



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ
НАУКА У НОВОМ САДУ



Душко Алексић

IONIC ОКВИР
ЗА РАЗВОЈ ХИБРИДНИХ
МОБИЛНИХ АПЛИКАЦИЈА

МАСТЕР РАД

Нови Сад, 2016



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ
НАУКА

21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:

Идентификациони број, ИБР:

Тип документације, ТД:

Монографска публикација

Тип записа, ТЗ:

Текстуални штампани документ

Врста рада, ВР:

Мастер рад

Аутор, АУ:

Душко Алексић

Ментор, МН:

Др Стеван Гостојић

Наслов рада, НР:

Ionic оквир за развој хибридних мобилних апликација

Језик публикације, ЈП:

Српски

Језик извода, ЈИ:

Српски / енглески

Земља публиковања, ЗП:

Република Србија

Уже географско подручје, УГП:

Војводина

Година, ГО:

2016

Издавач, ИЗ:

Ауторски репринт

Место и адреса, МА:

Нови Сад, Факултет техничких наука
Трг Доситеја Обрадовића 6

Физички опис рада, ФО:

7 / 60 / 0 / 9 / 26 / 0 / 16

(поглавља/страна/
чланак/глава/справка/табела/прилога)

Научна област, НО:

Електотехничко и рачунарско инжењерство

Научна дисциплина, НД:

Програмирање мобилних апликација

Предметна одредница/Кључне речи, ПО:

мобилне апликације, хибридне апликације, Ionic, Cordova, AngularJS, JavaScript

УДК

Чува се, ЧУ:

Библиотека Факултета техничких наука
Трг Доситеја Обрадовића 6, 21000 Нови Сад

Важна напомена, ВН:

Извод, ИЗ:

Тема мастер рада спада у дисциплину програмирања мобилних апликација. У њему су анализирани оквири за развој хибридних мобилних апликација. Међу анализираним оквирима је изабран Ionic оквир. Описан је поступак развоја хибридне мобилне апликације коришћењем овог оквира, као и апликација која је резултат овог процеса.

Датум прихватања теме, ДП:

Датум одбране, ДО:

Чланови комисије, КО: Председник:

др Мирослав Зарић, доцент,
Факултет техничких наука, Нови Сад

Члан:

др Имре Лендак, доцент,
Факултет техничких наука, Нови Сад

Потпис
ментора

Члан, ментор:

др Стеван Гостојић, доцент,
Факултет техничких наука, Нови Сад

Образац Q2.NA.04-05 - Издање 1



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES

21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO:	
Identification number, INO:	
Document type, DT:	Monographic publication
Type of record, TR:	Textual material
Contents code, CC:	Master thesis
Author, AU:	Dusko Aleksic
Mentor, MN:	Dr Stevan Gostojic
Title, TI:	Ionic hybrid mobile app framework
Language of text, LT:	Serbian
Language of abstract, LA:	Serbian / English
Country of publication, CP:	Republic of Serbia
Locality of publication, LP:	Vojvodina
Publication year, PY:	2016
Publisher, PB:	Author's reprint
Publication place, PP:	Novi Sad, Faculty of technical sciences, Trg Dositeja Obradovica 2
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendixes)	7 / 60 / 0 / 9 / 26 / 0 / 16
Scientific field, SF:	Electrical and computer engineering
Scientific discipline, SD:	Mobile app development
Subject/Key words, S/KW:	mobile apps, hybrid apps, Ionic, Cordova, AngularJS, JavaScript
UC	
Holding data, HD:	Library of the Faculty of technical sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, N:	

Abstract, **AB:**
The subject of the master thesis belongs to the discipline of mobile app development. The thesis analyses hybrid mobile app development frameworks and chooses Ionic framework as the framework best suited for development of mobile apps. It describes the process of developing mobile app using this framework and the app that is the result of this process.

Accepted by the Scientific Board on, **ASB:**

Defended on, **DE:**

Defended Board, DB:	President: Miroslav Zaric, PhD, ass. prof., Faculty of Technical Sciences, Novi Sad	
Member:	Imre Lendak, PhD, ass. prof., Faculty of Technical Sciences, Novi Sad	Menthor's sign
Member, Mentor:	Stevan Gostojic, PhD, ass. prof., Faculty of Technical Sciences, Novi Sad	

Obrazac **Q2.HA.04-05** - Izdanje 1



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ
НАУКА

21000 НОВИ САД, Трг Доситеја Обрадовића 6

ЗАДАТАК ЗА МАСТЕР РАД

Број:

Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. Др Мирослав Поповић

Студент:	Душко Алексић	Број индекса:	E2 40/2014			
Област:	Електротехничко и рачунарско инжењерство					
Ментор:	Др Стеван Гостојић					
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА						
ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:						
<ul style="list-style-type: none">- проблем – тема рада;- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;						

НАСЛОВ МАСТЕР РАДА:

Ionic оквир за развој хибридних мобилних апликација

ТЕКСТ ЗАДАТКА:

Анализирати оквире за развој хибридних мобилних апликација. Специфицирати захтеве мобилне апликације за руковање фотографијама и албумима. Дизајнирати специфицирану мобилну апликацију. Имплементирати дизајн коришћењем изабраног оквира. Тестирати имплементирану апликацију. Документовати апликацију и процес њеног развоја.

Руководилац студијског програма:	Ментор рада:

Примерак за: О - Студента; О - Ментора

САДРЖАЈ

1.	УВОД	11
1.1	Класификација мобилних апликација	11
1.2	Оквири за развој хибридних апликација.....	12
2.	IONIC	15
2.1	Cordova	15
2.1.1	Cordova прикључци	16
2.2	AngularJS.....	17
2.2.1	Основни концепти у AngularJS-у	17
2.3	Основе Ionic оквира	21
2.3.1	Почетак рада са Ionic оквиром	21
2.3.2	Генерирање новог пројекта	22
2.3.3	Структура апликације	23
2.3.4	Главне компоненте корисничког интерфејса	24
2.3.5	Инсталација Cordova прикључака	28
2.3.6	Покретање апликације	28
2.3.7	Експортовање апликације за мобилну продавницу	29
3.	СПЕЦИФИКАЦИЈА ЗАХТЕВА.....	31
4.	ДИЗАЈН	37
4.1	Дијаграм класа	37
4.2	Дијаграм распореда	38
4.3	Дијаграм секвенци	39
4.3.1	Дијаграм секвенце за регистрацију корисника.....	39
4.3.2	Дијаграм секвенце за промену албума	40
4.3.3	Дијаграм секвенце за брисање фотографије.....	41
5.	ИМПЛЕМЕНТАЦИЈА	43
5.1	Почетак рада	43
5.2	Рад са базом података	43
5.3	Унос података	45
5.4	Кратке поруке.....	46
5.5	Фотографисање	47
5.6	Преузимање локације уређаја.....	48
5.7	Рад са фајл системом	48
6.	ДЕМО.....	51
7.	ЗАКЉУЧАК	57
	ЛИТЕРАТУРА	59
	БИОГРАФИЈА	60

1. УВОД

Већину мобилних уређаја покрећу два доминантна оперативна система: *Android* [1] и *iOS* [2]. Савремени мобилни оперативни системи омогућили су релативно једноставан развој мобилних апликација, које је могуће извршити на великом броју различитих типова уређаја (мобилним телефонима, табличним рачунарима, паметним сотовима, телевизорима, аутомобилима, итд.). Ово је створило велику експанзију тржишта мобилних аликација, које је данас једно од највећих тржишта софтвера. Временом се искристалисало неколико парадигми за развој мобилних апликација.

1.1 Класификација мобилних апликација

Постоје три основна типа мобилних апликација, која се разликују по принципу развоја и извршавања. То су:

- матичне (енг. *native*)
- веб
- хибридне

Матичне апликације пружају најбоље карактеристике и најбоље свеукупно мобилно искуство. Ово постижу захваљујући томе што су развијане за одређену платформу. Из перспективе развоја апликација, користе се развојни алати, програмски језици и програмске библиотеке које платформа подржава. За *Android* то су *Eclipse* [3] или *Android Studio* [4] и Јава [5], док су за *iOS* то *Xcode* [6] и *Objective-C* [7] или *Swift* [8]. Из перспективе крајњег корисника, постоји могућност проналажења апликација у продавници мобилних апликација. Када се у продавници појави нова верзија апликације, матична апликација обавештава корисника. Корисник затим има могућност да надогради апликацију на најновију верзију. Ово је једна од мана овог типа апликација јер зависи од корисника да ли ће имати ажурирану и побољшану верзију апликације. Још једна негативна одлика овог типа мобилних апликација је да развој за више платформи захтева поновно писање апликације на програмском језику специфичном за ту платформу, што значајно повећава трошкове развоја.

Веб мобилне апликације су веб апликације дизајниране да раде на малим екранима. Као такве, веб апликације су способне да раде на различитим системима без посебног прилагођавања и могу их отворити било који модерни претраживачи. Предност ових апликација је лакша дистрибуција и подршка него код матичних апликација. Отпремањем нове верзије апликације за резултат има ефекат да сви корисници виде нове измене. Оно што је негативна страна овог типа мобилних апликација је да немају приступ матичним функцијама мобилних

уређаја (камера, сензори, итд.). Такође, корисници неће имати конзистентан „*look and feel*“ са осталим апликацијама система, као што је то случај код матичних апликација. Највећа мана овог типа мобилних апликација је да захтевају интернет конекцију како би корисник имао приступ апликацији.

Хибридне апликације комбинују најбоље особине матичних и веб мобилних апликација. Ово се постиже двослојном архитектуром апликације. За имплементацију пословне логике се користе постојеће веб технологије попут *HTML-a*, *CSS-a* и *JavaScript-a*. То је унутрашњи слој. Спољашњи слој је „танак“ матични контејнер који омогућава приступ матичним функцијама платформе попут камере, акцелерометра, локалног складишта података итд. Захваљујући контејнеру, хибридне апликације је такође могуће инсталирати коришћењем продавница мобилних апликација. Велика предност хибридних апликација у односу на матичне је да се једном написан код може извршавати на више различитих платформи. Тиме се значајно смањују трошкови израду апликације. Свака платформа у свом развојном оквиру (енг. *software development kit*) има подршку за веб компоненту као део корисничког интерфејса. Ова веб компонента приказује *HTML* садржај стране хибридне апликације, а матични контејнер је задужен, између осталог, за учитавање ових страница у веб контролу. Како би се омогућио лакши развој хибридних апликација, настали су развојни оквири који садрже скуп компонети корисничког интерфејса и метода за приступ и манипулацију матичним функцијама мобилних уређаја.

1.2 Оквири за развој хибридних апликација

У последњих неколико година јавља се велико интересовање за хибридне оквире за развој мобилних апликација. Главни разлог је платформска независност која смањује трошкове израде мобилне апликације за више платформи. Други разлог је лакши почетак развоја мобилних апликација употребом познатих веб технологија. Тиме се омогућава коришћење постојећих *JavaScript* библиотека и других веб оквира. Неки од популарних оквира су:

- *Ionic* оквир [9]
- *Appcelerator Titanium* [10]
- *Sencha Touch* [11]
- *Framework 7* [12]
- *Kendo UI* [13]
- *Onsen UI* [14]

Одлика *Appcelerator Titanium* је да *JavaScript* логику претвара у матичан код изабране платформе и коначан резултат даје матичну апликацију. Овим се знатно побољшавају перформансе саме апликације. Међутим, мана овог оквира је

смањена флексибилност јер је за сложен интерфејс потребан посебан код за сваку платформу.

Sencha Touch је *JavaScript* оквир за развој мобилних апликација. Оквир чине скуп компоненти корисничког интерфејса и део за приступ матичним функцијама платформе. Велика предност је што поседује посебан пакет компоненти за визуализацију података (графикони, таблеле, итд.). Велика мана овог оквира је то да није бесплатан за употребу.

Оквири *Framework 7*, *Kendo UI*, *Onsen UI* чине скуп компоненти корисничког интерфејса. Главна мана им је да не садрже део за приступ матичним функцијама платформе. То значи да је потребан додатан напор за његову интеграцију у апликацији. *Framework 7* и *Onsen UI* се слободно могу користити, док *Kendo UI* није бесплатан за употребу.

Један од најпопуларнијих оквира за развој хибридних апликација је *Ionic*. Поседује добру документацију и велику заједницу. Постоји посебан форум [15] на коме је могуће потражити помоћ од других чланова заједнице, како би се решио неки проблем или сазнalo нешто више о *Ionic* оквиру. Укључује велики број компоненти корисничког интерфејса које је лако прилагодити жељеном корисничком интерфејсу. Интегрисан је део за приступ матичним функцијама мобилних уређаја, што није случај код *Framework 7*, *Kendo UI*, *Onsen UI* оквира. *Ionic* је потпуно бесплатан за коришћење за разлику од *Sencha Touch* и *Kendo UI* оквира. Због ових карактеристика, *Ionic* оквир је изабран за имплементацију мобилне апликације описане у овом раду.

Рад се састоји од седам поглавља.

Прво поглавље је уводно. У њему ће бити дат преглед типова мобилних апликација, њихових карактеристика и постојећих оквира за њихов развој.

Друго поглавље даје преглед и карактеристике *Ionic* оквира. Описан је сам почетак рада, инсталација оквира и преглед најважнијих компоненти корисничког интерфејса. Такође, дат је опис додатних технологија које овај оквир користи као што су *Cordova* [16] и *AngularJS* [17].

Треће поглавље описује спецификацију захтева апликације која је имплементирана користећи *Ionic* оквир.

Четврто поглавље приказује спецификацију дизајна. Приложени су дијаграм класа, дијаграм распореда и дијаграм секвенци.

Пето поглавље описује имплементацију мобилне апликације помоћу *Ionic* оквира.

Шесто поглавље демонстрира имплементацију мобилне апликације.

Седмо поглавље анализира постигнуте резултате, даје закључке и правце даљег развоја апликације.

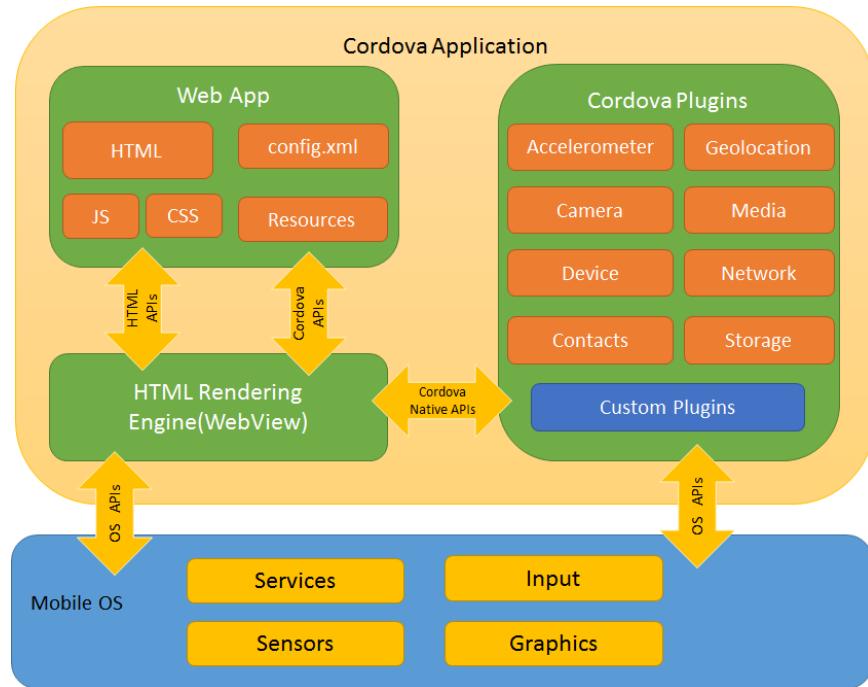
2. IONIC

Ionic је оквир за развој хибридних мобилних апликација. Коришћењем стандардних веб технологија попут *HTML*-а, *CSS*-а и *JavaScript*-а, *Ionic* омогућава развој мобилних апликација за више мобилних платформи. Апликација се извршава унутар контејнера који је специфичан за сваку платформу. Да би мобилна апликација била хибридна, мора имати две одлике. Прва је да садржи матични контејнер који омогућава приступ матичним функцијама мобилног уређаја. Матични контејнер који користи *Ionic* назива се *Cordova*. Друга је да садржи компоненте које служе за креирање корисничког интерфејса и имплементацију пословне логике. У ту сврху *Ionic* користи *AngularJS* оквир.

2.1 Cordova

Cordova је матични контејнер хибридних мобилних апликација. Употребом *JavaScript*-а омогућава приступ матичним функцијама платформе. Апликација се извршава унутар контејнера специфичног за сваку платформу [16].

Архитектуру мобилне апликације развијене помоћу *Cordova* оквира чине неколико компоненти, које чине архитектуру апликације на највишем нивоу апстракције (слика 2.1) [16].



Слика 2.1 - Архитектура *Cordova* апликације [16]

Као што се може видети, архитектуру чине три главне компоненте. То су веб апликација, *WebView* [16] и приклучци.

Веб апликацију чини пословна логика написана у *JavaScript*-у, кориснички интерфејс дефинисан у *HTML*-у и *CSS*-у и ресурси (попут слика и осталих медија фајлова).

Компонента *Cordova* апликације задужена за приказ и извршавање компоненте веб апликације назива се *WebView*.

Cordova прикључци су модуларни. Сваки прикључак омогућава приступ једној матичној функцији платформе. Могуће је имплементирати сопствене прикључке.

Mobile OS компонента представља мобилну платформу чије функције *Cordova* апликација позива. Садржи функције платформе као што су разни сервиси, сензори, камера, итд.

2.1.1 Cordova прикључци

Прикључак је програмски пакет који омогућава *WebView* компоненти у оквиру којег се апликација рендерује, да комуницира са матичном платформом на којој се извршава. Прикључак обезбеђује приступе функционалностима уређаја и платформе које су обично недоступне апликацијама заснованим на веб технологијама. Прикључак чине један *JavaScript* интерфејс заједно са одговарајућим библиотекама матичног кода за сваку подржану платформу [16].

Све главне функционалности *Cordova* оквира су имплементиране као прикључци. Неки од најчешће коришћених су:

- ***cordova-plugin-geolocation*** – пружа информације о локацији уређаја, као што су географска ширина и дужина;
- ***cordova-plugin-camera*** – омогућава снимање форографија и избор фотографије из системске библиотеке фотографија;
- ***cordova-plugin-file*** – омогућава креирање и читање фајлова који се налазе на уређају;
- ***cordova-plugin-splashscreen*** – приказује и крије уводни екран током покретања апликације;
- ***cordova-plugin-device-motion*** – омогућава приступ акцелерометру уређаја. Акцелометар је сензор покрета који детектује промену убрзања уређаја дуж *X*, *Y* и *Z* осе;
- ***cordova-plugin-device*** – користи се за приступ подацима о хардверу и софтверу уређаја;
- ***cordova-plugin-network-information*** – пружа информације о *WiFi* вези уређаја и вези са интернетом;

2.2 AngularJS

AngularJS је *JavaScript* оквир за развој *single page* веб апликација. Олакшава употребу и контролу напредних опција. Неке од њих су:

- развојености логике апликације, модела података и шаблона за приказ података;
- манипулисања историје претраживача;
- *dependency injection* – креира и повезује објекте и функције;

2.2.1 Основни концепти у AngularJS-у

Овај оквир користи неколико концепата који *AngularJS* чине веома моћним овиром за развој веб апликација. Најважнији од њих су:

- **шаблон** (енг. *template*) – *HTML* странице са додатним ознакама;
- **директива** (енг. *directive*) – омогућава проширење синтаксе *HTML*-а са новим елементима и атрибутима;
- **модел** (енг. *model*) – подаци који су се обрађују;
- **контролер** (енг. *controller*) – пословна логика иза погледа;
- **опсег** (енг. *scope*) – контекст у коме је модел смештен;
- **израз** - приступ променљивама и методама из опсега;
- **поглед** (енг. *view*) – *HTML* контејнер који приказује апликацију;
- **повезивање података** (енг. *data binding*) – синхронизација података између модела и *view*-а;
- **модул** (енг. *module*) – контејнер за различите делове апликације као што су контролери, директиве, сервиси, филтери, итд.;
- **сервис** (енг. *service*) – пословна логика независна од погледа;
- **филтер** (енг. *filter*) – дефинише формат за приказ података;

2.2.1.1 Директива

Директива представља начин за креирање нових ставки *HTML*-а који имају сопствено понашање. Нове ставке могу бити атрибути елемената или нови елементи. Потребно их је дефинисати програмски наводећи назив, врсту директиве („E” за елемент, а „A” за атрибут), шаблон директиве и опционо контролер. Листинг 2.1 приказује пример дефинисања новог елемента.

```
angular.module('App', [])
```

```

.directive('mojElement', function() {
  return {
    restrict: 'E',
    template: '<a href="http://google.com">Google</a>',
    controller: function($scope) {
      ...
    }
  }
}) ;

```

Листинг 2.1 - Пример употребе директиве

Нови елемент има назив 'mojElement', а представља линк ка *Google* страници. Да би се користио нови елемент, потребно је у делу за кориснички интерфејс написати следећи део кода:

```
<moj-element></moj-element>
```

Као резултат рендеровања странице, приказаће се линк са називом „*Google*“ дефинисан у делу за кориснички интерфејс.

2.2.1.2 Контролер

Контролери постоје да би омогућили имплементацију пословне логике над моделом на клијентској страни. У основи садрже скуп метода које описују поведење опсега апликације. Контролер је прво потребно дефинисати, а затим га је потребно регистровати код погледа чију пословну логику имплементира. За идентификацију контролера користи се јединствено име које се при регистровању прослеђује директиви под називом *ng-controller*. Листинг 2.2 даје пример дефинисања и регистрања контролера. Први део листинга приказује дефинисање контролера под називом 'myCtrl'. Други део је пример регистрације 'myCtrl' контролера код дела погледа чију пословну логику имплементира, наводећи назив контролера за вредност *ng-controller* атрибута.

```

app.controller('myCtrl', function($scope) {
  ...
});

<div ng-app="myApp" ng-controller="myCtrl">
  ...
</div>

```

Листинг 2.2 - Дефинисање и регистрање контролера

2.2.1.3 Опсег

Ово је најбитнији део *AngularJS* апликације. Опсег повезује контролер и поглед. Овом контексту приступамо помоћу *\$scope* објекта. У *\$scope* објекту налази се пословна логика апликације, сачињена од метода контролера и модела

погледа. Због ове везе, `$scope` објекат ће изменити модел ако се подаци у погледу промене. Исто тако, поглед ће се освежити ако се модел промени. Листинг 2.3 показује пример употребе `$scope` објекта. У контролеру су дефинисане променљива `ime` и функција `promeniIme` који припадају `$scope` објекту. Кликом на дугме „Promeni me“, извршава се функција `promeniIme` која мења вредност променљиве `ime` у „Marko“.

```
<div ng-app="myApp" ng-controller="myCtrl">
  <h1>{{ime}}</h1>
  <button ngClick="promeniIme()">Promeni ime</button>
</div>

<script>
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {
  $scope.ime = "Pera";
  $scope.promeniIme = function() {
    $scope.ime = "Marko";
  }
});
</script>
```

Листинг 2.3 - Употреба `$scope` објекта

2.2.1.4 Израз

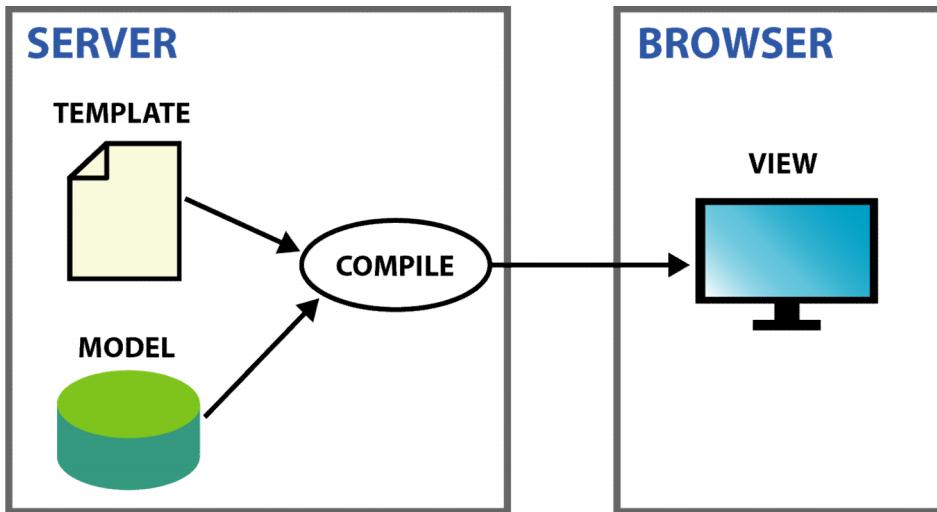
Израз представља део погледа који рендерије динамичке податке на станици. Израз је доступан само у опсегу у којем је дефинисан. То значи да је могуће позвати само оне променљиве дефинисане унутар тог опсега. Листинг 2.4 приказује пример употребе израза. У контролеру је дефинисана променљива `naslov`. Да би се приказала вредност променљиве, потребно је унутар нотације `{{ }}` ставити назив променљиве. Када се страница са овим изразом рендерије, на место променљиве `naslov` приказају се вредност „Naslov 1”.

```
<script>
app.controller('myCtrl', function($scope) {
  $scope.naslov = "Naslov 1";
})
</script>
<div>
  <h1>{{ naslov }}</h1>
</div>
```

Листинг 2.4 - Употреба израза

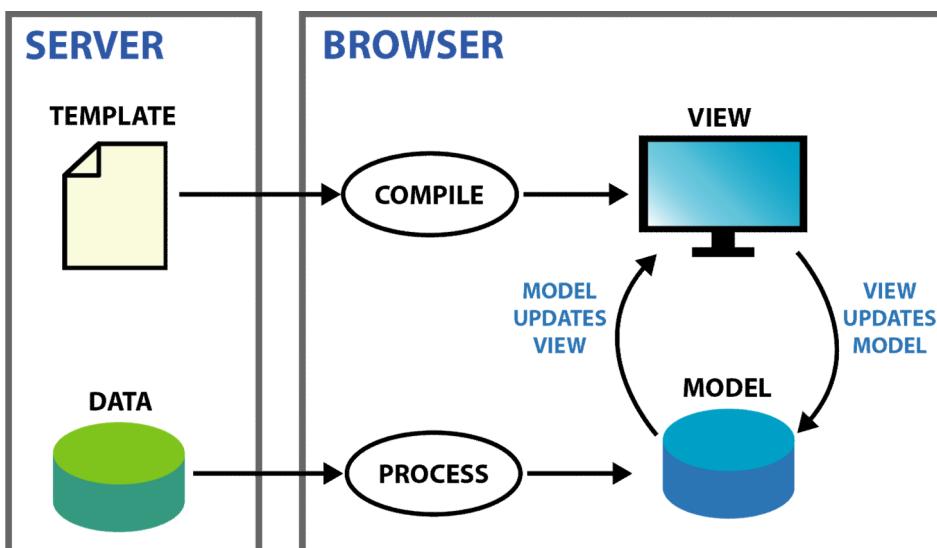
2.2.1.5 Повезивање података у AngularJS-у

Класични оквири за развој веб апликација користе контролер који прикупља податке, податке убацује у шаблон генеришући *HTML* документ и тај документ шаље клијенту. Овај начин повезивања података назива се једноструко спрегнута веза (слика 2.1). Приказани подаци су рефлексија стања модела података у тренутку рендеровања шаблона за приказ [17].



Слика 2.1 - Једноструко спрегнута веза

Модерни веб оквири који се користе на клијентској страни нуде другачији начин повезивања података. Шаблони за приказ и подаци се шаљу независно ка претраживачу. Претраживач затим претвара шаблон у поглед, а податке у модел. На овај начин се ствара двострука веза између погледа и модела. Свака промена на моделу рефлектује се на погледу и обратно, свака промена на погледу изазвана корисничком акцијом резултује променом на моделу. Овај вид повезивања података назива се двоструко спрегнута веза (слика 2.2) [17].



Слика 2.2 - Двоструко спрегнута веза

2.2.1.6 Филтер

Филтер дефинише начин форматирања података за приказ. Постоје предефинисани филтери које је могуће користити, а могуће је дефинисати и сопствене филтере. Користи се унутар израза. Неки од најчешће коришћених филтера:

- *currency* – приказивање валуте;
- *date* – филтер за форматирање датума;
- *lowercase* – претвара израз у мала слова;
- *uppercase* – претвара израз у велика слова;

Листинг 2.5 приказује употребу филтера за форматирање датума. Постоји неколико додатних опција за форматирање датума. Дат је пример додатних опција и приказ њихових резултата.

```
 {{ danasnji_datum | date:'medium' }} <!-- Aug 09, 2013 12:09:02 PM -->
 {{ danasnji_datum | date:'short' }} <!-- 8/9/13 12:09 PM -->
 {{ danasnji_datum | date:'longDate' }} <!-- August 09, 2013 -->
 {{ danasnji_datum | date:'mediumDate' }} <!-- Aug 09, 2013 -->
 {{ danasnji_datum | date:'shortDate' }} <!-- 8/9/13 -->
 {{ danasnji_datum | date:'mediumTime' }} <!-- 12:09:02 PM -->
 {{ danasnji_datum | date:'shortTime' }} <!-- 12:09 PM -->
```

Листинг 2.5 - Употреба *date* филтера

2.3 Основе Ionic оквира

У овом поглављу биће описан поступак инсталације *Ionic* оквира, генерирање новог пројекта и структура пројекта. Такође, биће објашњени најважнији концепти за имплементацију корисничког интерфејса, као и употреба *Cordova* приклучака за приступ матичним функцијама уређаја. На крају је приказано како се апликација покреће на мобилном уређају и на претраживачу.

2.3.1 Почетак рада са Ionic оквиром

Да би се могао користити *Ionic* оквир, потребно га је инсталирати. Као предуслов за коришћење овог оквира потребно је инсталирати *Node.js* [18]. Са његовом инсталацијом долази и пакет менаџер звани *npm* (енг. *Node package manager*) [19]. Пакет је у основи обичан директоријум који садржи код који се може поново употребити и посебан фајл *package.json*. У овом фајлу се налазе метаподаци о пакету као што су назив, верзија, итд. Употреба *npm*-а је погодна када постоји потреба за библиотекама у неком пројекту.

Још један предуслов који мора бити испуњен да би се могао користити *Ionic* оквир је инсталiran *SDK* (енг. *software development kit*) за циљну мобилну платформу. *SDK* чини скуп развојних алата, програмских библиотека и

документације. *Ionic* тренутно подржава два доминантне мобилне платформе, *Android* и *iOS*, потребно је инсталирати *SDK* за ове платформе. За потребе овог рада користиће се *Android*.

Када су сви предуслови задовољени, може се инсталирати *Ionic*. За инсталацију користи се *npm*. Потребно је покренути конзолу и извршити команду:

```
$ npm install -g cordova ionic
```

Ова команда ће инсталирати потребне алате за рад са *Cordova*-ом и *Ionic*-ом преко конзоле.

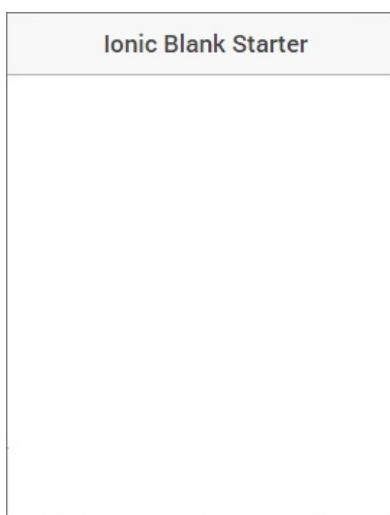
2.3.2 Генерисање новог пројекта

Како би имплементирали нову апликацију, прво се мора генерисати нови *Ionic* пројекат. За то ће нам послужити раније инсталирани алати, чије команде позивамо из конзоле. Потребно је извршити команду:

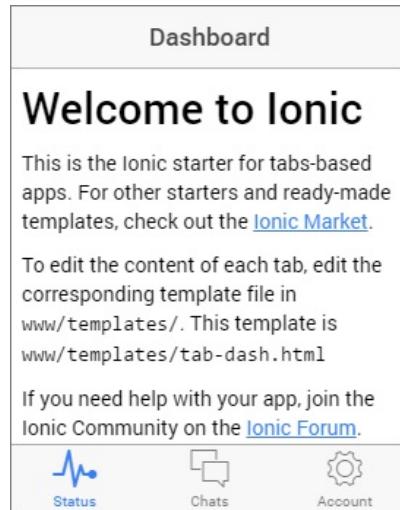
```
$ ionic start <putanja> [naziv šablona]
```

Резултат ове команде биће нови пројекат. Као параметри се прослеђују путања са називом новом пројекта и врста шаблона. *Ionic* пружа могућност креирања предефинисаних типова интерфејса нове апликације. Могуће је изабрати једну од три опције:

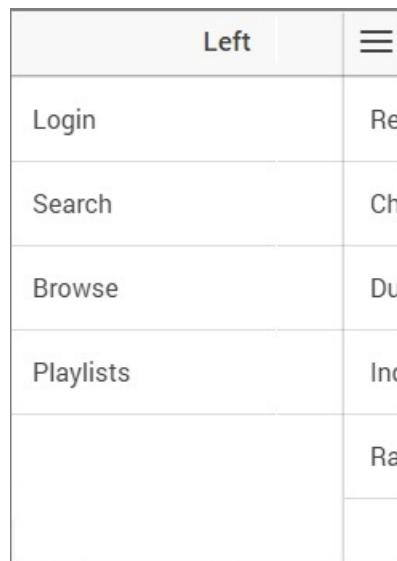
- **blank** – празан интерфејс (слика 2.3);
- **tabs** – интерфејс апликације са картицама (слика 2.4);
- **sidemenu** – интерфејс апликације са фиоком за навигацију (слика 2.5);



Слика 2.3 - Апликација са празним интерфејсом



Слика 2.4 - Апликација са картицама

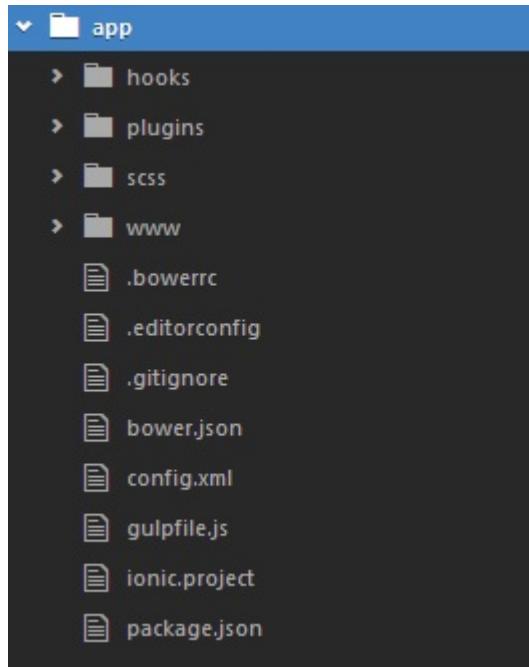


Слика 2.5 - Апликација са фиоком за навигацију

Постоји могућност креирања и сопствених шаблона. Потребно је направити репозиторијум на *Github*-у са називом `ionic-starter-<naziv šablonu>`.

2.3.3 Структура апликације

Након генерисања новог пројекта помоћу конзолне команде, направљена је иницијална структура пројекта која се неће много мењати у току даљег рада на пројекту (слика 2.6). Када се додају функције специфичне за платформу, креираће се додатни фолдер под називом *platforms*.



Слика 2.6 - Структура *Ionic* апликације

Иницијална структура се састоји од:

- **hooks** – садржи прилагођене *Cordova* пречице за извршавање специфичних команди;
- **plugins** – садржи инсталиране *Cordova* приклучке;
- **scss** – садржи стилове апликације које је могуће модификовати;
- **www** – садржи пословну логику апликације, структуру корисничког интерфејса и стилове;
- **.bowerrc** – садржи вредност путање где ће бити инсталиране додатне библиотеке, обично има вредност *www/lib*;
- **bower.json** – садржи листу додатних библиотека које су потребне;
- **config.xml** – поред информација о апликацији и атору, садржи и додатна подешавања пројекта која је могуће променити;
- **ionic.project** – дефинише назив и јединствени идентификатор апликације;

2.3.4 Главне компоненте корисничког интерфејса

Једна од главних предности *Ionic* оквира је велики скуп компоненти за креирање корисничког интерфејса. Захваљујући овим компонентама, могуће је веома брзо направити жељени кориснички интерфејс. Компоненте је такође могуће комбиновати у циљу креирања сложенијих компоненти (употребом *AngularJS-a*). Једна од предности развоја корисничког интерфејса употребом веб технологија је могућност прилагођавања садржаја различитим екранима. Наравно,

ова могућност постоји и код развоја матичних апликација али је мање флексибилна. У наставку ће бити направљен преглед најважнијих компоненти *Ionic* оквира.

2.3.4.1 Button компонента

Ова компонента је једна од есенцијалних компоненти корисничког интерфејса. У оквиру *Ionic*-а, омогућен је велики број подешавања везаних за ову компоненту. Могуће је променити боју, величину, ивицу, итд. једноставном променом CSS класе. Листинг 2.3 приказује примере употребе ове компоненте, а слика 2.7 њен рендерован изглед.

```
<button class="button button-block button-positive">  
    button-positive  
</button>  
  
<button class="button button-full button-positive">  
    button-positive  
</button>  
  
<button class="button button-outline button-positive">  
    button-positive  
</button>
```

Листинг 2.3 - Употреба *button* компоненте



Слика 2.7 - Приказ *button* компоненти

2.3.4.2 Toggle компонента

У суштини, ова компонента је по природи *HTML checkbox*. Једина разлика је што изгледа другачије и лакше ју је користити на уређајима који су осетљиви на додир. Такође, могуће је променити боју компоненте у зависности од њеног стања (укључено, искључено). Листинг 2.4 приказује изворни код ове компоненте, а слика 2.8 њен рендерован изглед.

```
<label class="toggle">  
    <input type="checkbox">
```

```

<div class="track">
  <div class="handle"></div>
</div>
</label>

```

Листинг 2.4 - Употреба *toggle* компоненте



Слика 2.8 - *Toggle* компонета у „on“ и „off“ стању

2.3.4.3 Range компонента

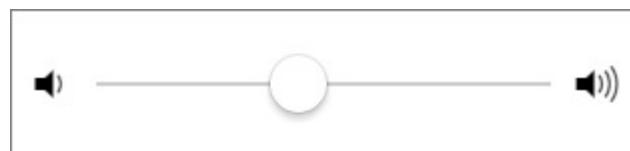
Компонента која је уведена *HTML5* стандардом. Она у ствари представља слайдер који може имати вредност у опсегу од минималне до максималне. Листинг 2.5 приказује изворни код ове компоненте, а слика 2.9 њен рендерован изглед.

```

<div class="item range">
  <i class="icon ion-volume-low"></i>
  <input type="range" name="volume" min="0" max="30" value="15">
  <i class="icon ion-volume-high"></i>
</div>

```

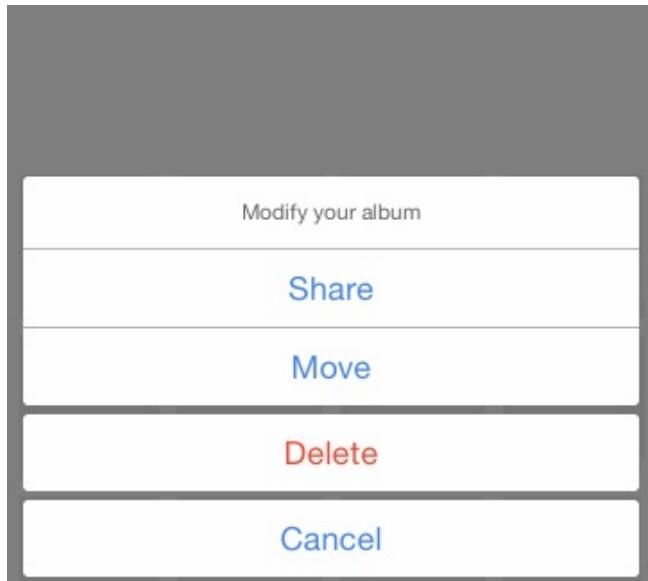
Листинг 2.5 - Употреба *range* компоненте



Слика 2.9 - Приказ *range* компоненте

2.3.4.4 Action sheet компонента

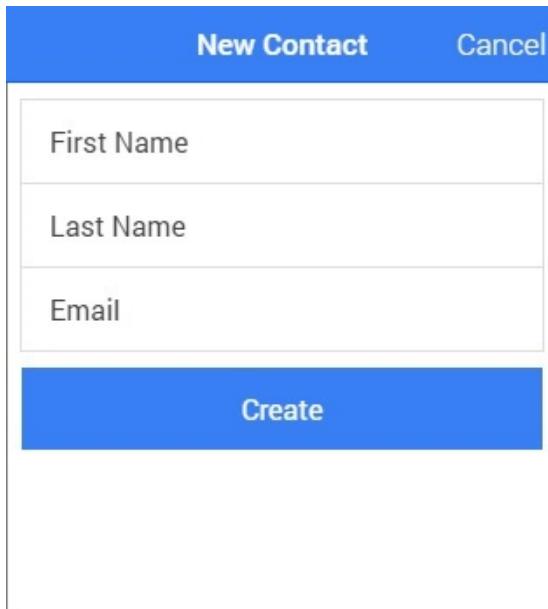
Компонента која представља панел са ставкама које је могуће активирати помоћу подржаних гестова или догађаја (слика 2.10). Дефинише се у *AngularJS* коду.



Слика 2.10 - Приказ *action sheet* компоненте

2.3.4.5 Modal компонента

Ова компонента је модални дијалог (дијалог који узима фокус). Обично се користи за потврду извршавања акције или промену података. Као и *action sheet* компонета, и ова се дефинише програмски (слика 2.11).



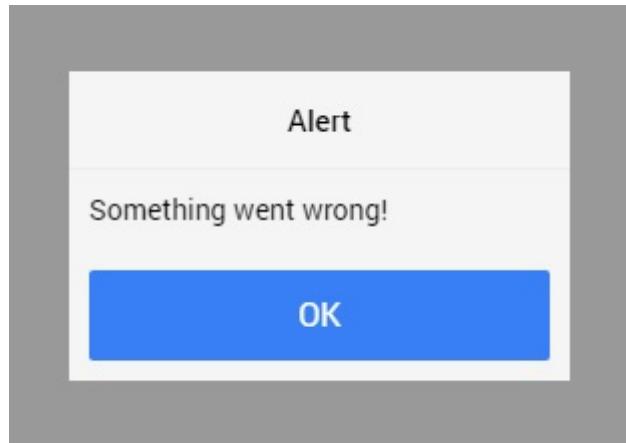
Слика 2.11 - Приказ *modal* компоненте

2.3.4.6 Pop up компонента

Ова компонента је слична *modal* компоненти, али не заузима цео екран. То је класични дијалог који тражи потврду корисника за наставак радње (слика 2.12). Дефинише се програмски. Постоје три типа дијалога (у зависности од методе која се позива при креирању):

- *show* – дијалог коме је могуће поставити произвољан садржај;

- ***confirm*** – приказује поруку уз могућност прихватања или одбијања акције;
- ***alert*** – приказује поруку обавештења;



Слика 2.12 - Приказ *popup* компоненте

2.3.5 Инсталација Cordova прикључака

Инсталација прикључака је једноставна. Потребно је извршити следећу команду:

```
$ ionic plugin add <naziv priključka>
```

Овом командом се инсталира нови прикључак (додаје се у *plugins* фолдер). Након тога, у апликацији је могуће користити прикључак за приступ матичним функцијама уређаја.

2.3.6 Покретање апликације

Као што је већ споменуто, *Ionic* је овир за креирање хибридних мобилних апликација. Ова карактеристика омогућава апликацији да се по потреби понаша као мобилна апликација или као веб апликација.

Прво је неопходно додати подршку за жељену платформу. Да би се то постигло потребно је извршити следећу команду:

```
$ ionic platform add <naziv platforme>
```

Ова команда ће проширити структуру почетног пројекта и додати нови директоријум под називом *platforms*. У пројекат ће додати неопходне зависности везане за специфичну платформу у циљу превођења и извршавања апликације. Тренутно подржане платформе су *Android* и *iOS*, тако да се као параметар може проследити једна од две вредности - *android* или *ios*. Након што је одређена мобилна платформа на којој ће се извршавати апликација, апликацију могуће је извршити.

Често се на почетку развоја мобилне апликацији прво имплементира кориснички интерфејс. Да би овај поступак био бржи и једноставнији, *Ionic* апликацију је могуће покренути у оквиру претраживача. Ово је могуће зато што

се за развој користе веб технологије. Да би се апликација извршавала у облику веб апликације потребно ју је покренути следећом командом:

```
$ ionic serve
```

Резултат команде покреће локални сервер и отвара се нови прозор претраживача. Сервер прати промене над фајловима у којима је дефинисан кориснички интерфејс. Након чувања измена аутоматски ће се освежити приказ екрана.

Примарни облик *Ionic* апликације је мобилна апликација. Постоје два начина за извршавање апликације као мобилне апликације. Први начин подразумева извршавање апликације на симулираном уређају. Да би се то постигло потребно је извршити команду:

```
$ ionic emulate <naziv platforme>
```

Апликација ће се покренути унутар емулатора који долази заједно са инсталацијом развојних алата (у овом случају *Android SDK*). Велика предност емулатора је могућност подешавање разних резолуција екрана и других конфигурација уређаја. Велика мана емулатора је немогућност емулације сензора и ниже перформансе у односу на физичке уређаје.

Како би се апликација покренула на правом уређају и могла да искористи све могућности које исти пружа, неопходно је извршити команду:

```
$ ionic run <naziv platforme>
```

Као предуслов, потребно је повезати уређај са рачунаром. Ако то није урађено, покренуће се емулатор.

2.3.7 Експортовање апликације за мобилну продавницу

Када је развој апликације готов, потребно ју је објавити. Захваљујући томе што хибридна апликација има особине матичне апликације, могуће ју је отпремити у мобилну продавницу. Као резултат, апликација ће бити доступна свим корисницима платформе за коју је направљена. Да би се направио исталациони фајл, потребно је покренути команду:

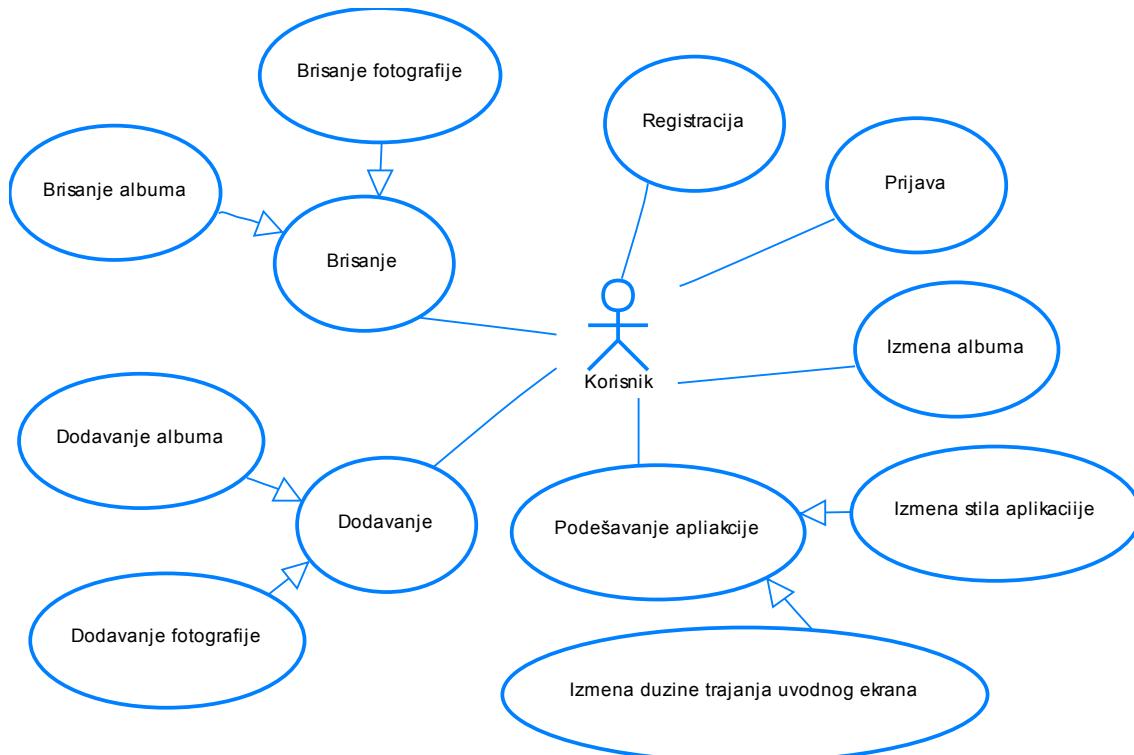
```
$ ionic build <naziv platforme> --release
```

Резултат ове команде биће генерисани инсталациони фајл. У случају *Android-a*, фајл ће имати *.apk* екstenзију и налазиће се на релативној путањи пројекта *platforms/android/build/outputs/apk*. Све што остаје да се уради је да се апликација објави пратећи правила за матичне апликације специфичне за дату платформу. Након тог процеса, корисници ће моћи да је инсталирају и користе на својим мобилним уређајима.

3. СПЕЦИФИКАЦИЈА ЗАХТЕВА

Као пример употребе *Ionic* оквира имплементирана је фото-албум мобилна апликација. Апликација кориснику омогућава прављење фотографија и њихово организовање у виду албума.

За спецификацију функционалних захтева коришћени су случајеви коришћења (енг. *use cases*), помоћу којих ће бити описане функције апликације (тј. акције које корисник може да изврши). Слика 3.1 приказује дијаграм случајева коришћења корисника апликације.



Слика 3.1 - Дијаграм случајева коришћења корисника апликације

Табела 3.1 приказује случај коришћења регистрације новог корисника.

Случај коришћења	Регистрација новог корисника
Главни учесници	Корисник
Предуслови	Корисник није регистрован.
Главни сценарио успеха	
1. Корисник уноси своје пуно име. 2. Корисник уноси <i>e-mail</i> адресу. 3. Корисник уноси шифру. 4. Корисник притиска дугме за регистрацију. 5. Систем креира новог корисника.	

6. Систем редиректује корисника на страницу за пријаву.
Проширења
1а. Корисник није унео пуно име. <ul style="list-style-type: none"> 1a1. Систем обавештава корисника да поље не сме бити празно. 1a2. Корисник уноси пуно име.
2а. Корисник уписује <i>e-mail</i> адресу погрешног формата. <ul style="list-style-type: none"> 2a1. Систем обавештава корисника о грешци. 2a2. Корисник уноси валидну <i>e-mail</i> адресу.
3а. Корисник уноси шифру дужине мању од осам карактера. <ul style="list-style-type: none"> 3a1. Систем обавештава корисника о грешци. 3a2. Корисник уноси шифру дужине осам или више карактера.

Табела 3.1 - Случај коришћења регистрације новог корисника

Табела 3.2 приказује случај коришћења пријаве корисника.

Случај коришћења	<i>Пријава корисника</i>
Главни учесници	Корисник
Предуслови	Корисник је регистрован.
Главни сценарио успеха	
1. Корисник уноси <i>e-mail</i> адресу. 2. Корисник уноси шифру. 3. Корисник притиска дугме за пријаву. 4. Систем проналази корисника у бази на основу <i>e-mail</i> адресе и шифре. 5. Систем обавештава корисника о успешној пријави. 6. Систем редиректује корисника на почетну страницу.	
Проширења	
1а. Корисник уписује <i>e-mail</i> адресу погрешног формата. <ul style="list-style-type: none"> 1a1. Систем обавештава корисника о грешци. 1a2. Корисник уноси валидну <i>e-mail</i> adresu. 	
2а. Корисник уноси шифру дужине мању од осам карактера. <ul style="list-style-type: none"> 2a1. Систем обавештава корисника о грешци. 2a2. Корисник уноси шифру дужине осам или више карактера. 	
4а. Систем није нашао корисника. <ul style="list-style-type: none"> 4a1. Систем обавештава да корисник не постоји. 4a2. Корисник поново уноси креденцијале. 	

Табела 3.2 - Случај коришћења пријаве корисника

Табела 3.3 приказује случај коришћења креирања новог албума.

Случај коришћења	<i>Креирање новог албума</i>
Главни учесници	Корисник

Предуслови	Корисник је пријављен. Корисник се налази на екрану за приказ листе албума тог корисника.
Главни сценарио успеха	
1. Корисник притиска дугме за додавање новог албума. 2. Корисник уноси назив албума. 3. Корисник уноси опис албума. 4. Корисник притиска дугме за креирање. 5. Систем проверава да ли постоји албум са унетим називом. 6. Систем креира нови албум. 7. Систем обавештава корисника да је албум направљен.	
Проширења	
2a. Корисник није унео назив албума. 2a1. Систем обавештава корисника да поље не сме бити празно. 2a2. Корисник уноси назив. 3a. Корисник није унео опис албума. 3a1. Систем обавештава корисника да поље не сме бити празно. 3a2. Корисник уноси опис. 5a. Систем проналази албум са истим називом. 5a1. Систем обавештава корисника да албум са истим називом већ постоји. 5a2. Корисник уноси нови назив за албум.	

Табела 3.3 - Случај коришћења креирање новог албума

Табела 3.4 приказује случај коришћења промене детаља постојећег албума.

Случај коришћења	Промена детаља постојећег албума
Главни учесници	Корисник
Предуслови	Корисник је пријављен. Корисник се налази на екрану за приказ листе албума тог корисника.
Главни сценарио успеха	
1. Корисник селектује албум који жели да промени. 2. Систем нуди листу опција (међу којима је и акција за промену детаља). 3. Корисник одабира опцију за промену детаља албума. 4. Систем приказује екран за промену детаља албума. 5. Корисник мења назив албума. 6. Корисник мења опис албума. 7. Корисник притиска дугме за чување. 8. Систем проверава да ли постоји албум са унетим називом. 9. Систем чува детаље жељени албум. 10. Систем обавештава корисника да је албум промењен.	
Проширења	

5a. Корисник уноси празан назив албума.
5a1. Систем обавештава корисника да поље не сме бити празно.
5a2. Корисник уноси назив.
6a. Корисник уноси празан опис албума.
6a1. Систем обавештава корисника да поље не сме бити празно.
6a2. Корисник уноси опис.
8a. Систем проналази албум са истим називом.
8a1. Систем обавештава корисника да албум са истим називом већ постоји.
8a2. Корисник уноси нови назив за албум.

Табела 3.4 - Случај коришћења промене детаља албума

Табела 3.5 приказује случај коришћења брисања постојећег албума.

Случај коришћења	<i>Брисање постојећег албума</i>
Главни учесници	Корисник
Предуслови	Корисник је пријављен. Корисник се налази на екрану за приказ листе албума тог корисника.
Главни сценарио успеха	
1. Корисник селектује албум коју жели да избрише. 2. Систем нуди листу опција (међу којима је и акција за брисање). 3. Корисник одабира опцију за брисање албума. 4. Систем брише жељени албум. 5. Систем обавештава корисника да је албум обрисан.	

Табела 3.5 - Случај коришћења брисање албума

Табела 3.6 приказује случај коришћења креирања нове фотографије.

Случај коришћења	<i>Креирање нове фотографије</i>
Главни учесници	Корисник
Предуслови	Корисник је пријављен. Корисник се налази на екрану за приказ детаља албума.
Главни сценарио успеха	
1. Корисник притиска дугме за додавање нове фотографије. 2. Систем активира апликацију Камера. 3. Корисник притиска дугме за креирање нове фотографије. 4. Систем редиректује корисника на екран за ефекте. 5. Корисник примењује жељени ефекат на фотографију. 6. Корисник уноси назив нове фотографије. 7. Корисник притиска дугме за чување фотографије. 8. Систем проверава да ли постоји фотографија са унетим називом. 9. Систем креира нову фотографију.	

10. Систем обавештава корисника да је фотографија направљена.
Проширења
6а. Корисник није унео назив фотографије. 6а1. Систем обавештава корисника да поље не сме бити празно. 6а2. Корисник уноси назив.
8а. Систем проналази фотографију са истим називом. 8а1. Систем обавештава корисника да фотографија са истим називом већ постоји. 8а2. Корисник уноси нови назив за фотографију.

Табела 3.6 - Случај коришћења креирања фотографије

Табела 3.7 приказује случај коришћења брисања фотографије.

Случај коришћења	<i>Брисање постојеће фотографије</i>
Главни учесници	Корисник
Предуслови	Корисник је пријављен. Корисник се налази на екрану за приказ детаља албума.
Главни сценарио успеха	
1. Корисник селектује фотографију коју жели да избрише. 2. Систем нуди листу опција (међу којима је и акција за брисање). 3. Корисник одабира опцију за брисање фотографије. 4. Систем брише жељену фотографију. 5. Систем обавештава корисника да је фотографија обрисана.	

Табела 3.7 - Случај коришћења брисања фотографије

Табела 3.8 приказује случај коришћења промене стила апликације.

Случај коришћења	<i>Промена стила апликације</i>
Главни учесници	Корисник
Предуслови	Корисник је пријављен. Корисник се налази на екрану за подешавања.
Главни сценарио успеха	
1. Корисник бира опцију за промену стила апликације. 2. Корисник бира између понуђених стилова. 3. Корисник притиска дугме за чување стила апликације. 4. Систем чува нови стил апликације. 5. Систем примењује сачувани стил на изглед апликације. 6. Систем редиректује корисника на екран за подешавања.	

Табела 3.8 - Случај коришћења промене стила апликације

Табела 3.9 приказује случај коришћења промене дужине трајања уводног екрана.

Случај коришћења	<i>Промена дужине трајања уводног екрана</i>
Главни учесници	Корисник
Предуслови	Корисник је пријављен. Корисник се налази на екрану за подешавања.
Главни сценарио успеха	
1. Корисник бира опцију за промену дужине трајања уводног екрана. 2. Корисник бира између понуђених временских интервала. 3. Корисник притиска дугме за чување дужине трајања уводног екрана. 4. Систем чува дужине трајања уводног екрана. 5. Систем редиректује корисника на екран за подешавања.	

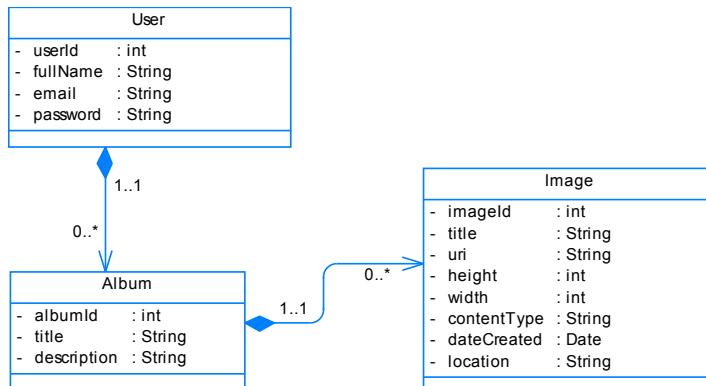
Табела 3.9 - Случај коришћења промене дужине уводног екрана

4. ДИЗАЈН

Спецификација дизајна описана је дијаграмом класа, дијаграмом секвенци и дијаграмом распореда. Дијаграм класа и дијаграм распореда описују статички модел система. Дијаграм секвенце описује динамички модел система.

4.1 Дијаграм класа

У овом делу приказан је дијаграм класа photo-албум апликације, који описује скуп класа и везе између класа (слика 4.1).



Слика 4.1 - Дијаграм класа photo-албум апликације

За photo-албум мобилну апликацију су нам потребне три класе: *User*, *Album* и *Image*.

Класа *User* представља корисника апликације. Јединствени идентификатор ове класе је *userId*. Поред њега, ова класа још садржи:

- *fullName* – представља пуно име корисника;
- *email* – представља *e-mail* корисника;
- *password* – представља тајну шифру корисника;

Један корисник може имати више албума, а не мора имати ниједан. Класа *Album* мора припадати тачно једном кориснику. Јединствени идентификатор је атрибут *albumId*. Ова класа садржи још два атрибута:

- *title* – представља назив албума;
- *description* – представља опис албума;

Албум може имати више фотографија, а не мора имати ни једну. Класа *Image* представља сачувану фотографију корисника и припада тачно једном, одређеном албуму. Поред јединственог идентификатора *imgId*, садржи следеће атрибути:

- *title* – представља назив фотографије;

- *uri* – представља путању на којој је налази фотографија на уређају;
- *height* – представља висину фотографије;
- *width* – представља ширину фотографије;
- *contentType* – представља формат фотографије;
- *dateCreated* – представља датум креирања фотографије;
- *location* - представља локацију фотографије на којој је направљена;

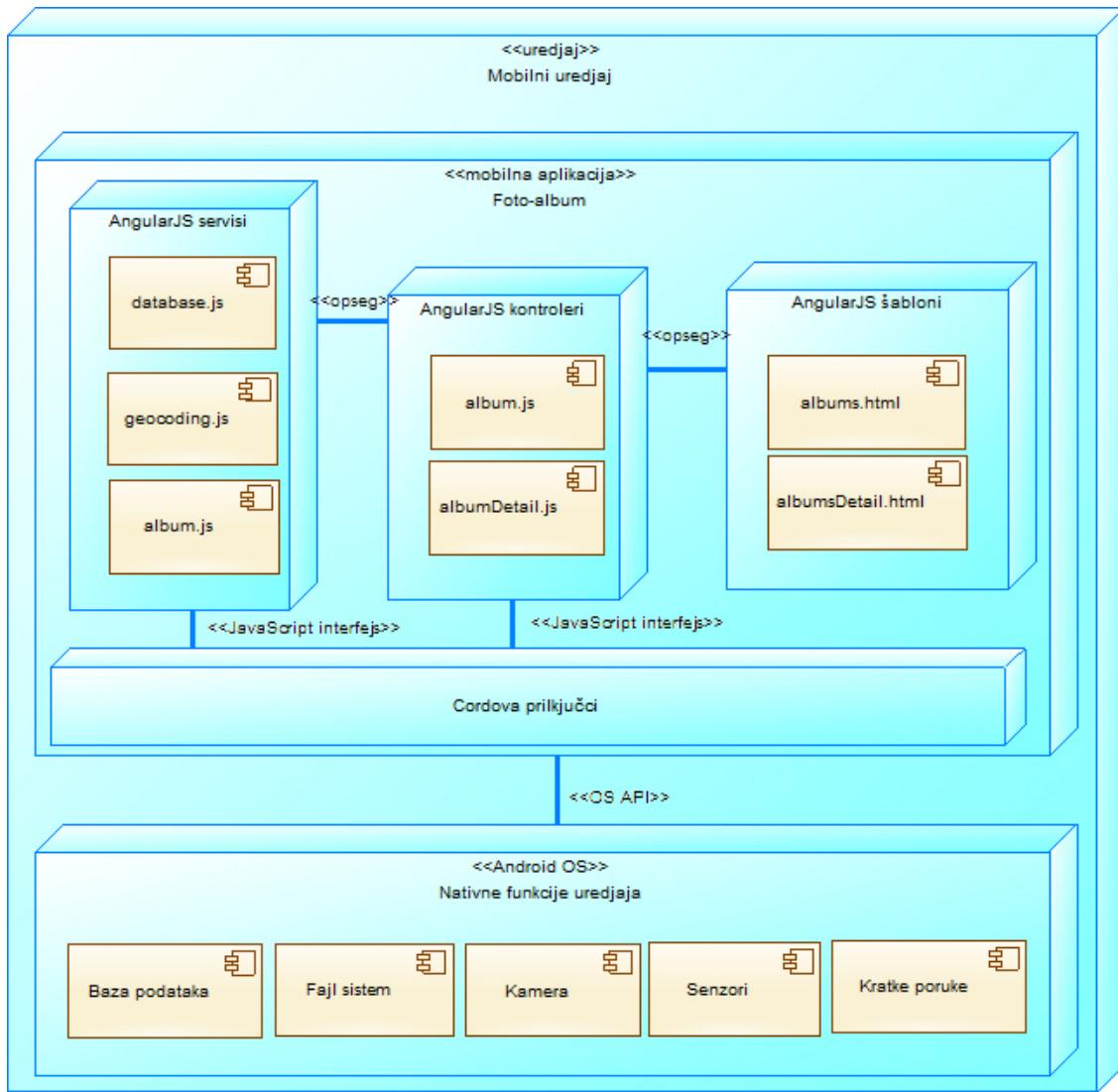
4.2 Дијаграм распореда

Као што је већ споменуто, дијаграм распореда приказује податке о распореду компоненти апликације. Слика 4.2 приказује дијаграма распореда фото-албум апликације.

Главни чвор на дијаграму је мобилни уређај на којем се извршава апликација. Та компонента садржи две компоненте: оперативни систем и мобилну апликацију.

Оперативни систем је компонента коју чине скуп компоненти матичних функција уређаја (база података, фајл систем, камера, сензори, итд.).

Апликација је фото-албум апликација која се извршава на уређају. Састоји се од неколико компоненти. Компонента *Cordova* прикључака чини скуп прикључака који омогућавају приступ матичним функцијама уређаја. Ова компонента је модуларног типа. Прикључци се могу додавати или брисати по потреби. Пословна логика апликације састоји се од сервиса и контролера имплементираних у *AngularJS* оквиру. Преко *JavaScript* интерфејса приступају *Cordova* прикључцима. Садрже код задужен за имплементацију пословне логике фото-албум апликације. Подаци се приказују и преузимају од стране корисника помоћу компоненте која представља шаблоне за приказ података. Са контролером је повезана преко опсега.



Слика 4.2 - Приказ дијаграма распореда

4.3 Дијаграм секвенци

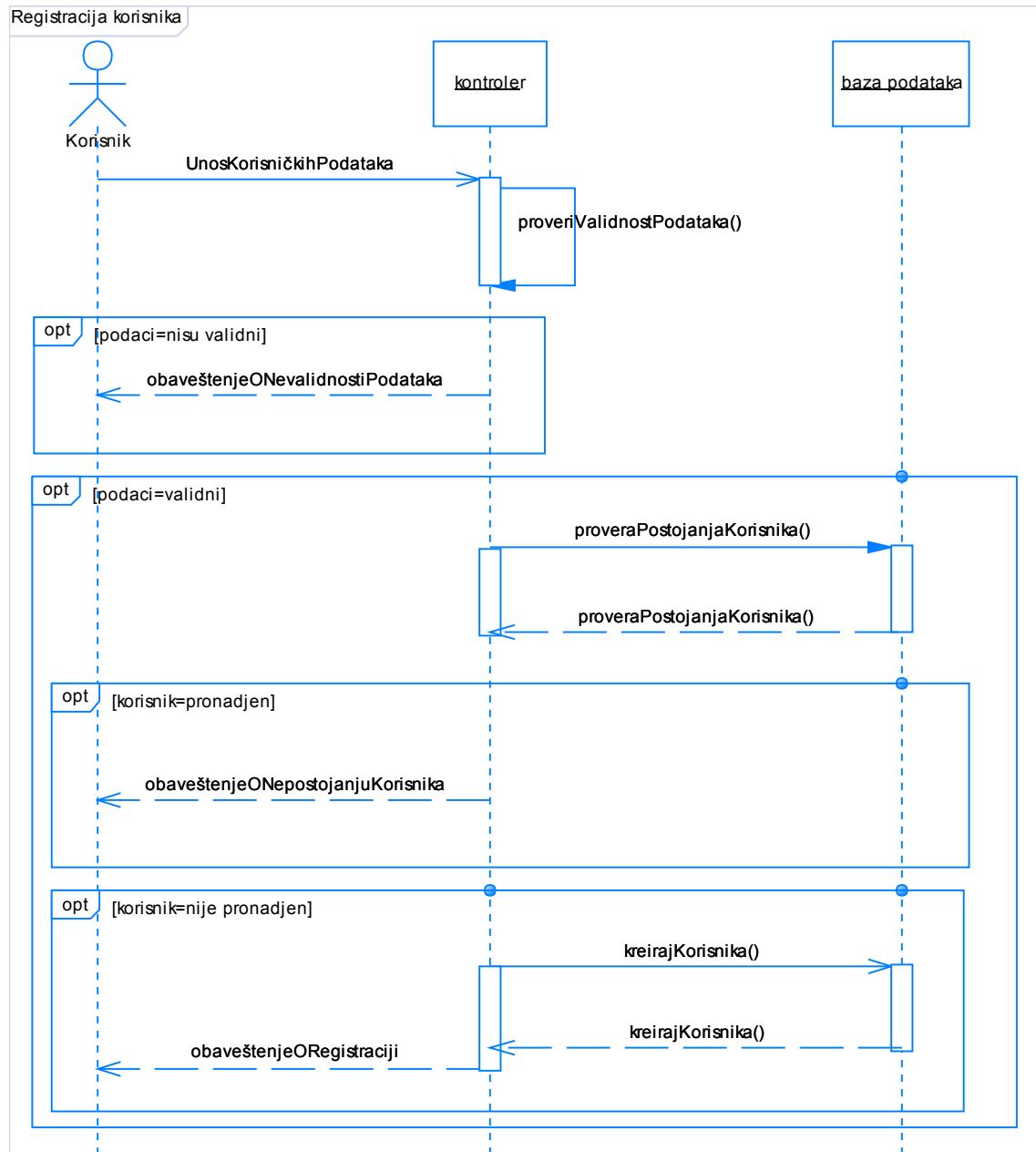
Овај одељак описује сенквенце које имплементирају репрезентативне случајеве коришћења.

4.3.1 Дијаграм секвенце за регистрацију корисника

Дијаграм секвенце за регистрацију корисника (слика 4.3) описује низ активности које су потребне да би се имплементирала ова функционалност.

Корисник иницијално покреће ову активност уносом имена, *e-mail* адресе и шифре. Те податке преузима контролер и позива део за валидацију података. У случају да валидација није задовољена, контролер обавештава корисника поруком о валидности података. Како би се процес регистрације наставио, потребно је да се сви подаци успешно валидирају. У том случају контролер проверава да ли корисник са тим подацима већ постоји у бази. Контролер добија одговор од базе података о постојању корисника. Ако је корисник са тим подацима пронађен,

контролер обавештава корисника о томе. Ако се корисник са унетим подацима не налази у бази, тада контролер позива део за креирање корисника у бази података. Након обављене активности, контролер обавештава корисника о успешној регистрацији корисника.



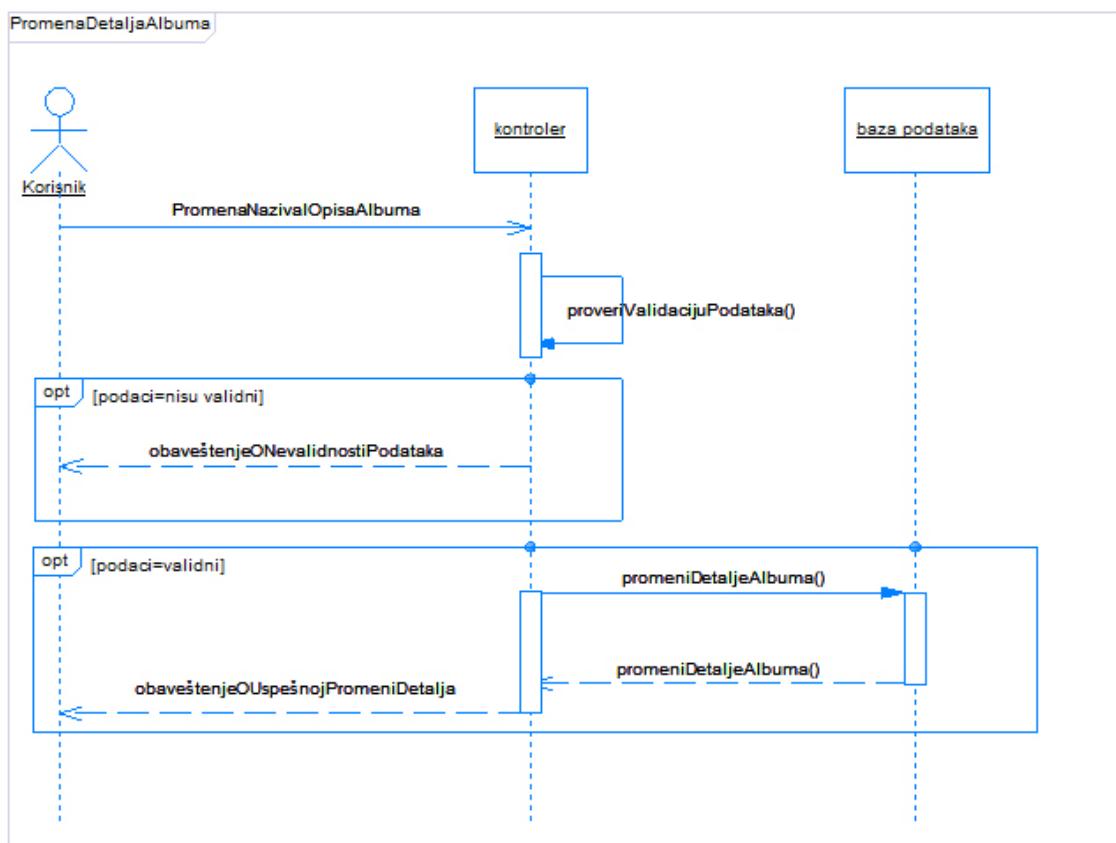
Слика 4.3 - Дијаграм секвенце регистрације корисника

4.3.2 Дијаграм секвенце за промену албума

Дијаграм секвенце за промену албума (слика 4.4) описује низ активности које су потребне да би се имплементирала ова функционалност.

Корисник иницијално покреће ову активност променом назива или описа албума. Подаци се прослеђују контролеру који врши валидацију података. Ако валидација назива или описа није успешна, контролер обавештава корисника о

тome. У случају да је валидација успешна, контролер покреће активност за промену албума у бази података.

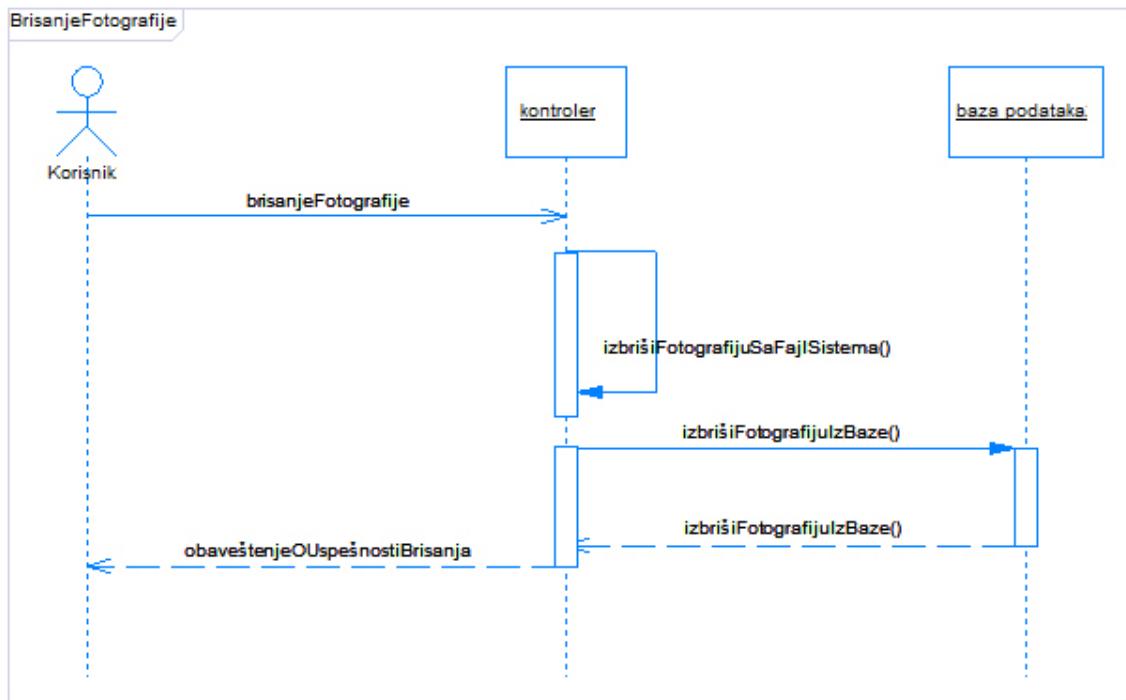


Слика 4.4 - Дијаграм секвенце за промену албума

4.3.3 Дијаграм секвенце за брисање фотографије

Дијаграм секвенце за брисање фотографије (слика 4.5) описује низ активности које су потребне да би се имплементирала ова функционалност.

Корисник иницијално покреће ову активност позивањем акције за брисање фотографије. Након тога, контролер врши брисање фајла фотографије позивом методе за брисање. Брисање фајла се врши на основу путање фајла и његовог назива. Када је брисање фајла завршено, контролер иницира брисање фотографије у бази података. По завршетку ове акције, контролер ће обавестити корисника о успешно извршеној операцији брисања фотографије.



Слика 4.5 - Дијаграм секвенце за брисање фотографије

5. ИМПЛЕМЕНТАЦИЈА

Ово поглавље описује кључне делове имплементације фото-албум мобилне апликације помоћу *Ionic* оквира. Посебно је приказано како апликација приступа матичним функцијама уређаја.

5.1 Почетак рада

Ionic оквир за креирање апликација користи *AngularJS* и *Cordova* оквире. *Cordova* је имплементирана у чистом *JavaScript* језику, док *AngularJS* користи додатне концепте са посебном *JavaScript* синтаксом. Како би се инсталирани прикључци користили у имплементацији, потребно их је прилагодити концептима и синтакси *AngularJS*-а. У ту сврху послужиће посебан *AngularJS* модул назван *ngCordova* [20]. Овај модул садржи скуп *AngularJS* сервиса који имплементирају логику прикључака. Након неговог укључивања у пројекат, потребно је регистровати *ng-cordova.min.js* фајл у *index.html* фајлу (листинг 4.1).

```
<!DOCTYPE html>
<html>
  <head>
    ...
    <script src="lib/ngCordova/dist/ng-cordova.min.js"></script>
    ...
  </head>
  <body>
    <body>
  </html>
```

Листинг 4.1 - Регистровање *ng-cordova.min.js* фајла

Након тога, потребно је регистровати модул како би се сервиси могли користити. То се ради у фајлу *app.js* (листинг 4.2).

```
angular.module('starter', ['ionic', 'ngCordova', ...])
...

```

Листинг 4.2 - Регистровање *ng-cordova.min.js* фајла

5.2 Рад са базом података

Како би сачували информације о кориснику, албумима и фотографијама, потребно је искористи базу података. Да би се ова функционалност могла искористити потребно је инсталирати *Cordova* прикључак за рад са базом. За инсталацију прикључка потребно је извршити следећу команду:

```
$ ionic plugin add cordova-sqlite-storage
```

Након успешне инсталације, могућ је рад са базом података. За комуникацију са базом направљен је посебан *AngularJS* сервис под називом *database.js*. У њему су смештене методе за иницијално креирање потребних табела и остале акције за рад над ентитетима, попут:

- уноса новог корисника, албума и фотографије;
- провере да ли у бази постоји корисник, албум или форографија;
- брисања албума и фотографија;
- претрагу по јединственом идентификатору корисника, албума и фотографија;

Основна функција за извршавање упита над базом назива се *execute*. Као параметре има базу података над којом се упит извршава, сам упит и низ параметара упита. Да би се могле извршавати акције над базом, потребно је укључити сервис *\$cordovaSQLite* модула *ngCordova*. Листинг 4.3 приказује део *database.js* сервиса са примерима метода за рад са базом. Приказана је имплементација методе *getDatabase* која добавља објекат базе позивом *openDB* методе *\$cordovaSQLite* сервиса. За брисање албума, користи се *deleteAlbum* метода која као параметар прима јединствени идентификатор албума. Јединствени идентификатор се прослеђује упиту за брисање у склопу позива методе за извршавање *execute* упита. Упит над базом који претражује да ли фотографија већ постоји са прослеђеним називом имплементира метода *imageExists*. Као параметар прослеђује се назив фотографије, а повратна вредност садржана је у *imageExists* променљивој (може бити *true* или *false*).

```
angular.module('starter.database', [])
.factory('DatabaseService', function ($q, $cordovaSQLite) {
return {
  getDatabase: function() {
    var db = $cordovaSQLite.openDB ('posterionic.db');
    return db;
  },
  ...
  deleteAlbum: function (albumId) {
    var query = "DELETE FROM album WHERE album_id = ?";
    $cordovaSQLite.execute(this.getDatabase(), query, [albumId])
      .then(function(res) {
        console.log("DELETED ID -> " + res.insertId);
      }, function (err) {
        console.error(err);
      });
  },
}
```

```

...
imageExists: function (title) {
    var q = $q.defer();
    var query = "SELECT * FROM image WHERE image_title = ?";
    var imageExists = false;
    $cordovaSQLite.execute(this.getDatabase(), query, [title])
        .then(function(res) {
            if(res.rows.length > 0) {
                imageExists = true;
                q.resolve(imageExists);
            } else {
                console.log("No results found");
                imageExists = false;
                q.resolve(imageExists);
            }
        }, function (err) {
            console.error(err);
            q.resolve(err);
        });
    return q.promise;
}, ...
}

```

Листинг 4.3 - Пример рада са базом

5.3 Унос података

База података манипулише подацима корисника, албума и фотографија. Како би информације доспеле до дела који брине о њиховом уносу или промени, потребно је омогућити кориснику да их унесе. За те потребе користи се посебна *Ionic* компонента звана *modal* компонента. Ова компонента има улогу дијалога која садржи поља за унос података.

Као пример послужиће дијалог за унос имена фотографије који се састоји од шаблона дефинисаним *AngularJS* елементом *ion-modal-view* (листинг 4.4) и логике за приказ имплементиране у *AngularJS*-у (листинг 4.5). Како би се компонента користила, потребно је регистровати сервис *\$ionicModal*. Метода *fromTemplateUrl* се користи као конструктор компоненте. Има као параметре путању до шаблона и објекат који репрезентује опције, као што је *\$scope* објекат, анимација дијалога итд. Дијалог се приказује позивом методе *show*, а скрива позивом методе *hide*.

```

<ion-modal-view>
    <ion-header-bar class="bar-positive">

```

```

<h1 class="title">Add photo</h1>
</ion-header-bar>
<ion-content> ... </ion-content>
</ion-modal-view>

```

Листинг 4.4 - Пример темплејта *modal* компоненте

```

$ionicModal.fromTemplateUrl('templates/modal/addImage.html', {
  scope: $scope,
  animation: 'slide-in-up'
}).then(function(modal) {
  $scope.modalAddImage = modal;
});

$scope.openModaAddImage = function(type) {
  ...
  $scope.modalAddImage.show();
  ...
};

$scope.closeModalAddImage = function() {
  $scope.modalAddImage.hide();
};

```

Листинг 4.5 - Пример логике *modal* компоненте

5.4 Кратке поруке

Када се заврши акција коју је изазвао корисник, потребно је обавестити корисника о статусу акције. У ту сврху се користи *toast* порука. То је компонента матичног дела систем. То значи да је потребно инсталирати одговарајући *Cordova* приклјучак. Следећом командом врши се његова инсталација:

```
$ ionic plugin add cordova-plugin-x-toast
```

Након инсталације, неопходно је регистровати *AngularJS* сервис *CordovaToast*. За приказ повратне поруке позива се метода *show*. Као параметри се прослеђују порука која се жели приказати, временски интервал приказа (може бити *short* или *long*) и позиција поруке на екрану (може бити *top*, *center* или *bottom*). Листинг 4.6 приказује пример употребе ове компоненте. Назив поруке је „*New photo is created.*“. Интервал приказа је кратак и порука ће се приказати при дну екрана.

```

$scope.addImage = function() {
  ...
  $cordovaToast.show('New photo is created.', 'short', 'bottom');
};

```

Листинг 4.6 - Употреба повратне поруке

5.5 Фотографисање

Фото-албум апликација чува корисничке фотографије у структури албума. Да би корисник могао правити фотографије помоћу свог уређаја, потребно је инсталирати *Cordova* прикључак. Следећа команда ће инсталирати прикључак за приступ камери уређаја:

```
$ ionic plugin add cordova-plugin-camera
```

Сервис који ће омогућити приступ камери и којег је потребно регистровати назива се *\$cordovaCamera*. Активност камере уређаја се приказује позивом методе *getPicture*. Метода поред функција успеха и грешке, прима објекат који садржи опције. Неке од опција су:

- *quality* – означава квалитет сачуване слике (вредности од 0 до 100);
- *destinationType* – формат повратне вредности:
 - *FILE_URI* враћа *URI* (енг. *Uniform Resource Identifier*) фајла;
 - *DATA_URL* враћа *base64* кодиран стринг;
- *sourceType* – поставља извор слике;
- *allowEdit* – омогућава измену слике пре селекције;
- *encodingType* – одабир *encoding*-а за слику;
- *saveToPhotoAlbum* – након фотографисања чува слику у албуму уређаја ;

Листинг 4.7 приказује начин употребе овог прикључка. Метода *getPicture* ће вратити као резултат *base64* кодиран стринг. Вредност стринга се додељује променљивој *imageCaptured* која ће бити прослеђена делу за кориснички интерфејс ради приkaza.

```
var options = {  
    quality: 100,  
    destinationType: Camera.DestinationType.DATA_URL,  
    sourceType: Camera.PictureSourceType.CAMERA,  
    allowEdit: false,  
    encodingType: Camera.EncodingType.JPEG,  
    saveToPhotoAlbum: false  
};  
  
$scope.$on('$ionicView.beforeEnter', function() {  
    $cordovaCamera.getPicture(options)  
    .then(function(imageData) {  
        $scope.imageCaptured = "data:image/jpeg;base64," + imageData;  
        $scope.imageData = imageData;  
        $ionicLoading.hide();  
    })  
});
```

```

}, function(err) {
  Console.log('Error while returning photo');
}) ;
...
});

```

Листинг 4.7 - Употреба прикључка за прављење фотографија

5.6 Преузимање локације уређаја

У спецификацији фото-албум апликације, класа *Image* садржи атрибут *location*. Да би се при чувању фотографија преузела локација уређаја, потребно је искористити посебан *Cordova* прикључак. Инсталација се извршава следећом командом:

```
$ ionic plugin add cordova-plugin-geolocation
```

Како би се преузела локација, потребно је укључити дозволу за локацију у подешавањима уређаја. Затим, потребно је регистровати *AngularJS* сервис *\$cordovaGeolocation* и позвати методу *getCurrentPosition*. Као параметар треба проследити објекат који садржи информацију о *timeout*-у и нивоу прецизности локације. Повратна вредност методе је објекат који садржи информацију о географској дужини и географској ширини. За претварање координата позиције у информацију који је град и држава у питању, искоришћена је *Google*-ова *maps* библиотека. Листинг 4.8 приказује употребу прикључка за преузимање локације уређаја.

```

var posOptions = {timeout: 10000, enableHighAccuracy: true};
$cordovaGeolocation.getCurrentPosition(posOptions)
.then(function (position) {
  var lat = position.coords.latitude;
  var long = position.coords.longitude;
  codeLatLng(lat, long).then(function(result) {
    userLocation = result;
    q.resolve(userLocation);
  });
})
}

```

Листинг 4.8 - Употреба геолокације у *Ionic* оквиру

5.7 Рад са фајл системом

Након рада на фотографији, потребно ју је сачувати. Да би се направио фајл и сачувао у меморији уређаја, неопходно је искористити *Cordova* прикључак за рад са фајл системом. Следећом командом се врши инсталација:

```
$ ionic plugin add cordova-plugin-file
```

Приклучак обезбеђује методе за креирање, читање, упис фајлова, креирање директоријума, итд. Могуће је специфицирати разне путање уређаја:

- **cordova.file.applicationDirectory** – одговара *file:///android_asset/* директоријуму који садржи ресурсе апликације;
- **cordova.file.externalCacheDirectory** – одговара *cache* директоријуму у који се смештају привремени кеширани фајлови;
- **cordova.file.externalRootDirectory** – одговара *<sdcard>/* директоријуму који представља коренски директоријум меморијске картице;
- **cordova.file.externalDataDirectory** – одговара *files* директоријуму у који се смештају фајлови којима може приступити само апликација (нису доступни кориснику);

Листинг 4.9 приказује употребу овог приклучка са примером креирања фотографије. Креирање фотографије извршава се позивом методе *createFile* којој се прослеђује путања и име фотографије. Након тога, позива се метода *writeFile* која ће уписати податке у креирани фајл, прослеђивањем путање, имена и података за упис.

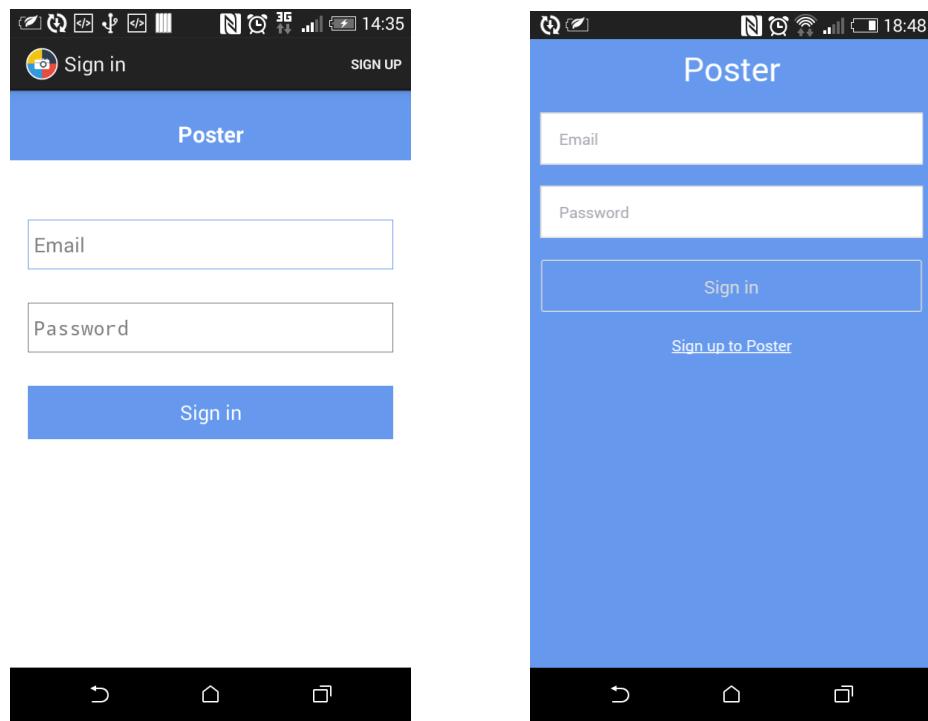
```
$cordovaFile.createFile(cordova.file.externalDataDirectory,
    name, false)
.then(function(result) {
    fileName = result.name;
    uri = result.nativeURL;
    $cordovaFile.writeFile(cordova.file.externalDataDirectory,
        fileName,
        Base64Binary.decodeArrayBuffer(data), true)
    .then(function(writeResult) {
        ...
        ImageService.addImage($scope.image.imageTitle,
            $scope.userLocation,uri, width,
            height, 'image/jpeg', album);
        $cordovaToast.show('New photo is created.', 'short', 'bottom');
        $state.go('app.albumDetail', {albumId: $stateParams.albumId});
    });
})
});
```

Листинг 4.9 - Употреба смештања фотографије на фајл систем

6. ДЕМО

У овом поглављу приказан је крајњи изглед фото-албум апликације. Упоређене су апликација имплементирана помоћу *Android* оквира и *Ionic* оквира. Прво ће бити дат приказ матичне апликације, а потом хибридне за исте екране.

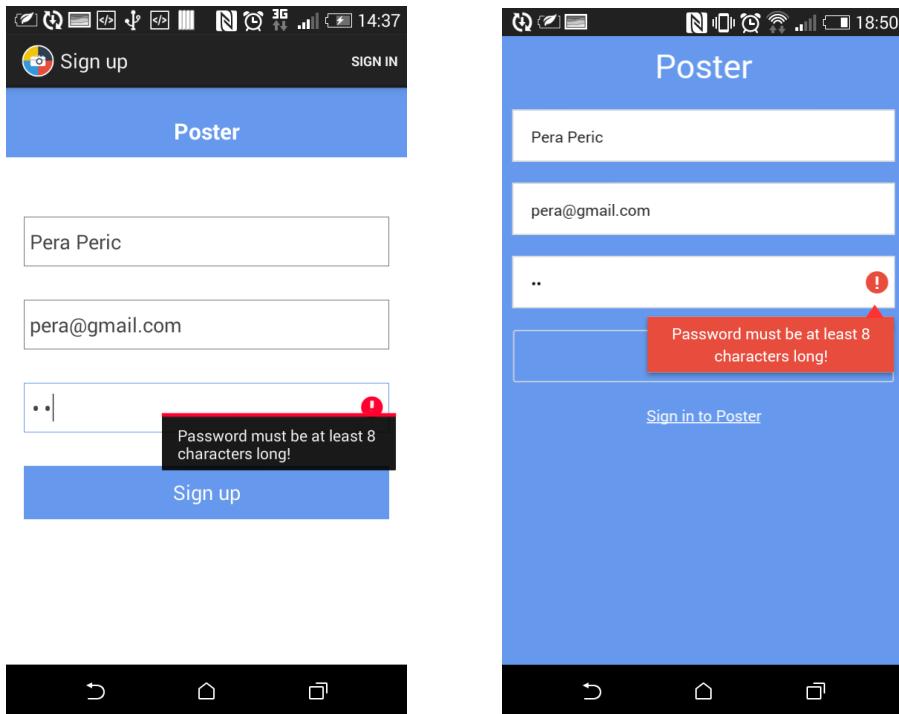
На самом почетку након уводног екрана, апликација приказује страницу за пријаву (слика 6.1). Екран за пријаву корисника код матичне и хибридне апликације су скоро идентични. Састоје се од два текстуална поља за унос *e-mail* адресе и шифре, дугмета за пријаву и опције за прелаз на екран за регистрацију. Једина разлика је у позицији опције за прелаз на екран за регистрацију. Код матичне апликације ова опција се налази у *Action bar-y* [1]. Док код хибридне апликације ова опција представља линк ка екрану за регистрацију и налази се испод дугмета за пријаву.



Слика 6.1 - Приказ странице за пријаву матичне и хибридне апликације

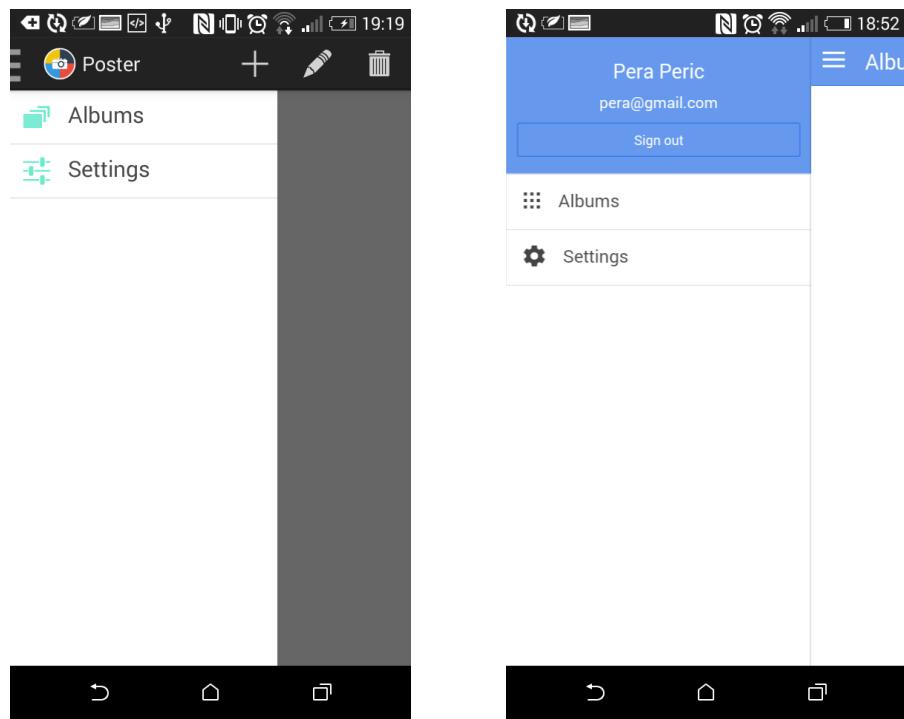
Ако корисник није регистрован, мора то да учини да би могао да се пријави. Након бирања опције за регистрацију приказује се посебна страница за ту операцију (слика 6.2). Екран за регистрацију корисника код матичне и хибридне апликације састоје се од два текстуална поља за унос пуног имена, *e-mail* адресе, шифре, дугмета за регистрацију и опције за прелаз на екран за пријаву. Разлика између екрана за регистрацију матичне и хибридне је у позицији опције за прелаз на екран за пријаву корисника. Код матичне апликације ова опција се налази у *Action bar-y*. Док код хибридне апликације ова опција представља линк ка екрану за пријаву и налази се испод дугмета за регистрацију.

Такође, виде се елементи валидације који обавештавају корисника да податак није исправан. Код матичне апликације ова порука је саставни део текстуалног поља за унос, док се код хибридне апликације користи посебна компонента звана *popover* компонента.



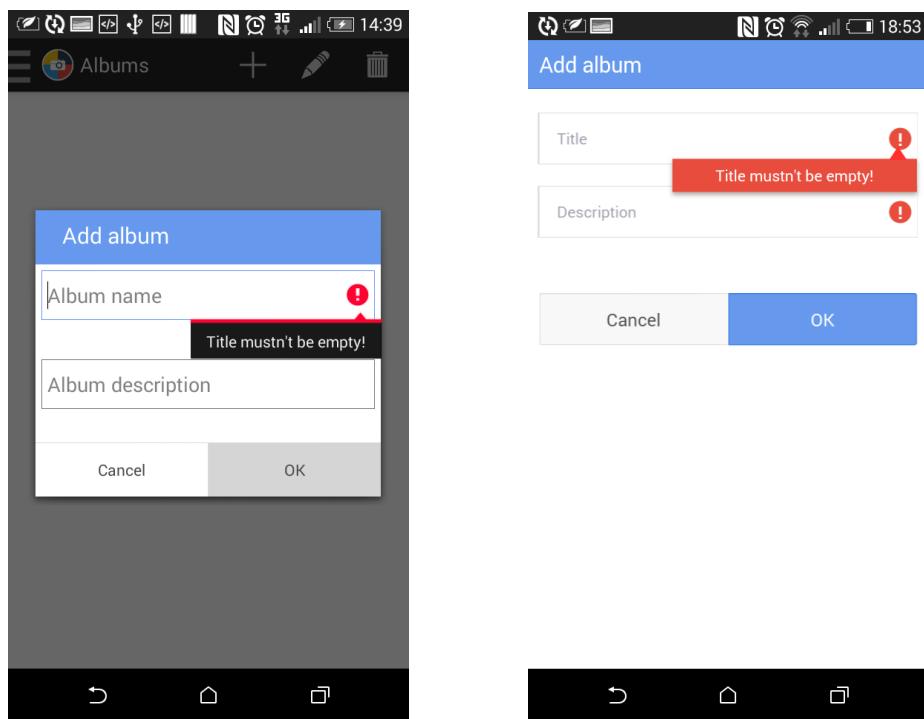
Слика 6.2 - Приказ странице за регистрацију матичне и хибридне апликације

После завршетка регистрације и успешне пријаве, приказује се почетни екран. У горњем левом углу је посебно дугме за приказ фиоке за навигацију (слика 6.3). Преко ње корисник може да изабере приказ албума или подешавања апликације. Код матичне апликације у *Action bar*-у се налазе опције за креирање, промену и брисање албума. Код хибридне апликације у *Action bar*-у се налази опција за додавање албума, док се опције за промену и брисање албума активирају селектовањем албума чиме се приказује *action sheet* компонента. За коришћење фиоке за навигацију код матичне апликације је потребна посебна имплементација. Код хибридне апликације је могуће искористити шаблон *sidemenu* при креирању новог пројекта. Тада ће фиока за навигацију бити доступна за употребу без било какве имплементације и подешавања.



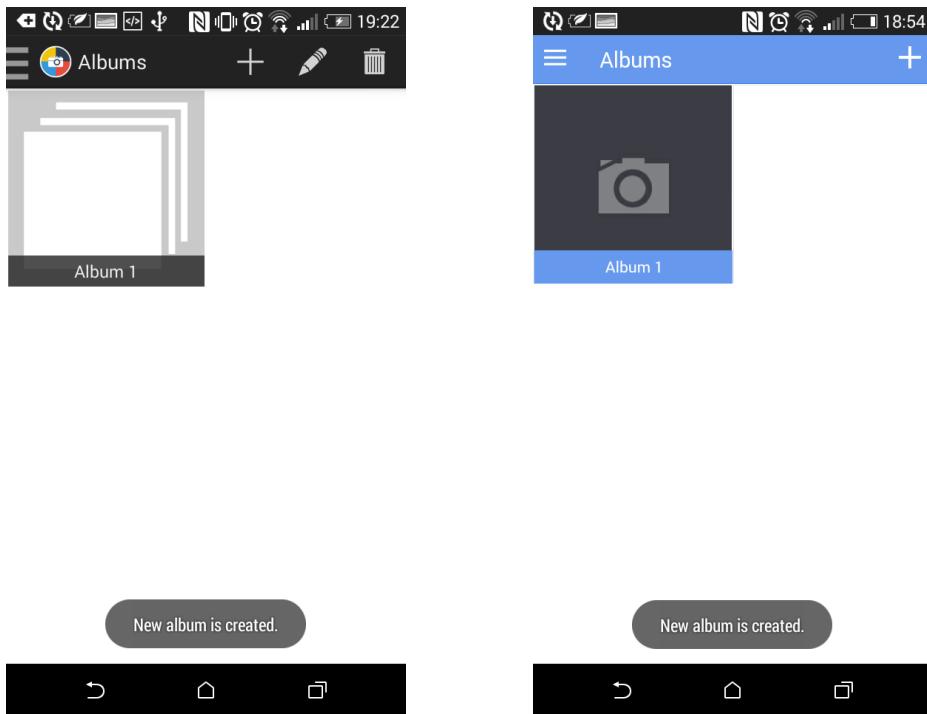
Слика 6.3 - Приказ фиоке за навигацију матичне и хибридне апликације

Одабиром опције за креирање новог албума, на матичној апликацији појављује се дијалог за унос назива и описа. На хибридној се у ту сврху користи *modal* компонента (слика 6.4). Дијалог код матичне апликације и *modal* компонента код хибридне апликације садрже исте компоненте корисничког интерфејса. Састоје се од два текстуална поља за унос имена и описа албума и два дугмента за прекид или потврду уноса новог албума.



Слика 6.4 - Приказ дијалога за унос новог албума матичне и хибридне апликације

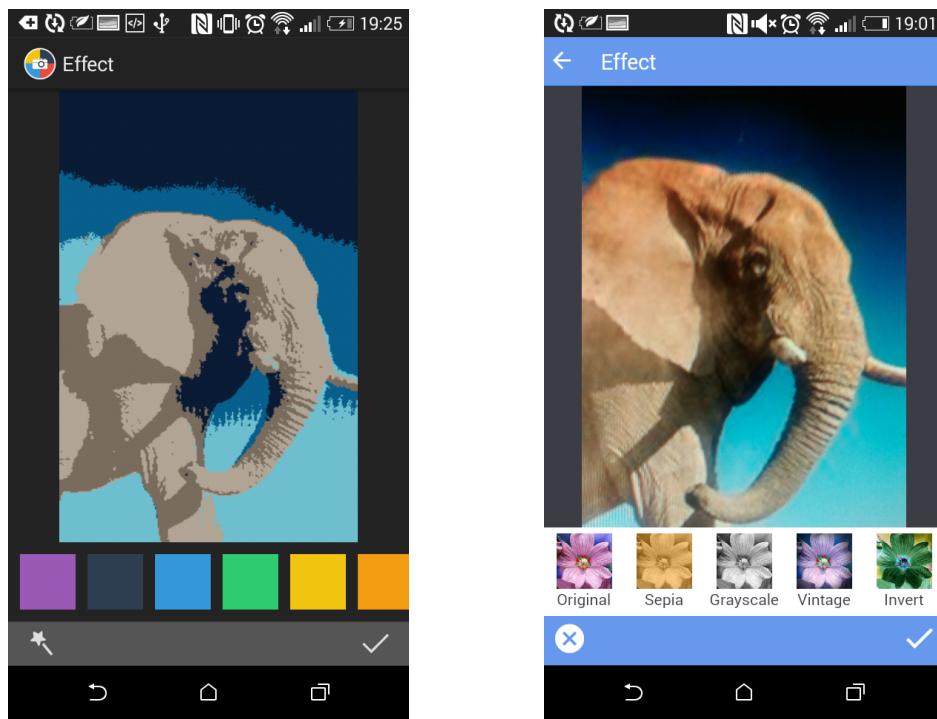
Након успешног додавања новог албума, приказује се у листи (слика 6.5). Како би се приказала листа албума код матичне апликације, потребно је имплементирати посебан адаптер за приказ сложенијих ставки листе. Једна ставка листе албума се састоји од слике празног албума или прве слике из албума и дела који приказује наслов албума. За приказ листе албума код хибридне апликације искоришћен је обичан *HTML* контејнер и стил за приказ листе у хоризонталном положају.



Слика 6.5 - Приказ успешног додавања албума матичне и хибридне апликације

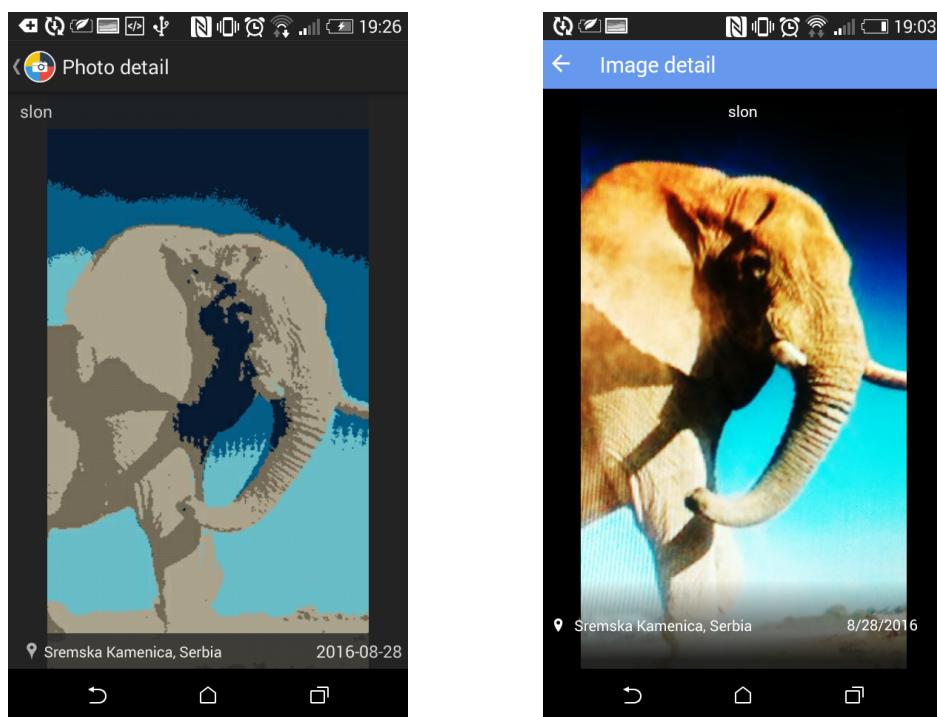
Кликом на албум, корисник се редиректује на екран за приказ детаља албума. Одатле је могуће изабрати опцију за креирање нове фотографије. Прво се приказује апликација Камера, а након тога прелази на екран за ефекте (слика 6.6).

Матична апликација омогућава примену ефекта који скенира све боје на слици и бира пет најзаступљенијих притиском на леву иконицу „чаробни штапић”. Нако тога, корисник селекцијом боје у листи и кликом на неки део слике мења боју тог дела у селектовану боју. Овај ефекат није примењен у хибридној апликацији јер би његово извршавање дуго трајало због слабе процесне моћи *JavaScript-a*. Због тога, кориснику је дата могућност примене једноставнијих ефеката попут инверзије боја, претварање слике у црно-белу, итд.



Слика 6.6 – Приказ екрана за ефекте матичне и хибридне апликације

Када је корисник задовољан применуним ефектом, потребно је да сачува фотографију. Након тога, може прегледати фотографију (слика 6.7). Екран за приказ детаља фотографије приказује наслов фотографије, локацију на којој је фотографија направљена и датум када је фотографија направљена.



Слика 6.7 – Приказ детаља фотографије матичне и хибридне апликације

7. ЗАКЉУЧАК

Задатак овог рада је анализа *Ionic* оквира и његова примена у имплементацији хибридне мобилне апликације. У уводном поглављу дат је преглед типова мобилних апликација, као и преглед развојних оквира за хибридне мобилне апликације. Затим су приказане технологије које користи *Ionic* оквир и дат је преглед најважнијих компоненти корисничког интерфејса. У поглављима о спецификацији захтева и дизајна специфицирани су захтеви фото-албум апликације која је дизајнирана уз ослонац на *Ionic* оквир. У поглављу о имплементацији су приказани битни елементи имплементације уз акценат на употребу *Cordova* приклучака који се користе за приступ матичним функцијама уређаја. Након тога је демонстрирана имплементирана апликација.

Сви типови мобилних апликација (матичне, веб и хибридне) имају своје предности и мање. Потребно је одлучити се за тип мобилне апликације који најбоље одговара захтевим пројекта.

Матичне апликације постижу најбоље перформансе захваљујући томе што су развијане технологијама специфичним за платформу. Апликације рачунане помоћу ових технологија се могу допремити на уређај из мобилних продавница. Велика мања овог типа мобилних апликација су велики трошкови развоја ако се апликација развија за више платформи (зато што ју је потребно независно развијати за сваку платформу).

Хибридне апликације користе најбоље одлике матичних и веб апликација. Платформски су независне што значи да се једном написан изворни код може употребити за генерирање апликације на више мобилних платформи. За развој се користе познате веб технологије (*HTML*, *CSS* и *JavaScript*).

Постоји велики број оквира за развој хибридних мобилних апликација. *Ionic* оквир се показао јако добро при развоју корисничког интерфејса зато што користи опробане технологије као што су *HTML*-а и *CSS*-а. Компонете показују велику флексибилност и могуће их је лако прилагодити жељеном дизајну. Могуће је релативно брзо развити кориснички интерфејс захваљујући већ коришћењу постојећих компоненти. Такође, омогућено је комбиновање постојећих компоненти и компоненти које су наменски направљене. За то је заслужан *AngularJS* оквир који подржава креирање нових компоненти проширујући синтаксу *HTML*-а. Све ове одлике омогућавају релативно једноставно креирање богатог и динамичног корисничког искуства. То је предност *Ionic* оквира у односу на *Appcelerator Titanium* оквир, код кога се могу јавити проблеми при креирању сложеног корисничког интерфејса јер је за сложени кориснички интерфејс потребан посебан код за сваку платформу.

Оквири *Framework 7*, *Kendo UI* и *Onsen UI* оквири не подржавају приступ матичним функцијама уређаја, за разлику од *Ionic* оквира који их подржава уз

помоћ прикључака. Употреба прикључака у Ionic оквиру је врло једноставна. Прво је потребно инсталирати жељени прикључак. Затим треба искористити већ постојећу *AngularJS* библиотеку за позивање функција које прикључак омогућава. Ако *ngCordova* нема подршку за одређени прикључак, потребно је направити сопствену имплементацију у виду *AngularJS* сервиса. Много већи проблем се јавља ако треба користити функцију матичне платформе за коју није имплементиран прикључак. Међутим, ово се ретко дешава зато што за већину матичних функција постоје имплементирани прикључци.

Када су у питању хибридне мобилне апликације, стиче се утисак да овакав тип апликација даје најбоље резултате ако се користи за конзумирање и приказ података. Једноставно, процесна моћ *JavaScript*-а који се извршава у оквиру „такног“ матичног контејнера је далеко мања од процесне моћи матичне апликације. Свака захтевнија обрада података може осетно успорити рад апликације што може имати утицај на крајњег корисника.

Што се тиче даљег развоја апликације описане у овом раду, могуће ју је унапредити имплементацијом синхронизације података са базом података у облаку, интеграцијом апликације са друштвеним мрежама и проширивањем скупа ефеката који се могу применити на фотографије. Имплементацијом синхронизације би се копија података налазила „на сигурном“. У случају да корисник обрише апликацију или да се уређај поквари, све што би било потребно да се реконструишу подаци је да се корисник региструје са истим корисничким именом. Тада би се извршила синхронизација података који би се поново нашли и у локалу. Поред синхронизације података, један од правца развоја би била и могућност дељења фотографија на најпопуларнијим друштвеним мрежама (*Facebook*, *Twitter*, *Google plus*, *Instagram*, итд.). Корисник би за креiranу фотографију имао опцију на екрану за приказ фотографије. Избором ове опције нудила би се могућност избора на коју друштвену мрежу ће фотографију поделити. Фото-албум апликација даје могућност примене једноставних ефеката. Један правац даљег развоја је проширење овог скупа ефеката и имплементација комплексних и оптимизованих ефеката који на процесирање фотографије неће утрошити много времена. Проширен скуп ефеката садржао би додавање текста на фотографију, могућност исецања дела фотографије и манипулацију исечком (померање, брисање, промена величине, итд.).

ЛИТЕРАТУРА

- [1] Android, <https://www.android.com/> (приступљено 15.09.2016.)
- [2] iOS, <http://www.apple.com/ios> (приступљено 15.09.2016.)
- [3] Eclipse, <https://eclipse.org/> (приступљено 15.09.2016.)
- [4] Android Studio, goo.gl/Haaovn (приступљено 15.09.2016.)
- [5] Java, <https://java.com/en/> (приступљено 15.09.2016.)
- [6] Xcode, <https://developer.apple.com/xcode/> (приступљено 15.09.2016.)
- [7] Objective-C, goo.gl/Gufpmm (приступљено 15.09.2016.)
- [8] Swift, <https://swift.org/> (приступљено 15.09.2016.)
- [9] Ionic Framework, ionicframework.com/ (приступљено 20.05.2016.)
- [10] Appcelerator Titanium, www.appcelerator.com/ (приступљено 20.05.2016.)
- [11] Sencha Touch, goo.gl/ht0oTB (приступљено 20.05.2016.)
- [12] Framework 7, <https://framework7.io/> (приступљено 20.05.2016.)
- [13] Kendo UI, www.telerik.com/kendo-ui (приступљено 20.05.2016.)
- [14] Onsen UI, <https://onsenui.io/> (приступљено 20.05.2016.)
- [15] Ionic forum, <https://forum.ionicframework.com/> (приступљено 22.05.2016.)
- [16] Cordova, <https://cordova.apache.org/> (приступљено 20.05.2016.)
- [17] AngularJS, <http://angularjs.org/> (приступљено 10.09.2016.)
- [18] Node.js, <http://nodejs.org/> (приступљено 15.09.2016.)
- [19] npm, <https://www.npmjs.com/> (приступљено 15.09.2016.)
- [20] ngCordova, <http://ngcordova.com/> (приступљено 10.09.2016.)

БИОГРАФИЈА



Кандидат Душко Алексић је рођен 02.04.1991. године у Лозници, држава Република Србија. Завршио је гимназију „Вук Караџић“ у Лозници 2010. године. Основне академске студије на Факултету техничких наука у Новом Саду завршио је 2014. године. Мастер академске студије студије на Факултету техничких наука у Новом Саду уписао је 2014. године. Положио је све испите предвиђене студијским програмом.