur lietotņu veikaliem.

Pēc infrastruktūras novērtējumu salīdzinājuma, autors secina, ka abām izstrādes pieejām ir vairāk kopīgu īpašību nekā atšķirīgu. Abas pieejas nodrošina bezmaksas, atvērtā pirmkoda rīkus un ietvarus, kas ļauj izstrādāt komerciālas lietotnes bezmaksas. Komunikācijā ar iekārtu interfeisu, abām pieejām ir nepieciešams izmantot paplašinājumus, kurus iespējams lejupielādēt no datubāzes vai izstrādāt pašam. Vairums *Adobe AIR* izstrādes pieejas paplašinājumu ir maksas paplašinājumi, kamēr *HTML5* paplašinājumi ir bezmaksas. Abi pieeju rīki un tehnoloģijas tiek regulāri atjaunoti, kas nozīmē, ka abas pieejas ir ilgtspējīgas. *HTML5* platforma nodrošina atbalstu lielākam skaitam operētājsistēmu (kopumā septiņām mobilo lietotņu operētājsistēmām), kamēr *Adobe AIR* platforma nodrošina atbalstu divām populārākajām mobilo lietotņu operētājsistēmām (*iOS*, *Android*).

Abas pieejas nodrošina lietotņu distribūciju. Lai veiktu distribūciju uz *Mac OS X* vai *iOS* lietotnēm, nepieciešams to darīt *Mac OS X* vidē, jo gan *iOS SDK*, gan *Mac OS X SDK* ir pieejams tikai *Mac OS X* vidē. *Adobe AIR* versijās tiek iebūvēti jaunākie *iOS* un *Mac OS X SDK*, kas ļauj izstrādāt šo operētājsistēmu lietotnes arī no citām operētājsistēmām kā *Windows*, kas ir viena no galvenajām *Adobe AIR* priekšrocībām, kamēr *Apache Cordova* gadījumā ir jānodrošina dators, kas atbalsta *Mac OS X* operētājsistēmu. *Adobe AIR* lietotnes tiek atjaunotas caur lietotņu veikaliem (*AppStore*, *Google Play*), kamēr *HTML5* gadījumā lietotnes saturu iespējams atjaunot attālināti, neveicot pašas lietotnes atjauninājumu caur lietotņu veikaliem.

Zemāk ir aplūkojams *HTML5* un *Adobe AIR* izstrādātāju kategorijas novērtējuma salīdzinājums.

2. tabula

***Adobe AIR* un *HTML5* izstrādātāju kategorijas novērtējuma salīdzinājums**

D1. Izstrādes vide
Programmatūras izstrāde uz *Adobe AIR* platformas tika veikta *FlashDevelop* izstrādes vidē, kas ir bezmaksas, atvērtā pirmkoda izstrādes vide. Izstrādes vide nodrošina vairāku platformu lietotņu kompilēšanu uz attiecīgas platformas iekārtas vai simulatorā. *Apache Cordova* gadījumā, ja nepieciešams izstrādāt mobilo lietotni, jāizmanto izstrādes vidi, kas atbalsta konkrētās platformas lietotņu izstrādi, piemēram, *iOS* gadījumā tā ir *Xcode* izstrādes vide. Alternatīva pieeja ir izmantot *Adobe* maksas servisu – *PhoneGap Build. Adobe AIR* platforma nodrošina vienu plašu izstrādes vidi, kamēr *Apache Cordova* platforma ir atkarīga no vairākiem platformu izstrādes rīkiem.

D2. Vizuālais interfeiss

Abas izstrādes pieejas nodrošina vizuālo interfeisu izstrādi, izmantojot vizuālo interfeisu izstrādes rīkus. *Adobe AIR* gadījumā, šāds rīks ir pieejams tikai maksas izstrādes vidē kā *Adobe Flash Professional*, kamēr *HTML5* nodrošina arī bezmaksas šādus rīkus. Alternatīva iespēja abām izstrādes pieejām ir vizuālo interfeisu izstrādes nodrošinājums caur programmatūras kodu.

D3. Izstrādes sarežģītība

Adobe AIR programmēšanas valoda – *ActionScript 3* ir standarta objektorientēta programmēšanas valoda, kas ir līdzīga programmēšanas valodai *Java* vai *JavaScript. Apache Cordova* vidē lietotnes tiek izveidotas standarta tīmekļa programmēšanas valodās – *HTML*, *CSS* un *JS*. Abas pieejas nodrošina plašas un detalizētas dokumentācijas, kuras nodrošina arī programmatūras koda piemērus. *Adobe* atbalsta serviss ir ļoti plašs un vairumā problēmu atbildes ir atrodamas atbalsta lapā vai interneta forumos. Tomēr *Adobe SDK* nodrošina sarežģītu paplašinājumu izstrādi specifiskām platformām kā *Android* vai *iOS*. Vairums iespēju ir maksas pakalpojumi, piemēram, izstrādes vides. Izmantojot *Apache Cordova* ietvaru, jauna projekta uzsākšana ir ilgs process. Katra izstrādes vide prasa citādāku projektu izveidi, konfigurāciju, kas aizņem daudz laika. *NetBeans IDE* nenodrošina *code-hinting*, kas palēnina programmatūras koda rakstīšanu. Būtisku problēmu autoram sagādāja mainīgo tipu neesamība, kas apgrūtināja testēšanu, veicot tipu pārbaudi un neuzrādot kļūdas kompilēšanas brīdī. Autors secināja, ka izstrādes sarežģītība *Adobe AIR* platformai ir zemāka nekā *HTML5* platformai, tomēr izstrādes sarežģītība ir atkarīga no ietvaru un izstrādes vides izvēles.

D4. Uzturamība

Tā kā *Adobe AIR* izmanto objektorientētu programmēšanas valodu, tad projekti tiek veidoti pēc objektorientētas pieejas vadlīnijām – klašu un objektu izveide, mantošana u.c. iespējas. Kods ir strukturēts pēc klasēm un objektiem. Objektiem ir definētas metodes un parametri.

Izmantojot *HTML5* izstrādes pieeju, grūtības sagādā faila un objektu organizēšana, jo *JavaScript* nenodrošina klašu izstrādi – tās nepieciešamas mākslīgi izstrādāt, kas apgrūtina *MVC* arhitektūras modeļa ievērošanu. *Apache Cordova* ietvara trūkums ir daudzu paplašinājumu atkarība, kas palielina datņu apjomu un apgrūtina uzturamību.

Uz *Adobe AIR* platformas lietotnei "Prātnieks" tika izmantotas **39** datnes ar **2800** programmatūras koda rindiņām, bet kompilētā mobilā datne ir **10MB** liela, savukārt, uz *HTML5* platformas lietotnei tika izmantotas **26** datnes ar **1892** programmatūras koda

rindiņām un kompilētā mobilā datne ir **2.5MB** liela. *Adobe AIR* lietotne aizņem četras reizes vairāk atmiņas, jo mobilā lietotne satur *Adobe AIR* izstrādātāja rīkkopu.

D5. Mērogojamība

Adobe AIR izstrādātāja rīkkopa (SDK) nav atvērtā pirmkoda rīkkopa, kas nozīmē, ka uz jebkurām platformas izmaiņām (piemēram, jaunu operētājsistēmas versiju), nepieciešams gaidīt līdz jaunas izstrādātāja rīkkopas izveidei, kuru izstrādā pati *Adobe*. Tomēr, *Adobe AIR* izstrādātāja rīkkopa nodrošina atbalstu vairākām operētājsistēmas versijām, kas pielāgojas konkrētās operētājsistēmas versijas īpatnībām. *Apache Cordova* nav platformas atkarīgs rīks, kas nozīmē, ka jebkuras platformas izmaiņas (piemēram, jauna operētājsistēmas versija) neveicina ietvara atjaunošanu. Tā kā *Apache Cordova* ir atvērtā pirmkoda ietvars, tad kļūdas ir brīvi labojamas arī pie lielākām platformas izmaiņām. Tomēr *Apache Cordova* paplašinājumu darbība ir savstarpēji atkarīga, kas nozīmē, ka kļūda vienā paplašinājumā var veicināt kļūdas citā paplašinājumā.

D6. Veiktspēja

Kārtošanas algoritma rezultāti: pieci miljoni skaitļu *Adobe AIR* platformā tika sakārtoti 22,3 sekunžu laikā, kamēr *HTML5* platforma skaitļus sakārtoja 11,2 sekunžu laikā. Abas platformas izmantoja *QuickSort* kārtošanas algoritmu, izstrādājot programmatūras kodu katrai platformai individuāli.

***BunnyMark* veiktspējas vidējie rezultāti**: *Adobe AIR* platforma 60 FPS robežu sasniedza ar 3000 bitkaršu spraitiem, bet 30 FPS, kas ir akceptējošs rezultāts, sasniedza pie 15 500 bitkaršu spraitiem. Tika secināts, ka FPS bija stabils līdz 3000 bitkaršu spraitiem, pēc kuriem tas sāka strauji kristies līdz sasniedza 4 FPS robežu pie 150 000 bitkaršu spraitiem un pārstāja kristies.

HTML5 platforma 60 FPS robežu noturēja līdz 2000 bitkaršu spraitiem, bet 30 FPS, kas ir akceptējošs rezultāts, tika noturēts līdz 4000 bitkaršu spraitiem. Tika secināts, ka FPS bija stabils līdz 2000 bitkaršu spraitiem, pēc kuriem tas sāka strauji kristies līdz sasniedza 4 FPS robežu pie 30 000 bitkaršu spraitiem un pārstāja kristies.

Adobe AIR platforma, sasniedzot 100 000 bitkaršu spraitus, izmantoja 7GB iekārtas atmiņas un aizņēma 10MB no iekārtas atmiņas, savukārt, *HTML5* platforma izmantoja 60MB iekārtas atmiņas un aizņēma 2MB no iekārtas atmiņas. Autors secina, ka *HTML5* algoritmiskās darbības veic ātrāk nekā *Adobe AIR* platforma, savukārt, *Adobe AIR* veic efektīvāku vizuālo interfeisu renderēšanu nekā *HTML5* platforma.

Pēc izstrādātāju kategorijas novērtējuma salīdzinājuma, autors secina, ka abām izstrādes pieejām tiek nodrošināti bezmaksas izstrādes rīki. *FlashDevelop* izstrādes vide nodrošina kompilāciju daudzām platformām, kamēr *HTML5* pieejas izstrādes vide nodrošina konkrētas platformas kompilāciju. Alternatīva ir mākoņu kompilācija, kas ir maksas serviss. Autors uzskata, ka pēc izstrādes sarežģītības un uzturamības, *Adobe AIR* platforma ir pārāka par *HTML5* platformu, jo *Adobe AIR* jauna projekta uzsākšanai nav nepieciešama vairāku paplašinājumu uzstādīšana un konfigurācija, kā tas ir *Apache Cordova* gadījumā, bet *Adobe AIR* vairums servisu, tajā skaitā paplašinājumi, ir maksas pakalpojumi. *HTML5* izstrādes pieejas trūkums ir sliktāka programmatūras koda organizēšana nekā tas ir *Adobe AIR* izstrādes pieejas gadījumā, kur tiek ievērots MVC arhitektūras modelis, kaut arī izstrādes sarežģītība ir atkarīga no ietvaru un izstrādes vides izvēles.

No mērogojamības viedokļa, abas platformas ir līdzīgas. *Adobe AIR* izstrādātāja rīkkopa (SDK) nav atvērtā pirmkoda rīkkopa, kas nozīmē, ka uz jebkurām platformas izmaiņām (piemēram, jaunu operētājsistēmas versiju), nepieciešams gaidīt līdz jaunas izstrādātāja rīkkopas izveidei, kuru izstrādā pati *Adobe,* savukārt, *HTML5* pieejas paplašinājumi ir savstarpēji atkarīgi.

Veiktspējas testi liecina, ka *HTML5* platforma ir spējīgāka algoritmiskos rēķinos, jo veica piecu miljonu skaitļu kārtošanu 11,2 sekunžu laikā, kamēr *Adobe AIR* platforma to spēja izdarīt 22,3 sekunžu laikā. Vizuālo objektu renderēšanas testos *Adobe AIR* platforma bija spējīgāka, nodrošinot 30 FPS pie 15 000 bitkaršu spraitiem, kamēr *HTML5* platforma 30 FPS robežu sasniedza jau pie 4000 bitkaršu spraitiem.

Zemāk ir aplūkojams abu izstrāžu pieeju kopīgo/atšķirīgo īpašību un priekšrocību/trūkumu salīdzinājums.

3. tabula

***HTML5* un *Adobe AIR* pieeju īpašību salīdzinājums**

Skaidrojumi: priekšrocība neitrāli daļēji priekšrocība/trūkums trūkums

Kritērijs	HTML5 platforma	Adobe AIR platforma
I1. Licences un izmaksas	Abas platformas nodrošina bezmaksas rīkus un komerciālu lietotņu izstrādi	
I2. Atbalstītās platformas	Nodrošina atbalstu septiņām mobilajām operētājsistēmām	Nodrošina atbalstu divām mobilajām operētājsistēmām
I3. Piekļuve platformu specifiskajām funkcijām	Nodrošina pamatiespējas ar iekārtas komponentēm	
I4. Ilgtermiņa iespējamība	Abi projekti ir jauni projekti, kuri tiek regulāri atjaunināti un nodrošina atbalsta servisu	
I5. Lietojamība	Abu pieeju mobilās lietotnes dzīves cikli tiek ievēroti pēc lietotnes dzīvescikla vadlīnijām un no lietojamības puses neatšķiras	
I6. Lietotnes ātrums	Izstrādes lietotne bija nelietojama vecākām mobilajām operētājsistēmas versijām	Platforma nodrošina pieeju grafiskajai kartei (*GPU*) arī vecākām mobilajām operētājsistēmas versijām
I7. Distribūcija	Nepieciešams dators ar *Mac OS X* atbalstu, bet ir iespēja atjaunināt attālināti	Nav nepieciešams dators ar *Mac OS X* atbalstu, bet nav iespēja atjaunināt attālināti
D1. Izstrādes vide	Nepieciešamas vairākas izstrādes vides/rīki	Nodrošina vienu rīku visu darbību veikšanai
D2. Vizuālais interfeiss	Nodrošina bezmaksas vizuālā interfeisa rīkus	Nodrošina maksas vizuālā interfeisa rīkus
D3. Izstrādes sarežģītība	Grūta projekta uzsākšana un konfigurēšana	Vienkārša projekta uzsākšana un testēšana
D4. Uzturamība	Koda organizēšanas trūkums	Vienkārša koda organizēšana pēc MVC arhitektūras modeļa
D5. Mērogojamība	Abas platformas ir izmaiņu atkarīgas. *Adobe AIR* programmētāja rīkkopa ir atkarīga no operētājsistēmas izmaiņām, kamēr *HTML5* platformas paplašinājumi ir savstarpēji atkarīgi	
D6. Veiktspēja	Ātri algoritmiskie rēķini, bet lēna vizuālo objektu renderēšana	Lēni algoritmiskie rēķini, bet ātra vizuālo objektu renderēšana

Kopumā abas platformas tika vērtētas pēc 13 kritērijiem, kuri bija sadalīti divās kategorijās. Salīdzinot vairumu kritēriju, rezultāti abām platformām bija vienādi. Abas platformas nodrošina bezmaksas izstrādes rīkus un bezmaksas komerciālu mobilo lietotņu izstrādi, publicēšanu. Gan *Apache Cordova*, gan *Adobe AIR* izstrādātāju rīkkopa nodrošina pamatfunkcionalitāti iekārtu interfeisu piekļuvei, piemēram, kameras funkcijai, GPS, akselerometra datu iegūšanu u.c. funkcijas. Abas izstrādes pieejas ir samērā jaunas (prezentētas 2008. un 2011.gadā) un tiek regulāri atjauninātas un pilnveidotas. Tomēr abām izstrādes pieejām bija kopīgs trūkums – tās bija atjauninājumu atkarīgas, kas nozīmē, ka abas platformas pie operētājsistēmu atjaunošanas un funkcionalitātes maiņas ir nepieciešami atjauninājumi.

HTML5 platformas priekšrocība ir vairākplatformu atbalsts, septiņu mobilo operētājsistēmu atbalsta nodrošinājums, kamēr *Adobe AIR* nodrošina atbalstu tikai divām populārākajām mobilajām operētājsistēmām. Par priekšrocību tika uzskatīts arī *HTML5* bezmaksas izstrādes rīku piedāvājums, kas ļauj uzņēmumiem bezmaksas izmantot jaunākās pieejamākās izstrādes tehnoloģijas.

Par *HTML5* trūkumiem autors uzskata daudzo izstrāžu rīku nepieciešamību. Piemēram, lai izstrādātu mobilo lietotni priekš *iOS*, ir nepieciešama izstrādes vide tīmekļa lapai un *Xcode* izstrādes vide, kas veic lietotnes kompilāciju uz *iOS* platformu. Būtisks aspekts ir projekta uzsākšana un konfigurācija. *Apache Cordova* ietvarā komandas pārsvarā tiek veiktas, izmantojot komandrindu, kas prasa papildus zināšanas šajā jomā. Turklāt vienkārša projekta gadījumā, bija nepieciešami daudzi paplašinājumi, kuri ir atkarīgi cits no cita, nodrošinot sarežģītu projekta organizēšanu. *HTML5* platformas trūkums ir nespēja nodrošināt pieeju grafiskajai kartei (*GPU*) vecākām mobilajām operētājsistēmas versijām (tiek nodrošināta sākot no *iOS 8*, *Android* 5 versijām), kas būtiski samazina lietotnes veiktspēju.

Veiktspējas testi liecina, ka *HTML5* platforma strādā ātrāk uz algoritmiskiem – matemātiskiem rēķiniem, bet lēnāk uz vizuālo objektu renderēšanu. Būtiska *HTML5* platformas priekšrocība ir lietotnes satura atjaunināšana attālināti, kas nav pieejams *Adobe AIR* platformai.

Adobe AIR platformas priekšrocība ir viena izstrādes rīka nepieciešamība, lai izstrādātu mobilās lietotnes vienai platformai, ari *iOS* platformai, kam parasti ir nepieciešams dators ar *Mac OS X* operētājsistēmu. *Adobe AIR* izstrādātāju rīkkopa nodrošina *iOS* platformas izstrādātāju rīkkopu, kas ļauj mobilās lietotnes izstrādāt arī no citas operētājsistēmas datora. Tā kā pamata izstrādātāju rīkkopa (*Adobe AIR*) nodrošina pamatfunkcionalitāti iekārtu interfeisa izmantošanai, tad projekta datņu skaits ir mazāks nekā *HTML5* platformai, kas ļauj efektīvāk

uzturēt izstrādes projektu. *Adobe AIR* atbalsta pieeju grafiskajai kartei (*GPU*) vecākām mobilajām operētājsistēmām versijām, kas palielina lietotnes veiktspēju.

Adobe AIR pamattrūkums ir mazais platformu atbalstu skaits, nodrošinot atbalstu tikai divām mobilo operētājsistēmām (*iOS, Android*), atšķirībā no *HTML5* platformas, kas nodrošina atbalstu septiņām platformām. Diemžēl *Adobe AIR* tiek uzskatīts par komerciālu projektu, tādēļ pārsvarā visi pakalpojumi, izstrādes vides, paplašinājumi ir maksas pakalpojumi.

Veiktspējas testi liecina, ka *Adobe AIR* platforma strādā lēnāk uz algoritmiskiem – matemātiskiem rēķiniem, bet strādā ātrāk uz vizuālo objektu renderēšanu.

Darba gala rezultāti bija līdzīgi citu pētījumu rezultātiem [Lukasavage2011, Kaza2011, Cowart2013] . Autors uzskata, ka *HTML5* platforma ir piemērotāka vienkāršiem projektiem, kuriem vizuālo objektu renderēšana ir sekundārs faktors, bet ir nepieciešams plašs platformu atbalsts, savukārt, *Adobe AIR* platforma ir piemērotāka vizuāli sarežģītiem projektiem kā spēlēm, kur vizuālo objektu renderēšana ir svarīgs faktors, bet platformu nodrošinājums ir sekundārs faktors.

SECINĀJUMI

Mobilo lietotņu tirgus ir plašs un ar katru nākamo gadu lietotņu skaits strauji pieaug. 2014. gadā 80% pasaules iedzīvotājiem bija vismaz viens viedtālrunis un 2015. gadā ir plānots pārdot vienu miljardu viedtālruņu, kas ir divas reizes vairāk kā plānotais personālo datoru pārdošanu skaits. Viedtālruņu lietotājs vidēji 30 stundas mēnesī lieto viedtālruni, izmantojot vidēji 20 lietotnes mēnesī [Clifford2014].

Hibrīdā mobilā lietotne ir lietotnes paveids, kas apvieno divas tehnoloģijas – vairākplatformu tehnoloģiju bāzi kā tīmekļa implementāciju, un dzimto mobilo lietotņu tehnoloģijas [Rouse2014]. Tas nozīmē, ka visām mobilajām operētājsistēmām tiek nodrošināta viena implementācija (lietotnes izstrādes kods) ar iespēju izmantot dzimto mobilo lietotņu saskarni (API). Hibrīdās mobilās lietotnes ir pieprasītas vairākplatformu iespēju un zemo izmaksu dēļ.

Pēc dzimto platformu lietotņu un hibrīdo mobilo lietotņu salīdzinājuma autors secina, ka abām tehnoloģijām ir savas priekšrocības un trūkumi, kuras ir jāvērtē konkrētos gadījumos. Autors secina arī to ka, hibrīdās mobilās lietotnes būtiski neatpaliek no dzimtās platformas lietotnēm iespēju ziņā un spēj konkurēt ar tām.

Hibrīdajām mobilajām lietotnēm pasaulē ir dažādas izstrādes pieejas. Populārākās hibrīdo mobilo lietotņu izstrāžu pieejas ir tīmekļa izstrādes pieeja un bitkodu izstrādes pieeja [Raj2014]. Tīmekļa izstrādes pieeja balstās uz tīmekļa tehnoloģijām un implementācijām (tiek izmantots tīmekļa pārlūks), savukārt, bitkoda izstrādes pieeja veic augsta līmeņa programmēšanas valodu (piemēram, *Adobe ActionScript* vai *C#*) kompilāciju uz operētājsistēmai saprotamu mašīnkodu.

Kopumā abas platformas tika vērtētas pēc 13 kritērijiem, kuri bija sadalīti divās kategorijās. Salīdzinot vairumu kritēriju, rezultāti abām platformām bija vienādi. Darba gala rezultāti bija līdzīgi citu pētījumu rezultātiem [Lukasavage2011, Kaza2011, Cowart2013] . Autors uzskata, ka *HTML5* platforma ir piemērotāka vienkāršiem projektiem, kuriem vizuālo objektu renderēšana ir sekundārs faktors, bet ir nepieciešams plašs platformu atbalsts, savukārt, *Adobe AIR* platforma ir piemērotāka vizuāli sarežģītiem projektiem kā spēlēm, kur vizuālo objektu renderēšana ir svarīgs faktors, bet platformu nodrošinājums ir sekundārs faktors.

Attīstot šo pētījumu, varētu veikt citu hibrīdo mobilo lietotņu izstrādes pieeju salīdzināšanu kā *HaXe*, *Appcelerator Titanium.*

IZMANTOTĀ LITERATŪRA

[Adobe2011] **Adobe** *Developing Mobile Applications with ADOBE® FLEX® 4.6 and ADOBE® FLASH® BUILDERTM 4.6* ,Adobe, 2011, 1-8lpp

[Adobe2015a] **Adobe** *What is Adobe AIR?* [tiešsaiste] – [atsauce 22.03.2015.].
Pieejams: http://www.adobe.com/products/air.html

[Adobe2015b] **Adobe** *Gaming Frameworks* [tiešsaiste] – [atsauce 25.04.2015.].
Pieejams: http://www.adobe.com/devnet/games/frameworks.html

[Android2015] **Android** *The world's most popular mobile platform* [tiešsaiste] –[atsauce 08.03.2015.].
Pieejams: http://developer.android.com/about/index.html

[Apple2014a] **Apple** *App Development Process* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams:https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/AppDevelopmentProcess.html

[Apple2013b] **Apple** *Model-View-Controller* [tiešsaiste] – [atsauce 24.03.2015.].
Pieejams:
http://developer.apple.com/library/ios/#documentation/general/conceptual/devpedia-cocoacore/MVC.html

[Bansod2011] **Banson Aditya** *Developing for iOS using Flash Professional* [tiešsaiste] – [atsauce 22.03.2015.].
Pieejams: http://www.adobe.com/devnet/logged_in/abansod_iphone.html

[Beal2014a] **Beal Vangie** *Mobile operating system (OS)* [tiešsaiste] – [atsauce 18.03.2015.].
Pieejams: http://www.webopedia.com/TERM/M/mobile_operating_system.html

[Beal2014b] **Beal Vangie** *Apple iOS* [tiešsaiste] – [atsauce 18.03.2015.].
Pieejams: http://www.webopedia.com/TERM/A/apple_ios.html

[Brown2014] **Brown Carl** *App Accomplished: Strategies for App Development Success,* Addison-Wesley Professional, 2014, 104-107lpp

[Bunyan2015] **Bunyan Karl** *Build AN HTML5 Game*, William Pollok, 2015, 15-17lpp

[Chell2015] **Chell Dominic** *The Mobile Application Hacker's Handbook*
,John Wiley & Sons, 2015, 729-743lpp

[Clay2015] **Clay.io** *HTML5 game engines* [tiešsaiste] – [atsauce 25.04.2015.].
Pieejams: https://html5gameengine.com/

[Clark2011] **Clark Scott** *What Does the Google Dart Programming Language Mean for Web Developers?* [tiešsaiste] – [atsauce 21.03.2015.].
Pieejams: http://www.entrepreneur.com/article/236832
[Clifford2014] **Clifford Catherine** *By 2017, the App Market Will Be a $77 Billion Industry (Infographic)* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams:http://www.htmlgoodies.com/html5/tutorials/what-does-the-google-dart-programming-language-mean-for-web-developers.html
[Cordova2015] **Apache Cordova** *Platform for building native mobile applications using HTML, CSS mand JavaScript* [tiešsaiste] – [atsauce 06.05.2015.].
Pieejams: https://cordova.apache.org/#about
[Cowart2013] **Cowart Jim** *Pros and Cons of the Top 5 Cross-Platform Tools* [tiešsaiste] – [atsauce 24.05.2015.].
Pieejams: http://www.developereconomics.com/pros-cons-top-5-cross-platform-tools
[CreateJS2014] **CreateJS** *WEBGL SUPPORT IN EASELJS* [tiešsaiste] – [atsauce 06.05.2015.].
Pieejams: http://blog.createjs.com/webgl-support-easeljs/
[Dart2015] **Dart** *Get Started* [tiešsaiste] – [atsauce 21.03.2015.].
Pieejams: https://www.dartlang.org
[Davey2013] **Davey Richard** *How to Learn the Phaser HTML5 Game Engine* [tiešsaiste] – [atsauce 25.04.2015.].
Pieejams: http://gamedevelopment.tutsplus.com/articles/how-to-learn-the-phaser-html5-game-engine--gamedev-13643
[DZone2014] **DZone** *Mobile Development* , DZONE RESEARCH, 2014, 3-10lpp
[EngineersGarage2014] **Engineers Garage** *What is Android* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams: http://www.engineersgarage.com/articles/what-is-android-introduction
[FlashDelevop2015] **Flash Develop** [tiešsaiste] – [atsauce 26.03.2015.].
Pieejams: http://flashdevelop.org/
[GeoSpatial2012] **GeoSpatial** *Anatomy of a Hybrid Mobile GIS Application* [tiešsaiste] – [atsauce 21.03.2015.].
Pieejams: http://www.geospatialtraining.com/blog/index.php/anatomy-of-a-hybrid-mobile-gis-application/
[Graham2012] **Graham Charles** *HTML5 - Unabridged Guide* Tebbo, 2012, 1-23lpp

[Heitkötter2013] **Heitkötter Henning, Hanschke Sebastian, Majchrzak Tim A.** *Evaluating Cross-Platform Development Approaches for Mobile Applications* [tiešsaiste] – [atsauce 03.05.2015.].
Pieejams: http://www3.nd.edu/~cpoellab/teaching/cse40814/crossplatform.pdf
[IDC2014] **IDC Analyze future** *Smartphone OS Market Share, Q4 2014* [tiešsaiste] – [atsauce 18.03.2015.].
Pieejams: http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[Imbert2013] **Imbert Thibault** *Introducing Starling* O'Reilly, 2013, 1-5lpp
[Kaza2011] **Kaza Avinash** *Adobe AIR vs. HTML5 for Mobile Apps* [tiešsaiste] – [atsauce 24.05.2015.].
Pieejams: http://avinashkaza.com/blog/?p=273
[Korf2012] **Korf Mario, Oksman Eugene** *Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options* [tiešsaiste] – [atsauce 16.04.2015.].
Pieejams:
https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options
[Kshitiz2012] **Kshitiz Gupta** *AOT or Interpreter* [tiešsaiste] – [atsauce 22.03.2015.].
Pieejams: http://blogs.adobe.com/airodynamics/2012/07/04/aot-or-interpreter/
[Lee2013] **Lee Joel** *What Is HTML5, And How Does It Change The Way I Browse?* [tiešsaiste] – [atsauce 18.03.2015.].
Pieejams: http://www.makeuseof.com/tag/what-is-html5-and-how-does-it-change-the-way-i-browse-makeuseof-explains/
[Lukasavage2011] **Lukasavage Tony** *Review: Appcelerator vs. PhoneGap vs. Adobe Air* [tiešsaiste] – [atsauce 25.05.2015.].
Pieejams: http://tonylukasavage.com/blog/2011/01/18/review--appcelerator-vs--phonegap-vs--adobe-air/
[Lūse2013] **Lūse Aija** *HTML5 un CSS3 skolā* [tiešsaiste] – [atsauce 18.03.2015.].
Pieejams: http://startit.lv/files/conf2013/Aija_Luse_HTML5_un_CSS3_skolas.pptx
[NetBeans2015] **NetBeans IDE** [tiešsaiste] – [atsauce 26.03.2015.].
Pieejams: https://netbeans.org/
[Phaser2015] **Phaser.io** *Desktop and Mobile HTML5 game framework* [tiešsaiste] – [atsauce 25.04.2015.].
Pieejams: http://phaser.io/

[Pronschinske2014] **Pronschinske Mitch** *The State of Native vs. Web vs. Hybrid* [tiešsaiste] – [atsauce 16.04.2015.].
Pieejams: http://java.dzone.com/articles/state-native-vs-web-vs-hybrid
[Philomena2014] **Philomena Dolla, Nimisha, Govinda Gupta** *Faster compiling with AIR for iOS* [tiešsaiste] – [atsauce 22.03.2015.].
Pieejams: http://www.adobe.com/devnet/air/articles/ios-packaging-compiled-mode.html
[PhoneGap2012] **PhoneGap** *Explained Visually* [tiešsaiste] – [atsauce 21.03.2015.].
Pieejams: http://phonegap.com/2012/05/02/phonegap-explained-visually/
[Raj2014] **Raj Jay** *The Top 7 Hybrid Mobile App Frameworks* [tiešsaiste] – [atsauce 21.03.2015.].
Pieejams: http://www.sitepoint.com/top-7-hybrid-mobile-app-frameworks/
[Rouse2014] **Rouse Margaret** *Hybrid application (hybrid app)* [tiešsaiste] – [atsauce 21.03.2015.].
Pieejams: http://searchsoftwarequality.techtarget.com/definition/hybrid-application-hybrid-app
[Gamua2015] **Gamua** *Starling* [tiešsaiste] – [atsauce 26.04.2015.].
Pieejams: http://gamua.com/starling/
[Statista2015a] **Statista** *Number of mobile app downloads worldwide from 2009 to 2017 (in millions)* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams: *http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/*
[Statista2015b] **Statista** *Worldwide mobile app revenues from 2011 to 2017 (in billion U.S. dollars)* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams: *http://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/*
[Valdoz2014] **Valdoz Milagros** *The 10 Top Selling Smartphones in the World 2014* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams: http://www.insidermonkey.com/blog/the-10-top-selling-smartphones-in-the-world-2014-334251/
[Viswanathan2014] **Viswanathan Priya** *What is a mobile application?* [tiešsaiste] – [atsauce 08.03.2015.].
Pieejams: *http://mobiledevices.about.com/od/glossary/g/What-Is-A-Mobile-Application.htm*

[Xamarin2015] **Xamarin** *Understanding the Xamarin Mobile Platform* [tiešsaiste] – [atsauce 25.03.2015.].
Pieejams:http://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_1_-_understanding_the_xamarin_mobile_platform/

PIELIKUMS

1. *HTML5* mobilās lietotnes "Prātnieks" programmatūras koda fragments

Zemāk ir aplūkojams *HTML5* mobilās lietotnes "Prātnieks" programmatūras koda piemērs, kas apraksta funkcijas "Draugu palīdzība" vizualizācija.

```
    /* Konstruktors */
    function HelpOthersPopup(game, model)
    {
        this._model = model; /*Spēles modelis */
        this._bars;
        this._barWidth;
        this._barCount = 4;
        this._barMargin = 10;
        this._barColors = [0xab0b0f, 0x9b7722, 0x6296c1, 0xc7b040];
        this._barMaxHeight;
        this._textWidth = 100;
        this._topText;
        this._bottomText;
        this._barOffset = 70;
        PopupContentBase.call(this, game); /* Super construktors */
    }

    HelpOthersPopup.prototype = Object.create(PopupContentBase.prototype);
    HelpOthersPopup.prototype.constructor = HelpOthersPopup;
    /* Galvenā funkcija */
    HelpOthersPopup.prototype.addContent = function ()
    {
        PopupContentBase.prototype.addContent.call(this); /* override*/
        this._initData(); /* Datu inicializācija*/
        this._addTexts(); /* Pievieno tekstus */
        this._addBars(); /* Pievieno grafiku */
    };

    HelpOthersPopup.prototype._initData = function ()
    {
        this._barWidth = ((this._containerWidth - this._textWidth -
(this._barMargin * (this._barCount - 1))) / this._barCount);
        this._barMaxHeight = this._containerHeight - this._barOffset;
    };

    HelpOthersPopup.prototype._addTexts = function ()
    {
        this._topText = new Phaser.Text(this.game, 0, 0, "100%");
        this._topText.font = 'Amarante';
        this._topText.fontSize = 14*AppConstants.RESOLUTION;
        this._topText.fill = '#FFFFFF';
        this.container.addChild(this._topText);

        this._bottomText = new Phaser.Text(this.game, 0, 0, "0%");
        this._bottomText.font = 'Amarante';
```

```
        this._topText.y = 0;
        this._bottomText.x = this._topText.x + this._topText.width -
this._bottomText.width;
        this._bottomText.y = this._containerHeight - this._bottomText.height -
35;
    };

    HelpOthersPopup.prototype._addBars = function ()
    {
        this._bars = [];

        var source = this._model.availiableAnswersForCurrentQuestion; /*Atbilžu
varianti*/
        var length = source.length;
        var answer;

        var totalPercent = 100;
        var correctAnswerWeight =
this._getCorrectAnswerWeight(this._model.level); /*Pareizās atbildes svars
(atkarībā no jautājuma kārtas numurs)*/
        var currentAnswerWeight = 0;
        /*Pārējo atbilžu svars, kas paliek pāri no pareizas atbildes svara*/
        var otherAnswerWeights = this._getOtherAnswerWeights(length - 1,
totalPercent - correctAnswerWeight);
        var bar;
        var answerIndex;

        for (i = 0; i < length; i++)
        {
            answer = source[i];
            answerIndex = parseInt(answer.id);
               /*Pareizās atbildes grafiks un svars*/
            if (answer.id.toLocaleString() ===
this._model.currentQuestionData.correctAnswer.toLocaleString())
            {

                totalPercent -= correctAnswerWeight;
                currentAnswerWeight = correctAnswerWeight;
            }
            else
            {
                currentAnswerWeight = otherAnswerWeights.pop();
            }

            bar = new Bar(this.game);
            this.container.addChild(bar);
            bar.createBar(this._barWidth, this._barMaxHeight *
currentAnswerWeight / 100, (answerIndex + 1).toString(), this._barColors[i],
currentAnswerWeight);
            bar.y = this._containerHeight - bar.totalHeight;
            bar.x = this._textWidth + this._barWidth * answerIndex +
this._barMargin * answerIndex;
        }
    };
```

```
/*Aprēķina pārējo atbildes svaru atkarībā no pareizās atbildes svara.*/
HelpOthersPopup.prototype._getOtherAnswerWeights = function (parts, total)
{
   /*Ir tikai 2 atbildes - 1 pareizā un 1 nepareizā*/
    if (parts === 1)
        return [total];

    var result = [];

    var rest = total;
    var current;

    for (i = 0; i < parts; i++)
    {
        if (i !== parts - 1)
        {
           /*Nejaušs aprēķināts svars atkarībā no kopējā pieejamā svara*/
            current = ExtendedMath.randomInRange(rest, 0);
            rest -= current;
            result.push(current);
        }
        else
        {
            result.push(rest);
        }
    }

    return result;
};

/*Aprēķina pareizās atbildes svaru atkarībā no jautājuma kārtas nr.*/
HelpOthersPopup.prototype._getCorrectAnswerWeight = function (level)
{
   /*Nedegošo summu līmeņi*/
    if (level === 1)
    {
        return ExtendedMath.randomInRange(100, 80);
    }
    else if (level === 2)
    {
        return ExtendedMath.randomInRange(80, 60);
    }

    return ExtendedMath.randomInRange(60, 30);
};

/*Destruktors*/
HelpOthersPopup.prototype.destroy = function ()
{
    this._model = null;
```

```
/*Destruktors*/
HelpOthersPopup.prototype.destroy = function ()
{
    this._model = null;

    if (this._bars)
    {
        this._destroyBars(this._bars);
        this._bars.length = 0;
        this._bars = null;
    }

    if (this._topText)
        this._topText.destroy();
    this._topText = null;
    if (this._bottomText)
        this._bottomText.destroy();
    this._bottomText = null;

    PopupContentBase.prototype.destroy.call(this);
};

/*Grafika destruktors*/
HelpOthersPopup.prototype._destroyBars = function (source)
{
    var item;
    var length;

    if (source)
    {
        length = source.length;
        for (i = 0; i < length; i++)
        {
            item = source[i];
            if (item)
                item.destroy();
        }
    }
};
```

2.1. att. **HTML5 mobilās lietotnes "Prātnieks" programmatūras koda fragments**

HTML5 platformas lietotnei tika izmantotas **26** datnes ar **1892** programmatūras koda rindiņām un kompilētā mobilā datne ir **2.5MB** liela.

2. *Adobe AIR* mobilās lietotnes "Prātnieks" programmatūras koda fragments

Zemāk ir aplūkojams *Adobe AIR Adobe AIR* mobilās lietotnes "Prātnieks" programmatūras koda piemērs, kas apraksta funkcijas "Draugu palīdzība" vizualizācija.

```
package lv.pratnieks.view
{
   import lv.pratnieks.components.Bar;
   import lv.pratnieks.components.PopupContentBase;
   import lv.pratnieks.model.GameModel;
   import lv.pratnieks.utils.ExtendedMath;
   import lv.pratnieks.vo.AnwerVO;
   import starling.text.TextField;

   /**
    * ...
    * @author Armands Baurovskis
    */
   public class HelpOthersPopup extends PopupContentBase
   {
           private var model:GameModel; /*Spēles modelis*/
           private var bars:Vector.<Bar>;
           private var barWidth:uint;
           private var barCount:uint = 4;
           private var barMargin:int = 10;
           private var barColors:Array = [0xab0b0f, 0x9b7722, 0x6296c1,
0xc7b040]; /*Grafika krāsas*/
           private var barMaxHeight:uint;
           private var textWidth:uint = 50;
           private var topText:TextField;
           private var bottomText:TextField;
           private var barOffset:int = 70;

           public function HelpOthersPopup(model:GameModel)
           {
                   super();
                   this.model = model;
           }
           /*Spēles modelis*/
           override protected function addContent():void
           {
                   super.addContent();
                   initData();
                   addTexts();
                   addBars();
           }
```

```
                private function initData():void
                {
                        barWidth = ((containerWidth-textWidth - (barMargin *
(barCount - 1))) / barCount);
                        barMaxHeight = containerHeight - barOffset;
                }

                private function addTexts():void
                {
                        topText = new TextField(textWidth, barMaxHeight, "100 %",
"Amarante", 14, 0xFFFFFF);
                        topText.hAlign = "center";
                        topText.vAlign = "top";
                        container.addChild(topText);

                        bottomText = new TextField(textWidth, barMaxHeight, "0 %",
"Amarante", 14, 0xFFFFFF);
                        bottomText.hAlign = "center";
                        bottomText.vAlign = "bottom";
                        container.addChild(bottomText);

                        topText.y = bottomText.y = containerHeight - barMaxHeight-
35;
                }
                /*Grafika izveide*/
                private function addBars():void
                {
                        bars = new Vector.<Bar>();

                        var source:Vector.<AnwerVO> =
model.availiableAnswersForCurrentQuestion; /*Atbilžu variantu skaits*/
                        var length:int = source.length;
                        var answer:AnwerVO; /*Atbildes objekts*/

                        var totalPercent:uint = 100;
                        var correctAnswerWeight:uint =
getCorrectAnswerWeight(model.level); /*Pareizās atbildes svars (atkarībā no
jautājuma kārtas numurs)*/
                        var currentAnswerWeight:uint;
                        /*Pārējo atbilžu svars, kas paliek pāri no pareizas
atbildes svara*/
                        var otherAnswerWeights:Array =
getOtherAnswerWeights(length - 1, totalPercent - correctAnswerWeight);

                        var bar:Bar;
                        var answerIndex:uint;
```

```
                        for (var i:int = 0; i <length ; i++)
                        {
                                answer = source[i];
                                answerIndex = uint(answer.id);
                                /*Pareizās atbildes grafiks un svars*/
                                if (answer.id ==
model.currentQuestionData.correctAnswer)
                                {
                                        totalPercent -= correctAnswerWeight;
                                        currentAnswerWeight = correctAnswerWeight;
                                }
                                else
                                {
                                        currentAnswerWeight =
otherAnswerWeights.pop();
                                }

                                bar = new Bar();
                                container.addChild(bar);
                                bar.create(barWidth,
barMaxHeight*currentAnswerWeight/100, (answerIndex+1).toString(),
barColors[i],currentAnswerWeight);
                                bar.y = containerHeight - bar.height;
                                bar.x =
textWidth+barWidth*answerIndex+barMargin*answerIndex;
                        }

                }
                /*Aprēķina pārējo atbildes svaru atkarībā no pareizās atbildes
svara.*/
                private function getOtherAnswerWeights(parts:uint,
total:uint):Array
                {
                        /*Ir tikai 2 atbildes - 1 pareizā un 1 nepareizā*/
                        if (parts == 1)
                                return [total];

                        var result:Array = [];

                        var rest:uint = total;
                        var current:uint;
```

```
        /*Aprēķina pareizās atbildes svaru atkarībā no jautājuma kārtas nr.*/
                private function getCorrectAnswerWeight(level:uint):uint
                {
                        /*Nedegošo summu līmeņi*/
                        if (level == 1)
                        {
                                return ExtendedMath.randomInRange(100, 80);
                        }
                        else if (level == 2)
                        {
                                return ExtendedMath.randomInRange(80, 60);
                        }

                        return ExtendedMath.randomInRange(60, 30);
                }
                /*Destruktors*/
                override public function destroy():void
                {
                        super.destroy();
                        model = null;

                        if (bars)
                        {
                                destroyBars(bars);
                                bars.length = 0;
                                bars = null;
                        }

                        if (topText)
                                topText.dispose();
                        topText = null;
                        if (bottomText)
                                bottomText.dispose();
                        bottomText = null;
                }
                /*Grafika destruktors*/
                private function destroyBars(source:Vector.<Bar>):void
                {
                        if (source)
                        {
                                for each (var item:Bar in source)
                                {
                                        if (item)
                                                item.destroy();
                                }
                        }
                }
        }
}
```

2.1. att. **Adobe AIR mobilās lietotnes "Prātnieks" programmatūras koda fragments**

Uz *Adobe AIR* platformas lietotnei "Prātnieks" tika izmantotas **39** datnes ar **2800** programmatūras koda rindiņām, bet kompilētā mobilā datne ir **10MB** liela.

3. Licences

Zemāk ir uzskaitītas programmatūras ietvaru, attēlu un fonta licences;

3.1. Phaser ietvara licence

Phaser is distribted under the MIT License. This covers the framework itself. Please see the trademark policy for details about using the Phaser logo or branding.
Copyright © 2015 Richard Davey, Photon Storm Ltd.
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and / or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

3.2. Starling ietvara licence

Copyright 2011-2014 Gamua. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
THIS SOFTWARE IS PROVIDED BY GAMUA "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL GAMUA OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of Gamua.

3.3. Amarante fonta licence

Copyright (c) 2012 by Sorkin Type Co (www.sorkintype.com), with Reserved Font Name "Amarante".
This Font Software is licensed under the SIL Open Font License, Version 1.1.
This license is copied below, and is also available with a FAQ at:
http://scripts.sil.org/OFL

SIL OPEN FONT LICENSE Version 1.1 - 26 February 2007

PREAMBLE
The goals of the Open Font License (OFL) are to stimulate worldwide
development of collaborative font projects, to support the font creation
efforts of academic and linguistic communities, and to provide a free and
open framework in which fonts may be shared and improved in partnership
with others.

The OFL allows the licensed fonts to be used, studied, modified and
redistributed freely as long as they are not sold by themselves. The
fonts, including any derivative works, can be bundled, embedded,
redistributed and/or sold with any software provided that any reserved
names are not used by derivative works. The fonts and derivatives,
however, cannot be released under any other type of license. The
requirement for fonts to remain under this license does not apply
to any document created using the fonts or their derivatives.

DEFINITIONS
"Font Software" refers to the set of files released by the Copyright
Holder(s) under this license and clearly marked as such. This may
include source files, build scripts and documentation.

"Reserved Font Name" refers to any names specified as such after the
copyright statement(s).

"Original Version" refers to the collection of Font Software components as
distributed by the Copyright Holder(s).

"Modified Version" refers to any derivative made by adding to, deleting,
or substituting -- in part or in whole -- any of the components of the
Original Version, by changing formats or by porting the Font Software to a
new environment.

"Author" refers to any designer, engineer, programmer, technical
writer or other person who contributed to the Font Software.

PERMISSION & CONDITIONS
Permission is hereby granted, free of charge, to any person obtaining
a copy of the Font Software, to use, study, copy, merge, embed, modify,
redistribute, and sell modified and unmodified copies of the Font

Software, subject to the following conditions:

1) Neither the Font Software nor any of its individual components,
in Original or Modified Versions, may be sold by itself.

2) Original or Modified Versions of the Font Software may be bundled,
redistributed and/or sold with any software, provided that each copy
contains the above copyright notice and this license. These can be
included either as stand-alone text files, human-readable headers or
in the appropriate machine-readable metadata fields within text or
binary files as long as those fields can be easily viewed by the user.

3) No Modified Version of the Font Software may use the Reserved Font
Name(s) unless explicit written permission is granted by the corresponding
Copyright Holder. This restriction only applies to the primary font name as
presented to the users.

4) The name(s) of the Copyright Holder(s) or the Author(s) of the Font
Software shall not be used to promote, endorse or advertise any
Modified Version, except to acknowledge the contribution(s) of the
Copyright Holder(s) and the Author(s) or with their explicit written
permission.

5) The Font Software, modified or unmodified, in part or in whole,
must be distributed entirely under this license, and must not be
distributed under any other license. The requirement for fonts to
remain under this license does not apply to any document created
using the Font Software.

TERMINATION
This license becomes null and void if any of the above conditions are
not met.

DISCLAIMER
THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT
OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL THE
COPYRIGHT HOLDER BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM
OTHER DEALINGS IN THE FONT SOFTWARE.

3.4. Attēlu licence

Bakalaura darbs „HTML5 un Adobe AIR hibrīdo mobilo lietotņu izstrādes salīdzinājums” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Armands Baurovskis _____ ___.06.2015.

Rekomendēju darbu aizstāvēšanai
Vadītājs: asociētais profesors, Dr. dat. Uldis Straujums _____ ___.06.2015.

Recenzents: docents Dr. dat. Sergejs Kozlovičs

Darbs iesniegts Datorikas fakultātē 01.06.2015.
Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalauru galu pārbaudījuma komisijas sēdē
___.06.2015. prot. Nr. _____

Komisijas sekretārs(-e): _____