Primjena mobilne okoline u podršci pacijentima s autizmom

Novak, Marijan

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:614941

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Primjena mobilne okoline u podršci pacijentima s autizmom

Diplomski rad

Marijan Novak

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 05.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Marijan Novak
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 1075, 22.09.2017.
OIB studenta:	93160466387
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Dr.sc. Bruno Zorić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva:	Dr.sc. Bruno Zorić
Naslov diplomskog rada:	Primjena mobilne okoline u podršci pacijentima s autizmom
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu rada potrebno je opisati mogućnost korištenja mobilne platforme kao alata za pomoć pacijentima s autizmom, probleme s kojima se susreću ovi pacijenti i predložiti mogućnosti koje pruža mobilna okolina za njihovo ublažavanje. Osvrnuti se na ključne aspekte privatnosti i sigurnosti povjerljivih podataka kao i na postojeća rješenja dostupna za medicinske potrebe. U praktičnom dijelu rada ostvariti vlastito rješenje koje uključuje neke od dijagnostičkih, terapijskih ili metoda podrške opisanih u teorijskom dijelu u svrhu olakšavanja problema ili unaprjeđenja zdravstvenog stanja pacijenata. (sumentor: dr.sc. Bruno Zorić)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	05.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 08.09.2018.

Ime i prezime studenta:	Marijan Novak
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 1075, 22.09.2017.
Ephorus podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Primjena mobilne okoline u podršci pacijentima s autizmom**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora Dr.sc. Bruno Zorić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD ... 1

2. AUTIZAM I PROBLEMI KOJE DONOSI ... 2

2.1. Utjecaj autizma u svijetu i Hrvatskoj ... 4

2.2. Zdravstveni problemi s kojima se nose oboljeli ... 6

2.2.1. Napadaji i epilepsija ... 7

2.2.2. Poremećaji spavanja ... 7

2.2.3. Problemi vezani za probavu i prehranu ... 8

2.2.4. Glavobolje ... 9

2.2.5. Astma ... 9

2.3. Dijagnostički i terapijski postupci ... 9

2.3.1. Terapije s roditeljima ... 11

2.3.2. Primijenjena analiza ponašanja (ABA) ... 11

2.3.3. TEACHH pristup ... 12

2.3.4. CBT pristup ... 14

2.3.5. Farmakološke terapije ... 14

2.4. Važni pokazatelji zdravstvenog stanja i njihovo praćenje ... 15

2.5. Mogućnost primjene mobilne računalne okoline kod autističnih pacijenata ... 16

2.6. Postojeći računalni alati ... 17

2.6.1. MOSCOCO ... 17

2.6.2. Otismo ... 19

3. PROGRAMSKO RJEŠENJE ZA PODRŠKU PACIJENTIMA OBOLJELIM OD AUTIZMA ... 21

3.1. Zahtjevi na sustav ... 23

3.2. Model podataka ... 24

3.3. Korišteni alati i tehnologije ... 25

3.3.1. Android OS i Android Studio ... 25

3.3.2. Kotlin ... 28

3.3.3. MVVM ... 31

3.3.4. Architecture Components ... 32

3.3.5. Firebase ... 36

3.4. Način rada i struktura aplikacije ... 36

3.4.1. Prikaz prijave - *Login* ... 38

3.4.2. Glavni prikaz - *Main* ... 40

3.4.3. Roditeljski prikaz - *Parent* ... 41

3.4.4. Dječji prikaz - *Child* ... 44

3.5. Upute za korištenje 47
3.6. Testiranje implementacije 63
3.7. Anketa o uspješnosti implementacije 66
4. ZAKLJUČAK 69
LITERATURA 70
SAŽETAK 72
ABSTRACT 73
ŽIVOTOPIS 74
PRILOZI 75

1. UVOD

Tema ovog rada jest istražiti autizam te moguće terapijske metode primjenjive na mobilnu okolinu. Osobe s autizmom svakodnevno se susreću s brojnim problemima koje je moguće olakšati primjenom tehnologije. Ciljna skupina su djeca s autizmom pošto su simptomi kod njih mnogo izraženiji. Brojni odlasci na terapije, vježbe i ostalo znaju biti mukotrpni za dijete i roditelje te je ideja olakšati im svakodnevicu. Također, mnoga djeca vole tehnologiju i sve mogućnosti koje im pruža pa ih se uz tu činjenicu i korištenjem zanimljivog i atraktivnog sadržaja može potaknuti na svakodnevan rad. Ova vrsta terapijskog korištenja tehnologije već postoji, ali kvalitetnija softverska rješenja većinom se naplaćuju i nisu povoljna. Cilj je odabrati jednu ili više terapijskih metoda i implementirati ih pomoću mobilne okoline. Na taj način djeca s autizmom mogu raditi vježbe kod kuće i kad god žele, a roditelji mogu pratiti njihov napredak.

Konkretna implementacija terapijskih metoda izvest će se u obliku aplikacije za Android sustav kojoj će imati pristup svatko s Android pametnim telefonom. Aplikacija će biti podijeljena u posebne dijelove za djecu i roditelje, gdje će za pristup načinu rada za roditelje biti potrebna autentifikacija. Važan čimbenik bit će implementirati sučelje i sadržaj koji će potaknuti djecu na korištenje.

U drugom poglavlju bit će opisan autizam općenito, izazovi s kojima se oboljeli susreću i terapijske metode. Bit će obrađene mogućnosti primjene mobilne okoline za pomoć osobama s autizmom te već postojeća rješenja sa zaključcima o kvaliteti pojedinog. Treće poglavlje analizirat će zahtjeve na sustav na kojem rješenje mora raditi, te opisati korištene tehnologije. Bit će prikazan način rada sustava i njegovo testiranje. Na kraju, u četvrtom poglavlju dat će se zaključak o postignutim rezultatima te kratki opis načina izvedbe. Bit će iznesene prednosti i nedostaci izvedenog rješenja te dati smjernice o mogućim unaprjeđenjima.

2. AUTIZAM I PROBLEMI KOJE DONOSI

Autizam, ili poremećaj autističnog spektra, jest skupina neurorazvojnih poremećaja karakterizirana problemima sa socijalnim vještinama, ponavljajućim uzorcima ponašanja, govorom i neverbalnom komunikacijom, ali i jedinstvenim prednostima [1]. On je rezultat neurološkog poremećaja koji utječe na normalnu funkciju mozga. Postoji mnogo tipova autizma uzrokovanih različitim kombinacijama genetičkih utjecaja i utjecaja okoline. Ponekad se ovaj poremećaj javlja uz druge poput Downovog sindroma, epilepsije i drugog. Pojam autistični spektar odražava široke varijacije u problemima s kojima se susreću oboljeli i prednostima koje imaju. Najočitiji znakovi autizma najčešće se pojavljuju između dvije i tri godine starosti. U rjeđim slučajevima može se dijagnosticirati već s oko 18 mjeseci starosti. Ne postoji poznat jedinstven uzrok autizma, ali može se liječiti. Djeca ga ne prerastaju, ali znanstveno je dokazano da rana dijagnoza i intervencija znatno pomažu razvoju djeteta. Neki važniji pokazatelji uključuju sporije učenje jezika, izbjegavanje direktnog pogleda i održavanja razgovora, slabije motoričke sposobnosti, uzak opseg zanimanja te probleme s izvršavanjem zadataka i planiranjem. Često će se manifestirati držanjem svakodnevnih rutina i odupiranju čak i malim promjenama u tim rutinama [2]. Osoba s autizmom može imati mnogo ovih pokazatelja, samo nekoliko ili mnogo drugih. Dijagnoza se donosi prema analizi svih uzoraka ponašanja i njihove ozbiljnosti. Socijalne vještine često su najbolji pokazatelj prisutnosti ove bolesti. Interakcija autista s drugim osobama drugačija je od one kakvu ima većina populacije. Ako simptomi nisu toliko ozbiljni osoba će se činiti socijalno „nespretna“, te ponekad uvredljiva svojim komentarima. Kod ozbiljnijih simptoma želja za interakcijom s drugim osobama skoro potpuno nestaje. Također autizam se lako prepoznaje kod nedostatka empatije prema drugima. Osobe s autizmom ne razumiju osjećaje drugih osoba, ali čestim podsjećanjem ova sposobnost se značajno poboljšava. U nekim slučajevima to čak postaje i prirodna, nasuprot intelektualnoj radnji. Kroz razgovor s njima može se primijetiti puno manja razina razmjena ideja, mišljenja i osjećaja. Puno više će govoriti o sebi nego većina osoba. Osobe s autizmom u pravilu nisu sklone fizičkim dodirima, pa zagrljaji i slično mogu dati neočekivane rezultate. U ovoj situaciji često je u pitanju očekuje li osoba kontakt ili ne. Ovo se može odnositi i na iznenadne situacije poput glasnih zvukova, svjetla i mirisa. Ako osoba očekuje da će se nešto dogoditi ili zna da se nešto može dogoditi, mnogo će se lakše nositi s tim kad se dogodi. Mnoga djeca s autizmom uopće ne govore. Što je bolest ozbiljnija, više su pogođene vještine govorenja. Ponavljanje riječi i fraza koje čuju je učestalo. Osoba s autizmom voli rutine. Na primjer, ako prije spavanja oblači pidžamu, pa pere zube, a obično to radi obrnuto - uznemirit će se. Ove rutine mogu biti način na koji provode dan ili neke manje stvari koje učestalo

ponavljaju. Još jedna od učestalih pojava su opsesije određenim stvarima. Napredak i učenje mogu biti nepredvidivi. Djeca s autizmom mogu naučiti rješavati neke teže probleme, dok još imaju probleme s jednostavnijim. Može se reći da osobe s autizmom imaju „unutarnje kočnice“ koje im otežavaju uspostavljanje socijalne interakcije, verbalne ili neverbalne komunikacije, smanjuju razinu maštovitosti i senzorne integracije. Oni vide, čuju i osjećaju kao i svi ostali ljudi, ali te utiske ne mogu lako sklopiti u smislenu cjelinu, pa se zbog toga povlače u sebe. Imaju poteškoće s izražavanjem svojih osjećaja, želja, potreba i problema, što nerijetko njihovo ponašanje čini neobičnim.

Osobe s autizmom mogu imati od prosječne do ispodprosječne inteligencije. Na neverbalnim testovima inteligencije postižu mnogo bolje rezultate nego na verbalnim. Djeci s autizmom najčešće se onemogućuje druženje s vršnjacima skraćivanjem njihova boravka u vrtićima ili školama. Ovo usporava njihovo učenje socijalnih vještina i prilagodbu društvu. Neke osobe s autizmom talentirane su u područjima poput umjetnosti, matematike, pamćenja raznih podataka i drugih. Mogu imati izraženu sposobnost za samo jedno određeno područje, te ih tada nazivamo *autist-savant*. Ovaj izraz označava osobu koja je zaokupljena sitnicama, ali ne shvaća cjelinu. Nema sposobnost generalizacije te stalno ponavlja isto, drugih sadržaja nema. Njihove sposobnosti primijete se obično već u djetinjstvu, a razvojem drugih često se izgube ili više nisu naglašene. Neke od zanimljivih zabilježenih sposobnosti :

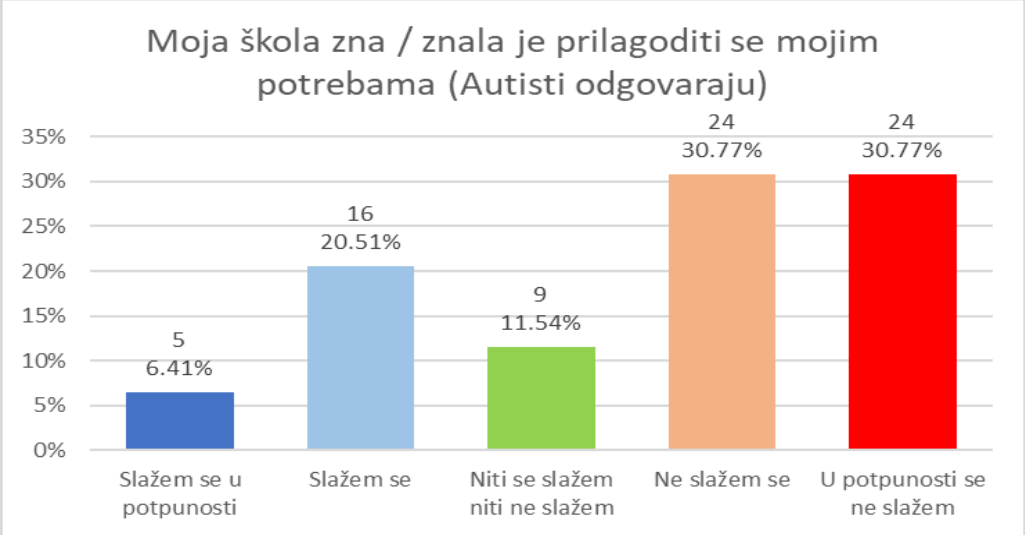
- Dijete od 6 mjeseci koje je moglo mrmljajući ponoviti cijele arije
- Blizanci koji su bez poteškoća računali s brojevima do čak 20 znamenaka
- Petogodišnji dječak koji je znao rastaviti i ponovno sastaviti elektroničke uređaje

Smatra se da uzrok ovih pojava hipokampus, dio limbičkog sustava velikog mozga odgovornog za emocije i dugoročno pamćenje [3]. Iz povijesti je poznato da su neki intelektualno nadprosječni ljudi imali neke od simptoma autizma. Na primjer, Albert Einstein nije govorio do pete godine i nije bio dobar učenik, a u djetinjstvu nije pokazivao nikakve posebne sposobnosti. Prema istraživanju Victora Lottera donesen je zaključak da 62 do 72% djece s autizmom ostaju cijeli život ovisne o drugima. Za manji dio djece (5-17%) navodi se dobra prognoza [3]. Ostali ostvaruju terapijski i edukacijski napredak, ali većini mnogi problemi ostaju prisutni. Christopher Gillberg navodi da djeca s boljom intelektualnom funkcijom imaju veću vjerojatnost za potpun oporavak. Naime, to je veoma rijetko. Oko polovice djece uspijeva postići razinu govora dostatnu

za socijalizaciju oko pete godine, ali ponekad i kasnije. Što su stariji odnosi s drugim osobama im se poboljšavaju, no ti odnosi i dalje ostaju neuobičajeni. Pokazuju rezerviranost i ne suosjećaju s drugima. Autizam ne utječe znatno na životni vijek, a jedini bitno veći postotak mortaliteta je u ranoj dobi do treće godine.

2.1. Utjecaj autizma u svijetu i Hrvatskoj

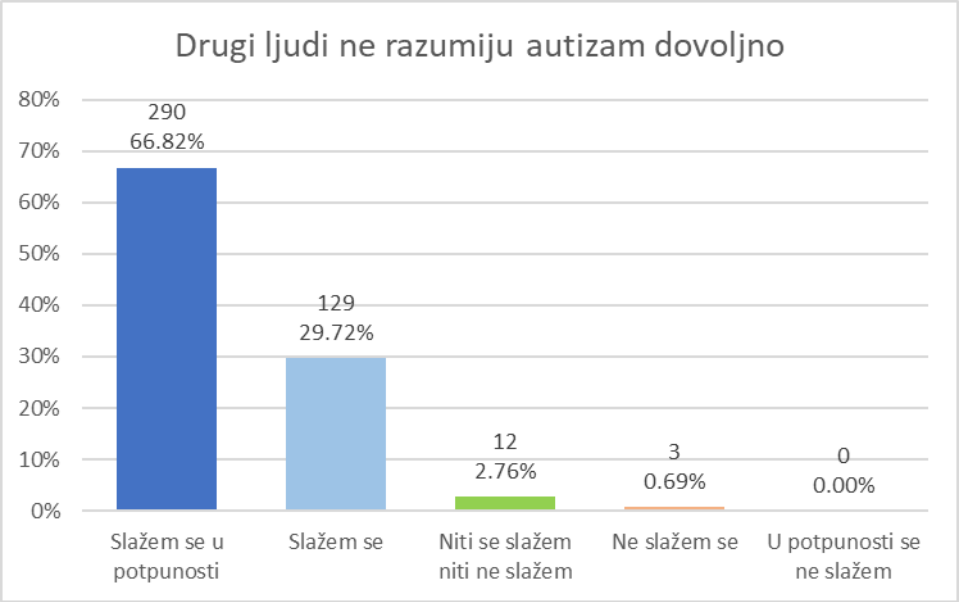
Mnoge osobe s autizmom nemaju pristup odgovarajućoj dijagnozi i uslugama, pa ne primaju potrebnu skrb. U školama učenici s autizmom predstavljaju jedinstven izazov učiteljima i nekad je teško zadovoljiti njihove neuobičajene potrebe. Zato učitelji trebaju biti dostatno educirani o autizmu i kako pomoći djeci koja boluju od njega da napreduju. Zdravstvene ustanove trebaju imati dostatan broj stručnjaka i pomagala koji će osobama s autizmom svakodnevno pomagati u razvoju. U Hrvatskoj ne postoji organizirana dijagnostička služba za autizam. Jedina takva služba postojala je u Zagrebu, no zatvorena je 1995. godine. U svijetu sve više autista završava škole, fakultete i osposobljava se za zanimanja, no kako bi se trend nastavio i u Hrvatskoj potrebno je organizirati sve potrebne usluge**.** Na slici 2.1. vidljiva je razina prilagodljivosti škola autističnih ispitanika njihovim potrebama prema provedenoj anketi [4].



Slika 2.1. Rezultati ankete o opremljenosti škole za učenike s autizmom [4]

Osobe s autizmom su isključene iz društva te ih ljudi nerijetko smatraju čudnima i neugodnima. Svjetski dan svjesnosti o autizmu obilježava se 2. travnja i nastoji ljudima približiti ovaj kompleksan poremećaj i objasniti ga. Prema anketi web stranice *Autistic not weird* oko 97%

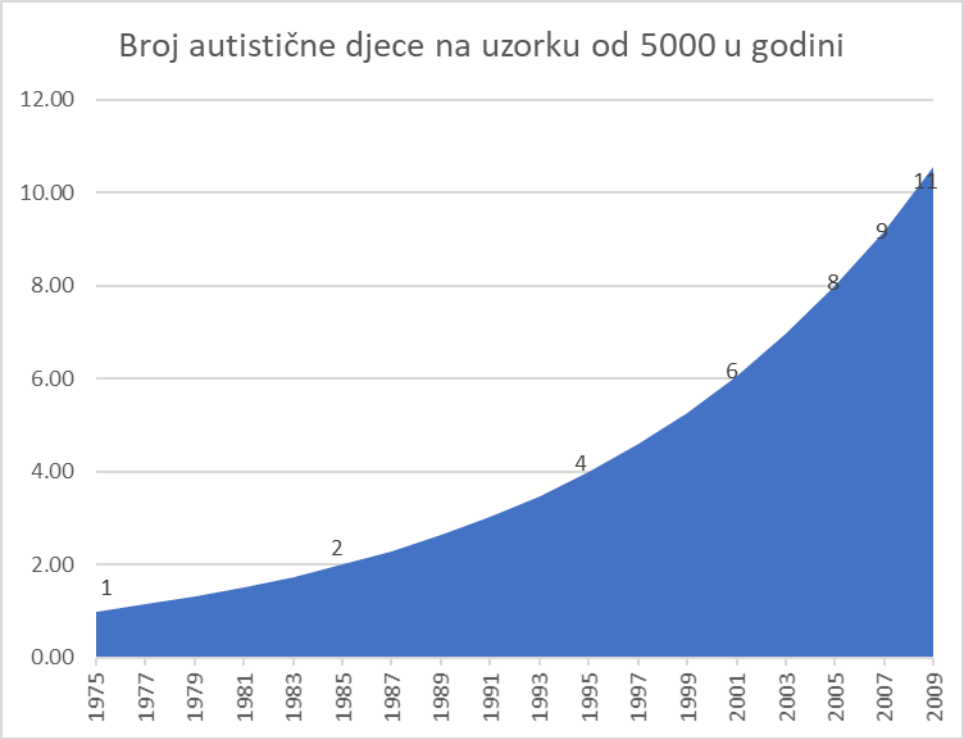
osoba s autizmom i onih koje imaju bliske osobe a autizmom slažu se da ljudi ne shvaćaju autizam dovoljno. Točni rezultati vidljivi su na slici 2.2.



Slika 2.2. Rezultati ankete svijesti ljudi o autizmu [4]

Do nedavno je autizam bio smatran rijetkom pojavom s učestalošću 10 od 10 000 djece. Međunarodna udruga *Autism Europe* procjenjuje prevalenciju od 1 na 100 djece, a u Europi broji 5 milijuna autističnih osoba [5]. Predviđa se da će se u budućnosti broj djece s autizmom znatno povisiti. Novije izvješće iz 2014. Američkog Centra za kontrolu bolesti i prevenciju (CDC) pokazuje prevalenciju autizma u SAD-u od 1 oboljelog na 68 djece, što je više od 1 na 88 djece u 2012. i 1 na 110 djece u 2006. godini [6]. U Hrvatskoj ima oko 22 000 osoba s autizmom, a oko pola milijuna ljudi je član obitelji osobe s autizmom. Naime, službeno je registrirano samo 1096 osoba [3]. Velik broj osoba s autizmom dijagnoziran je krivo i nije registriran zbog nedovoljnog poznavanja tih poremećaja i nepostojanja službenog centra za dijagnozu. Dječaci su 4 do 5 puta podložniji autizmu od djevojčica. U nekim je obiteljima autizam 50 do 100 puta češći od normalnog iz čega se vidi genetski utjecaj na pojavu ovog poremećaja. Nije poznato manifestira li se porast dijagnoza autizma zbog usavršavanja tehnike same dijagnoze ili stvarnog porasta broja oboljelih. Problematično je uspoređivati prevalencije autizma u svijetu u zadnjih 30 godina jer su se kriteriji mijenjali prema Dijagnostičkom i statističkom priručniku (engl. *Diagnostic and Statistical Manual,* DSM). U ranijim verzijama DSM-a kriteriji za dijagnozu bili su mnogo stroži, pa se vjeruje da se broj oboljelih osoba nije znatno povećavao. U SAD-u su ukupni troškovi

liječenja za osobe s autizmom do 6.2 puta veće u odnosu na uobičajene troškove liječenja [7]. Na slici 2.3. vidljiv je porast prevalencije autizma u godinama od 1975. do 2009.



Slika 2.3. Prevalencija autizma u svijetu po godinama [6]

2.2. Zdravstveni problemi s kojima se nose oboljeli

Autizam sam po sebi donosi širok spektar problema s kojima se oboljeli moraju nositi, ali uz njega često se pojavljuju i drugi zdravstveni problemi. Često ovi popratni zdravstveni problemi otežavaju dijagnozu autizma i navode stručnjake na krive dijagnoze. U nastavku će se opisati neki češći poremećaji. Autizam je nerijetko popraćen nekim drugim imunološkim poremećajima. Istraživanja ukazuju na genetičke predispozicije u obitelji, ali i na nešto veću mogućnost autizma kod djeteta majke koja je tijekom trudnoće doživjela upale ili infekcije imunološkog sustava. Na slici 2.4. vidljivi su rezultati istraživanja korelacije ovakvih slučajeva i autizma. Prikazan je broj slučajeva autizma ovisno o periodu kada je majka imala neku od ovih bolesti i koju određenu bolest je imala. Očito je da postoji značajna korelacija.

Period izlaganja	Broj slučajeva	Broj pregleda
Autoimune bolesti		
Prije trudnoće	22	98
Prvo tromjesečje	4	16
Drugo tromjesečje	4	20
Treće tromjesečje	14	54
Poslije poroda	20	71
Astma		
Prije trudnoće	42	140
Prvo tromjesečje	10	20
Drugo tromjesečje	14	37
Treće tromjesečje	23	77
Poslije poroda	21	79
Alergije		
Prije trudnoće	48	173
Prvo tromjesečje	7	35
Drugo tromjesečje	11	27
Treće tromjesečje	6	18
Poslije poroda	54	211

Slika 2.4. Slučajevi autizma u ovisnosti o autoimunim bolestima majke tijekom trudnoće ili prije [8]

2.2.1. Napadaji i epilepsija

Napadaji i epilepsija su jedan od najčešćih bolesti povezanih s poremećajima iz spektra autizma [9]. Statistika pokazuje da je vjerojatnost da dijete s autizmom ima epilepsiju u opsegu 5-38% nasuprot 1-2% koji vrijede za osobe bez autizma. Napadaji mogu nastupiti u bilo kojem stadiju života osobe, ali tipično se pojavljuju u adolescenciji, s tim da je prosjek oko 13 godina [9]. Najrizičnija skupina u ovom pogledu jest ona s mentalnim ograničenjima. Također, napadaji se pojavljuju češće kod osoba koje koriste antipsihotike u okviru terapije autizma. Iako se ishrana ne smatra čimbenikom u uzrokovanju napadaja, postoji mnogo slučajeva u kojima ih je ketogena dijeta značajno ublažila pa čak i potpuno eliminirala. Prema ovoj dijeti povećava se razina unosa masti na onu četiri puta veću od ugljikohidrata i proteina. Prema *Epilepsy Foundation* pomogla je otprilike trećini pacijenata [10].

2.2.2. Poremećaji spavanja

Djeca s poremećajima iz spektra autizma imaju značajne probleme sa spavanjem. Ovo uključuje čak 80% njih [11]. Najčešći oblici problema su ti da se često bude i teško zaspu. Mnogo je čimbenika koji uzrokuju ove poremećaje, ali jedan čest je činjenica da djeca s autizmom trpe veće razine stresa i tjeskobe. Neki od ovih problema mogu se pripisati problemima s ponašanjem

u slučaju gdje se dijete ne može priviknuti na postavljanje granica poput vremena za spavanje. Vrijedi i suprotno gdje poremećaji spavanja mogu uzrokovati probleme s ponašanjem. Uzimanje lijekova također može utjecati na spavanje. Roditelji trebaju uspostaviti rutine i optimalne uvjete za djetetov san. Neke od najboljih praksa su držanje sobe gdje dijete spava mračnom i svježom, izbjegavanje uzimanja kofeina, uspostavljanje točnog vremena za odlazak na spavanje i osiguravanje dovoljno kretanja i vježbe djetetu.

2.2.3. Problemi vezani za probavu i prehranu

Još jedna uobičajena bolest kod djece s autizmom je gastrointestinalni poremećaj. Procjenjuje se da ga čak do 85% ima u nekom obliku [12]. Simptomi ovog poremećaja uključuju povraćanje, konstipaciju, bol u trbuhu, žgaravicu i proljev. Kao i poremećaji spavanja, pridonosi problemima u ponašanju. Nije poznato zašto dolazi do ovog poremećaja, ali kao mogući razlog nameće se dijetalna alergija. U nekim slučajevima uklanjanje mliječnih proizvoda i glutena iz prehrane pokazalo je znatno poboljšanje. Probiotici se također pokazuju korisnima. Kao drugi mogući razlog navodi se povezanost sa slabijim imunitetom osoba s autizmom. Djeca koja boluju od gastrointestinalnih poremećaja nekad ne pokazuju tipične simptome pa nekad roditelji i ne znaju da problem postoji. Neki pokazatelji gastrointestinalnog poremećaja su pretjerano kašljanje, poteškoće s gutanjem hrane, pretjerano žvakanje, neobjašnjive promjene u ponašanju i drugi. Rano prepoznavanje simptoma ovog poremećaja može rezultirati smanjivanjem problema s ponašanjem. Djeca sa osjetilnim problemima izložena su opasnosti poremećaja ishrane. Ovo uključuje teškoće s ishranom ili gutanjem hrane određenih mirisa, tekstura i boja. Ovakva djeca izbirljiva su i nerijetko će se poprilično uznemiriti kada ih se tjera da jedu određenu hranu. Kod do 70% djece s autizmom pojavit će se simptomi poremećaja ishrane [13]. U 1990-ima godinama dolazi do velikog povećanja u broju dijagnoza autizma, ali i dijetalnih alergija. One, kao i autizam, imaju poveznicu sa imunološkim sustavom, zbog čega se razmatra mogućnost da su i oni nekako povezani. Dijetalne alergije nisu opasne po život, ali mogu pogoršati simptome autizma. Upale uzrokovane reakcijom na određene vrste hrane mogu uzrokovati pretjeranu aktivnost mozga, ali ju i usporiti. Hrana koja dokazano loše utječe na osobe s autizmom jest gluten, mliječni proizvodi, šećer, kukuruz i prerađevine poput konzervansa i umjetnih zaslađivača. Otkrivanje utjecaja određenih vrsta hrane izvodi se procesom eliminacije.

2.2.4. Glavobolje

Klasične glavobolje su jedan od rjeđih popratnih problema autizma. Jedan od razloga jest to što se autizam povezuje sa slabijom osjetljivošću na bol. Ovo znači da dijete s autizmom ne osjeća bol tipične glavobolje kao ono bez njega. Naime, migrene su mnogo češće. Studij [14] iz 2014. godine ukazao je da su im djeca sa osjetilnom hiperaktivnošću podložnija. Uz njih, podložna su i djeca koja trpe veće razine tjeskobe. Glavobolje mogu biti uzrokovane i gastrointestinalnim poremećajem. Promjene u prehrani, terapija ponašanja i ostali pristupi smanjivanja tjeskobe imaju značajnu ulogu u smanjivanju simptoma.

2.2.5. Astma

Nejasno je u kojoj je mjeri astma povezana a autizmom jer je ona sama vrlo čest respiratorni poremećaj. Naime, istraživanjima se pokazuje sve veća korelacija. Astma je poremećaj uzrokovan problemima s imunološkim sustavom, a kod osoba s autizmom imunološki i upalni procesi su nebalansirani. Povezanost oba poremećaja s imunološkim sustavom ukazuju na korelaciju.

2.3. Dijagnostički i terapijski postupci

Dijagnostički postupci nastoje odrediti uzroke, ozbiljnost poremećaja, postojećih i potencijalnih mogućnosti djeteta, te dijagnostičku raspodjelu prema pojedinim simptomima [3]. Pri dijagnozi prisutni s psihijatar, psiholog i edukacijski U dijagnosticiranju sudjeluju psihijatar, psiholog i edukacijski defektolog. Pri dijagnozi se promatra kako se dijete ponaša, tj. prisutnost ili odsutnost nekih učestalih uzoraka ponašanja, samopovređivanja, prisutnost verbalne i neverbalne komunikacije, mogućnosti za imitaciju, igru i socijalnu interakciju itd. Prisutnost samo nekih ovih znakova ne mora značiti ništa, no ako ih je više potrebna je intervencija. U kojoj mjeri će se poremećaj izliječiti ovisi o što ranijem otkrivanju i istovremenom uključivanju u terapijske programe. Iako je kritično vrijeme za dijagnozu autizma između 24. i 36. mjeseca života, dijagnoza se postavlja tek nakon 8. godine života. To se čini nakon dužeg praćenja jer neki simptomi mogu značiti i sasvim drugi poremećaj. U nastavku će se navesti grupe simptoma prema kojima se može postaviti dijagnoza. Prva grupa su poteškoće sa socijalnim interakcijama. Može se primijetiti oštećenje neverbalnih načina interakcije poput pogleda u oči, izraza lica, držanja i gesta. Odnos s vršnjacima se teško razvija, a empatija je na nezavidnoj razini. Druga važna grupa je komunikacija. Izražene su teškoće u verbalnim i neverbalnim oblicima komunikacije. Razvoj govora kasni ili izostaje, a nije praćen drugim načinima komunikacije. Osoba ne započinje i ne održava razgovor,

a kada ga vodi ne pokazuje emocije i interes. Cijela komunikacija se često svodi samo na osnovne životne potrebe i trenutnu situaciju. Treća grupa su obrasci ponašanja. Osobe s autizmom često su zaokupljene nekim interesom abnormalnim obzirom na intenzitet ili usmjerenost. Stalno ponavljanje nekih pokreta i akcija te inzistiranje na rutinama vrlo su uobičajeni. Simptomi se mogu podijeliti i prema dobi osobe. Najraniji znakovi autizma kod dojenčeta su poremećaji ishrane, odbijanje tjelesnog kontakta, plašljivost te smanjen interes za igračke. Kod djece do 2 godine pokazatelji autizma su izbjegavanje kontakta očima, neodazivanje na vlastito ime i slično. Dijete je nezainteresirano za igru s drugom djecom, igra se samo i uvijek ponavlja igre. Pri igri ne koristi maštu. Nema povezivanja s drugim osobama, čak ni onim bliskim. Još neki važni znakovi su kada dijete nije zahtjevno i gotovo ništa ne traži, ne boji se nepoznatih osoba, pokazuje izljeve bijesa bez očitog razloga te opsesivnost pojedinim objektima i podražajima. U nastavku na slici 2.5. prikazan je primjer uobičajenih simptoma i znakova autizma.



Slika 2.5. Dječak pokazuje velik interes za vlakove i repetitivan obrazac ponašanja

S povećanjem dijagnoza autizma kod novorođenčadi i djece u zadnjih dvadeset godina, nastaje sve veća potreba za efektivnim i ranim intervencijama. Prema [15] za djecu od rođenja do 8 godina starosti dane su smjernice koje treba slijediti za što efektivnije liječenje:

- Početi s terapijskim programima što je prije moguće
- Intenzivne terapije, pet puta tjedno po pet sati dnevno
- Iskorištenje repetitivnih prilika za učenje kroz kratke periode
- Individualizirana posvećenost svakom pacijentu na dnevnoj bazi

- Uključivanje obitelji u terapije
- Mehanizmi procjene napretka i mogućih prilagođavanja programa terapije

2.3.1. Terapije s roditeljima

Uključenje roditelja i cjelokupne obitelji smatra se značajnim elementom terapijskih programa za djecu s autizmom. Osnovna pretpostavka terapije ponašanja s roditeljima je da je ponašanje djeteta naučeno od njih samih. Roditelji trebaju biti naučeni reagirati na sve moguće situacije i poticati primjereno ponašanje djeteta pozitivnom motivacijom. Postoje mnogi dokazi koji pokazuju da je terapija s roditeljima efektivna metoda poboljšanja socijalnih vještina djeteta s autizmom. Naime, vrlo je važan način na koji su roditelji uklopljeni u proces terapije. Svaki pacijent je različit i terapija se mora prilagoditi svakom od njih. Moguće je i da dijete nema koristi od terapije s roditeljima. Naime u većini slučajeva zabilježen je dodatan stabilan napredak kada su roditelji uključeni u terapiju.

2.3.2. Primijenjena analiza ponašanja (ABA)

Primijenjena analiza ponašanja je znanost prilagodbe ponašanja u kojoj se primjenjuju postupci poboljšanja ponašanja primjerenog okolini i stjecanja novih socijalnih vještina kroz intenzivne i ciljane treninge. Ova analiza koristi proces koji započinje razvojem plana terapije ukazujući na probleme ponašanja pacijenta, zatim odabire primjere tehnike i kontinuirano evaluira napredak pacijenta i po potrebi modificira terapije. Točna dijagnoza i vrsta intervencije moraju se odrediti i prilagoditi prema potrebama svakog pojedinog djeteta. Procjene napretka se baziraju na skupini strategija koje pružaju informacije o varijablama povezanim sa specifičnim ponašanjem. Tehnike korištene kod ove metode su:

- Učenje s pozitivnom motivacijom: korištenje pohvala, hrane, igračaka i sličnih pozitivnih podražaja za postizanje željenog ponašanja
- Oblikovanje ponašanja nagrađivanjem aproksimacija ili komponenti željenog ponašanja sve dok se ono ne pojavi
- Postupno smanjivanje učestalosti poticanja na primjereno ponašanje radi povećanja neovisnosti
- Odstranjivanje motivacije za neželjeno ponašanje
- Kažnjavanje neželjenog ponašanja negativnim podražajima

- Diferencijalna motivacija, tj. pozitivni podražaji za primjereno ponašanje i odsutnost neželjenog ponašanja

Terapijski programi zasnovani na primijenjenoj analizi ponašanja su trenutno prva linija liječenja autizma u ranom djetinjstvu. Lovaas i ESDM modeli su ovakvi terapijski programi i postoji mnogo provedenih istraživanja koja potvrđuju njihovu efikasnost u poboljšanju razumijevanja, govora i adaptivnog ponašanja. Naime, čak i nakon dvije godine intenzivnih terapija po ovim programima većina djece u programu i dalje pokazuje simptome autizma. Empirički najuspješnijim programom pokazuje se EIBI (*Early Intensive Behavioral Intervention*) koji koristi Lovaas model. Ključne značajke programa su fokus na rani razvoj (djeca mlađa od 5 godina), intenzitet terapija, metode koje odrasli mogu primjenjivati kako bi pacijentima, sistematski pristup i relativna jednostavnost razumijevanja koncepta programa. Proces ABA terapije vidljiv je na slici 2.6.



Slika 2.6. ABA terapija

2.3.3. TEACHH pristup

Ovo je pristup namijenjen uporabi u školskim učionicama koji uključuje i profesionalno treniranje prosvjetnog osoblja kako bi tijekom nastave mogli pomagati djeci s autizmom. U širokoj je primjeni u sjevernoameričkim obrazovnim institucijama. Bitne značajke su strukturirano učenje, vizualno planiranje i predvidljivo okruženje u kojem se osobi s autizmom povećava samostalnost u situacijama s kojima se sreće u svakodnevnom životu [16]. Osnova strukturiranog učenja sastoji

se od organiziranja okruženja i aktivnosti na način koji dijete može razumjeti, koristeći njegove interese i snage u poticanju na učenje. Bitni aspekti strukturiranog učenja:

- Fizička struktura – razmještaj učionice i nastavnog materijala unutar nje
- Dnevni raspored – učenik zna što sve treba čini i kojim redom. Može biti prikazan predmetima, slikama ili u pisanom obliku
- Način rada – strukturirani sustav koji učeniku objašnjava što treba raditi, što sve taj rad podrazumijeva, kako ga treba izvršiti i što slijedi nakon njega
- Vizualna podrška – učeniku se zadatak daje crtežima ili slikama, naglašava se važnost svake akcije i njihovog redoslijeda

TEACHH pristup nastoji potaknuti osobe s autizmom na međusobno prihvaćanje različitosti. On se zasniva na ideji da se umjesto ublaživanja simptoma autizma okolina može prilagoditi osobi s autizmom. Tako se iskorištava maksimalan potencijal za edukaciju osobe s autizmom. Prema ovom pristupu osobe s autizmom su različite od drugih, ali nisu manje vrijedne. Kako bi učionica bila prilagođena TEACHH pristupu mora imati izričito odvojene dijelove za individualan ili grupni rad, slobodno vrijeme i odmor. Ovakvo odvajanje pomaže učenicima da znaju što točno trebaju raditi u kojem dijelu učionice. Uspješno implementiran TEACHH pristup umanjuje potrebu za uputama i naredbama učitelja za svaku pojedinu radnju ili zadatak te povećava samostalnost učenika. Ovim programom može se efikasno potaknuti dijete s autizmom na usredotočeno i samostalno rješavanje zadataka uz promjenu okolinu bez problema. Ističe se kako samostalnost djeteta uvelike utječe na njegovu sposobnost učenja. TEACHH priprema djecu s autizmom za samostalan život i posao. Novija istraživanja ističu ovaj pristup kao znanstvenu metodu. Na slici 2.7. vidljiva je učionica organizirana prema TEACHH pristupu.



Slika 2.7. Učionica raspoređena prema TEACHH pristupu

2.3.4. CBT pristup

CBT je psihosocijalna metoda intervencije koja pomaže osobama s autizmom i drugim poremećajima nositi se s problemima mijenjajući način na koji se ponašaju i razmišljaju. Preporuča se kod djece s manje izraženim poteškoćama. Uspješna je kod pretjeranog straha i tjeskobe koji nerijetko dolaze uz autizam. Može pomoći i kod opsesija i ispada. Razina uspješnosti doseže i do 78% što je vrlo važno jer čak do 40% djece s autizmom ima poteškoća s tjeskobom i strahovima [17]. Ovaj pristup uči dijete da bolje upravlja svojim emocijama i razvije samokontrolu. Tipični CBT terapijski program sastoji se od 6 do 18 sesija trajanja oko jedan sat sa razmacima od jednog do tri tjedna između sesija.

2.3.5. Farmakološke terapije

Farmakološke terapije su kod autizma u širokoj upotrebi kao dodatna terapija uz neku od prije navedenih psihosocijalnih. Cilj ove terapije je kontroliranje čestih popratnih simptoma autizma poput nesanice, hiperaktivnosti, impulzivnosti, agresivnosti, depresije, opsesije i drugog. Istraživanja su pokazala da 70% pacijenata ima bar jedan od ovih popratnih simptoma, a preko 50% ih ima dva ili više [18]. Trenutno ne postoje lijekovi koji mogu izliječiti socijalne i

komunikacijske poteškoće kod autizma, ali je upravo to tema intenzivnih znanstvenih istraživanja. Liječenje autističnih osoba lijekovima je uobičajena klinička praksa. Najkorišteniji lijekovi su redom antidepresivi, antipsihotici, antikonvulzivi i stimulansi.

2.4. Važni pokazatelji zdravstvenog stanja i njihovo praćenje

Osnovna klinička istraživanja o autizmu ovise o preciznoj ocjeni ozbiljnosti najčešćih simptoma. Istraživači koriste mnoge alate radi određivanja pokazatelja ozbiljnosti autizma, a neki od njih su ADI-R (*Autism Diagnostic Interview-Revised),* ADOS (*Autism Diagnostic Observation Schedule*), CARS (*Childhood Autism Rating Scale*) i SRS (*Social Responsiveness Scale*) [9]. Naime, rezultati ovih alata ovise značajno o verbalnim vještinama, starosti djeteta i IQ-u te nisu uvijek primjereni za usporedbu ozbiljnosti simptoma među oboljelima. ADOS je na primjer efektivan u kategorizaciji djece koja su zasigurno autistični ili ne, ali ne pravi preciznu razliku među djecom s blažim simptomima. Radi se na standardizaciji bodovanja simptoma pomoću ovih alata kako bi se lakše metrički odredila ozbiljnost stanja pacijenta, rasporedilo ih se u kategorije i olakšalo liječenje. Ovi alati znatno unaprjeđuju razumijevanje autizma. Ne postoje određene vrste liječenja teških oblika autizma, nego se rješavaju ciljanjem pojedinih simptoma.

Osobe s vrlo ozbiljnim simptomima nerijetko pokazuju ekstremne načine ponašanja koji su većinom rezultat frustracije, prevelikog broja vanjskih podražaja ili fizičke boli. Ako se ovakvo ponašanje ne uspijeva liječiti ili nositi se s njim, može biti vrlo opasno po oboljelog ili bliske osobe. Neka od opasnih ponašanja su:

- Samoozljeđivanje – može se pojaviti i kod blažih oblika autizma, ali postupci poput udaranja glavom i jedenja neprobavljivih objekata mnogo su češći kod težih oblika
- Agresivno i antisocijalno ponašanje – relativno rijetko kod autizma ali kod težih oblika se može pojaviti, zahtijeva brzu intervenciju
- Lutanje – često kod teških oblika autizma, velika mogućnost nailaska na opasne situacije

2.5. Mogućnost primjene mobilne računalne okoline kod autističnih pacijenata

Pametni mobilni uređaji koje većina ljudi danas ima mogu se koristiti kao uređaji za olakšanje svakodnevnog života, bila to jednostavna aplikacija za bilježenje zadataka ili nešto kompleksnije. Tehnologija predstavlja velik potencijal za pomoć osobama s autizmom. One mogu imati razne potrebe, od pomoći s komunikacijom do interaktivnog učenja i sličnog. Najčešće su ciljna skupina koja treba koristiti ove aplikacije djeca. Ona su najčešće zainteresirana za tehnologiju i lako ju uče koristiti. U kontekstu mobilnih uređaja načini na koje mogu pomoći djeci s autizmom su:

- Privlačenje pažnje – olakšava učenje
- Isprobavanje ideja, izvršavanje zadataka i učenje kroz pokušaje
- Mogućnost vježbanja vještina konzistentno i često
- Pomoć pri koncentraciji

Svako dijete s autizmom je drukčije, pa jedno rješenje neće obvezno pomoći svima. Nadalje, tehnička rješenja ne mogu pružiti prilike i mogućnosti učenja uz roditelje, učitelje i drugu djecu. Postoji rizik da će dijete zbog korištenja tehničkog rješenja zanemariti socijalnu interakciju i druge stvari iz svakodnevnog života. Vještine koje dijete nauči koristeći ovakva rješenja često ne zna primijeniti u svakodnevnom životu. Pri odabiru tehničkog rješenja za dijete treba razmisliti o njegovim potrebama, snagama i interesima. Također, dijete treba nadgledati tijekom korištenja rješenja. Za djecu ispod dvije godine ovakav pristup se ne preporuča. Bitno je posavjetovati se sa stručnjacima koji liječe dijete prije početka uporabe neke aplikacije za pomoć. Pošto su poteškoće s govorenjem i održavanjem razgovora jedan od najčešćih kod autizma, to je jedna od mogućnosti za koju mobilna okolina može pružiti pomoć i naveliko je istražena i razrađena. Koristi se AAC (*Augmentative and alternative communication*), tj. komunikacija kroz slaganje izraza pomoću sličica koje predstavljaju pojedini izraz. Softverskim rješenjima moguće je izvesti sustave s gotovo neograničenim brojem izraza, mogućnošću promjene izraza i dodavanja novih, njihovih izgovora itd. Prije su ovakvi sustavi bili glomazni i skupi, te im nije svatko imao pristup. Upravo ovdje je prednost mobilne okoline - dostupnost, prenosivost i lako korištenje. Dodirni zasloni današnjih pametnih mobilnih uređaja dodatno olakšavaju korištenje nekog sustava za pomoć osobama sa slabijim motoričkim sposobnostima [19]. Nadalje, pošto su osobe s autizmom više naklonjene „vizualnom načinu razmišljanja“ moguće je podatke koje im treba prenijeti prikazati u vizualnom obliku. Računalna grafika okupira i održava njihovu pažnju. Neki od oboljelih mogu biti osjetljivi

na prejake zvučne podražaje, što također možemo pratiti i upozoriti okolinu. Nadalje, neki pacijenti nemaju razumijevanja za slijed događaja i važnost obavljanja nekih radnji u određenom slijedu. Ne mogu se organizirati. Pomoću mobilne okoline može se zadatke podijeliti u korake te izraditi vizualnu reprezentaciju tih koraka. Nerijetko djeca s autizmom imaju poteškoća s obavljanjem školskih zadataka i učenja zbog slabijih motoričkih sposobnosti što otežava pisanje i crtanje. Korištenje virtualnih tipkovnica, dodirnog zaslona i sustava za pretvorbu govora u tekst čini učenje i obavljanje školskih zadataka lakšim i zabavnijim, te povećava produktivnost. Neki pojedinci nemaju dovoljno razvijen osjetilni sustav, pa ne mogu u isto vrijeme gledati i slušati nešto. Korištenjem mobilne okoline mogu postupno učiti koristiti oboje. Djeca mogu učiti čitati lakše uz vizualnu potporu, a nekima treba ona zvučna. Ukratko, mobilna okolina se može efektivno koristiti ne samo za zabavu i kao uređaj za pomoć pri govoru, nego i kao pomoć pri učenju, socijalnim, organizacijskim i vještinama potrebnim za samostalnost, obavljanju zadataka i usavršavanju motoričkih i osjetilnih sposobnosti.

2.6. Postojeći računalni alati

Postoje mnoge aplikacije za djecu s autizmom i njihove roditelje. Većina njih pomaže sa socijalnim, motoričkim, komunikacijskim, akademskim i drugim vještinama. U nastavku će biti opisani neki postojeći alati i njihove prednosti i mane.

2.6.1. MOSCOCO

MOSCOCO ili punim nazivom *Mobile Social Compass* jest mobilna aplikacija za pomoć djeci oboljeloj od autizma. Ona koristi proširenu stvarnost (AR) i vizualnu podršku kako bi pomogla djeci da vježbaju socijalne vještine u situacijama svakodnevnog života. Prema istraživanju provedenom od tvoraca aplikacije dokazano je da ona uistinu pomaže djeci pri vježbi i učenju socijalnih vještina, povećava kvalitetu i kvantitetu socijalnih interakcija, smanjuje broj neprimjerenih interakcija i ponašanja te pomaži djeci s integracijom u društvo. Aplikacija koristi principe opisane u kurikulum *The Social Compass*. To je kurikulum za edukaciju i ponašanje, te se bazira na šest osnovnih lekcija koje uče djecu kompletnoj socijalnoj interakciji:

- Direktan kontakt očima
- Prostor i blizina
- Započinjanje interakcije
- Postavljanje pitanja
- Dijeljenje interesa

- Završetak interakcije

Svaka lekcija sastavljena je od osam koraka i provodi se u grupama djece s autizmom i bez njega. Na početku se utvrđuje koja socijalna vještina će se učiti, te vizualna potpora za nju. Nadalje iznosi se socijalna priča koja pomaže djetetu povezati socijalnu situaciju, vještinu i vizualnu potporu. Demonstrira se kako se socijalna vještina koristi u situacijama te djeca vježbaju takve situacije u parovima. Nakon toga djeca igraju uloge u raznim situacijama, pa sami ocjenjuju svoj nastup. Na kraju djeca se ohrabruju da koriste naučenu socijalnu vještinu u raznim situacijama. Konkretna aplikacija radi na Android mobilnim uređajima te koristi kameru kako bi proširio stvarnost nekim svakodnevnim socijalnim situacijama. Aplikacija ima načine rada i za grupe i individualne osobe. Kod individualnog načina rada svaki korisnik ima jedinstvenu identifikaciju, a napredak se prati dodjeljivanjem zvjezdica. Sadrži formu za samoocjenjivanje te naputke za socijalne interakcije. Grupni način rada sadrži vizualni raspored interakcija, prikaz drugih korisnika s kojima je moguće komunicirati te onih koji bi mogli biti prikladni za socijalnu interakciju. Aplikacija predlaže druge korisnike za početak interakcije pokazujući njihovu fotografiju i neke podatke. Kada su djeca koja koriste aplikaciju dovoljno blizu automatski će se povezati i vježba će započeti. Vizualni rasporedi interakcija im se sinkroniziraju te se pokreće pomoć, tj. naputci za interakciju. Cijela interakcija se prati i podaci se pohranjuju za daljnju analizu. Nakon završetka pojedinog koraka interakcije korisnik se ocjenjuje pozitivnom ili negativnom ocjenom. Ako oba korisnika pozitivno ocjene interakciju zarađuju bodove u obliku zvjezdica, što potiče zajednički rad pri savladavanju socijalnih vještina. Grafičko sučelje aplikacije vidljivo je na slici 2.8.

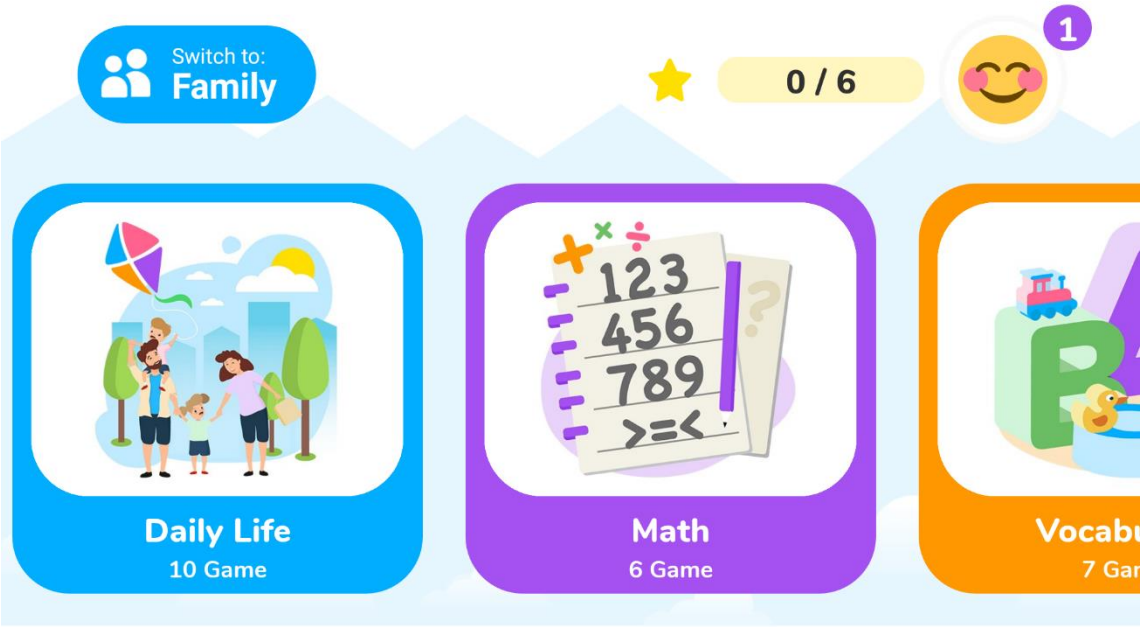


Slika 2.8. Grafičko sučelje MOSCOCO-a

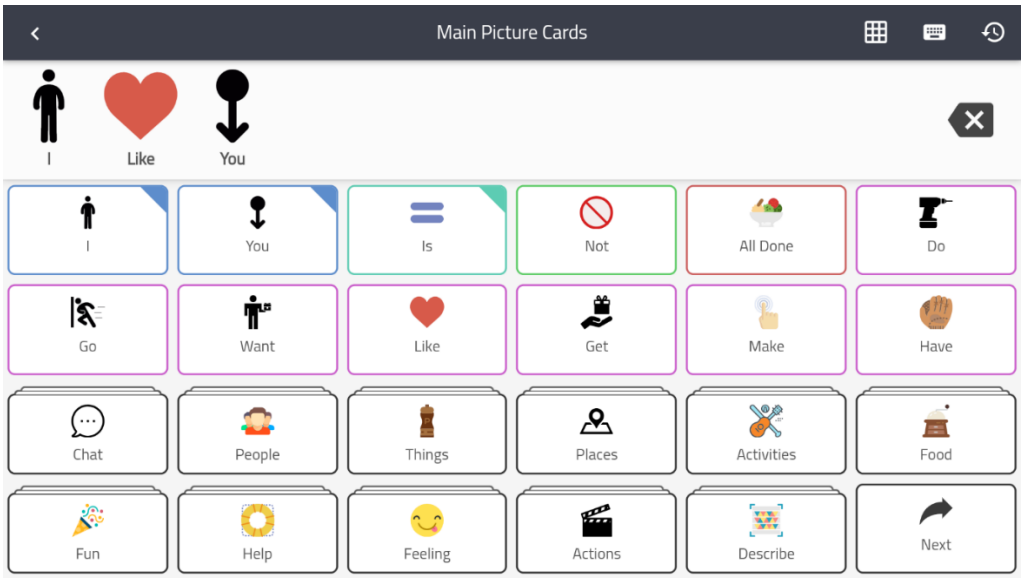
Aplikacija je jednostavna za korištenje i uistinu pomaže djeci oboljeloj od autizma unaprijediti svoje socijalne vještine. Umjesto odvlačenja od socijalne interakcije poput mnogih ostalih aplikacija ona motivira djecu na nju. Sustav bodovanja potiče i na timski rad, a naputci za interakciju mogu pomoći pri održavanju razgovora. Naime, ti naputci nekad mogu biti nerelevantni jer aplikacija nije svjesna o temi razgovora. Aplikacija koristi samo jednu od metoda za poboljšanje socijalne interakcije, što je svakako prostor za napredak.

2.6.2. Otismo

Otsimo je mobilna aplikacija za pomoć u razvoju djece s autizmom. Dostupna je na Android i iOS platformama besplatno. Djeca mogu sama koristiti aplikaciju bez ikakvih instrukcija i koristiti prednosti prethodno objašnjene ABA terapije u svom domu. Pomoću umjetne inteligencije aplikacija se prilagođava mentalnom stanju djeteta. Aplikacija sadrži preko pedeset igara koje pomažu mentalnom, socijalnom i motoričkom razvoju djeteta te omogućava lakšu komunikaciju pomoću AAC metode. Neke od igara koje sadrži su sparivanje raznih predmeta i pojava iz okoline s njihovim nazivima i značenjima, učenje boja, matematike, životinja i sličnog. Moguće je i podesiti težinu ovisno o starosti djeteta. Sustav je verificiran i validiran od strane nekoliko nadležnih tijela i korisnici dijele vrlo pozitivna iskustva. Razvijen je pod vodstvom dječjih psihologa i edukatora. Glavni cilj aplikacije je naučiti djecu osnovna znanja i pripremiti ih za daljnju edukaciju. Aplikacija je odvojena u sekcije za djecu i za obitelj. Dijete neprekidno može učiti bez smetnji u svojoj sekciji, a ostatak obitelji može pratiti njegov napredak, dodavati i prilagođavati igre i baviti se ostalim funkcijama aplikacije. Na slikama 2.9. i 2.10. vidljivi su neki ključni elementi grafičkog sučelja Otsimo sustava.



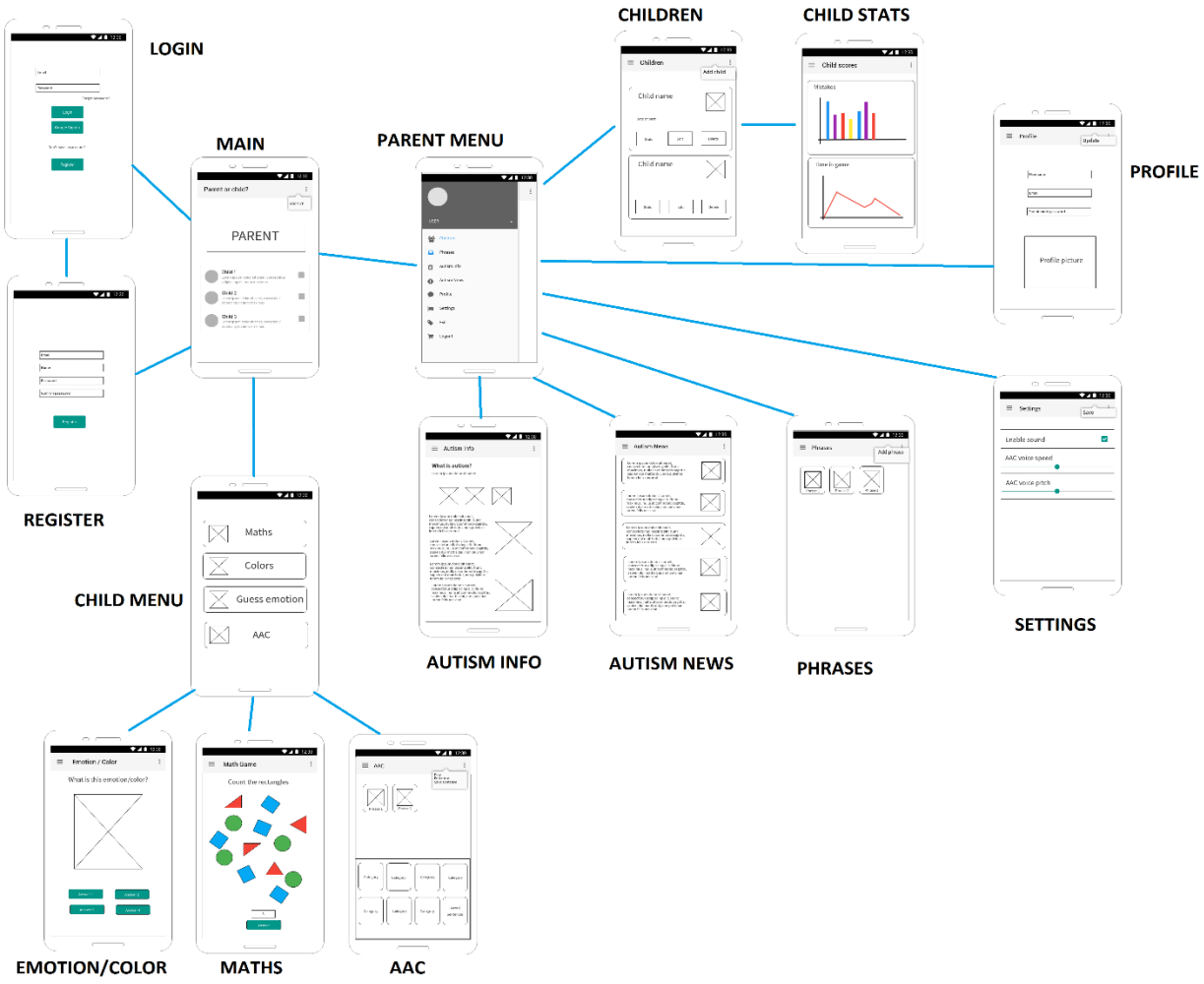
Slika 2.9. Kategorije igara Otsimo aplikacije



Slika 2.10. AAC sustav Otsimo aplikacije

3. PROGRAMSKO RJEŠENJE ZA PODRŠKU PACIJENTIMA OBOLJELIM OD AUTIZMA

Pregledom literature, istraživanja i postojećih rješenja donesena je odluka za implementacijom programskog rješenja koje će sadržavati elemente ABA terapije te samostalno izveden AAC sustav. Kako bi se korisnički podaci sačuvali bit će omogućena registracija korisničkog računa te sinkronizacija tih podataka. Za korištenje aplikacije korisnik će morati biti prijavljen. Aplikacija će se podijeliti u načine rada za roditelje i za djecu. Za pristup načinu rada za roditelje bit će potrebna autentifikacija tako da mu djeca ne mogu pristupiti i mijenjati postavke. Biti će moguće autentificirati se pomoću otiska prsta i samostalno postavljene lozinke. Na korisnički profil roditelji će dodavati dječje profile preko kojih će djeca koristiti aplikaciju i pratit će im se statistika. Podaci će se spremati lokalno i online. Pratit će se koliko vremena je dijete provelo u igri te koliko puta je dalo pogrešni odgovor u igrama pogađanja. Statistika će se prikazati grafički. Naknadno će se moći uređivati dječje profile i brisati ih. Implementirat će se i pregled, dodavanje i brisanje fraza za AAC sustav. Roditelji će moći čitati novosti povezane s autizmom i informirati se o njemu s jedne od vodećih internetskih stranica o autizmu. Na svom profilu roditelji će ažurirati svoje podatke te učitavati profilnu sliku, a na prikazu s postavkama postaviti postavke vezane za AAC sustav, zvukove aplikacije i slično. Kada je korisnik prijavljen, prikazat će se glavni izbornik za način rada za roditelje ili djecu. Dijete će moći odabrati svoje ime na popisu te početi koristiti aplikaciju. Moći će birati između igara brojanja, pogađanja boja i emocija te AAC sustava. Pri igri brojanja generirat će se nasumičan broj različitih oblika raznolikih boja te će biti potrebno prebrojati određene oblike. Kod pogađanja boja bit će potrebno odgovoriti koja je boja prikazana, a kod emocija trebat će emociju ljudi na slikama spariti sa sličicama koje odgovaraju toj emociji. Koristit će se šest osnovnih emocija – tuga, strah, iznenađenje, sreća, uzbuđenost i ljutnja. AAC sustav prikazivat će izbornik kategorija fraza te spremljenih složenih rečenica iz kojih će se odabirati fraze i dodavati na prikaz. Složena rečenica moći će se reproducirati i spremiti, a bit će moguće i izravno upisati željeni tekst. Fraze će se prikazivati pomoću opisnih ikona te teksta. Na slici 3.1. prikazan je *wireframe* aplikacije izveden prije same implementacije.



Slika 3.1. *Wireframe* aplikacije

Ulaz u aplikaciju jest *Login* prikaz. Korisnik se može prijaviti sa svojim podacima ili pomoću *Google Sign In* opcije pritiskom na određene elemente. Ako se aplikaciji prvi put pristupa nekim Google računom automatski se kreira korisnički račun. Pritiskom na *Forgot Password* opciju omogućit će se povratak zaboravljene lozinke, a *Register* vodi na prikaz za registraciju. Za registraciju bit će potrebno unijeti neke osobne podatke te dodati dječje profile. Nakon uspješnog unosa otvara se glavni prikaz sa odabirom načina rada za roditelje i djecu. *Login* prikaz preskočit će se ako je korisnik već prijavljen. Nadalje odabirom načina rada za roditelje otvorit će se prikaz *Parent Menu* pri uspješnoj autentifikaciji gdje će se moći izvesti sve akcije povezane s roditeljima prekog kliznog izbornika. Na *Children* prikazu moć će se vidjeti popis dječjih profila te ih brisati i uređivati. Pritiskom na *Stats* otvorit će se *Child Stats* prikaz gdje će biti vidljiva grafički prikazana statistika dječjeg profila. Na *Profile* i *Settings* prikazima postavljat će se postavke profila i

aplikacije, a *Autism News* i *Autism Info* pružit će roditeljima mogućnost informiranja o samom autizmu te praćenja novosti o njemu. Pritiskajući *Phrases* otvorit će se ekran sa popisom svih dostupnih fraza za AAC sustav te će ih se moći dodati još personaliziranih. Odabirom dječjeg profila s popisa na *Main* prikazu otvorit će se izbornik na kojem će se odabrati igra koja se želi pokrenuti ili AAC sustav. Kod igara s emocijama i bojama korisniku će biti prikazana slika emocije ili boje te ponuđeno četiri odgovora. Kod emocija mogući odgovori bit će prikazani sličicama koje odgovaraju ponuđenim emocijama. Kod igre prebrojavanja generirat će se nasumičan broj različitih geometrijskih oblika i trebat će prebrojati jedan određeni oblik te upisati odgovor u polje za unos. *AAC* prikaz bit će podijeljen u izbornik fraza na donjem i prikaz složene rečenice na gornjem dijelu ekrana. Unutar izbornika fraza bit će prvo ponuđene kategorije za lakše snalaženje te već spremljene složene rečenice.

3.1. Zahtjevi na sustav

Prije izvedbe aplikacije postavljeni su zahtjevi na sustav koje aplikacija mora zadovoljavati da bi se smatrala uspješnom implementacijom ideje i cilja koji je postavljen. Prvotno je odabrana platforma na kojoj aplikacija mora funkcionirati. Odabran je Android operacijski sustav zbog prijašnjeg iskustva u razvoju Android aplikacija, dostupnih besplatnih alata, lakog testiranja i raspodjele korisnika mobilnih uređaja u veliku korist Androida sustava. Aplikacija mora raditi na verzijama Android operacijskog sustava iznad 5.0 *Lollipop* koje su najzastupljenije kod korisnika. Zahtjevi na resurse uređaja nisu veliki zbog relativno male kompleksnosti aplikacije u usporedbi s mogućnostima današnjih mobilnih uređaja. Uređaj mora imati aktivnu internetsku vezu radi sinkronizacije korisničkih podataka i sadržaja aplikacije. Aplikacija neće zahtijevati nikakve posebne dozvole koje joj korisnik treba dati. Korisničko sučelje treba biti prilagodljivo različitim rezolucijama i omjerima ekrana uređaja.

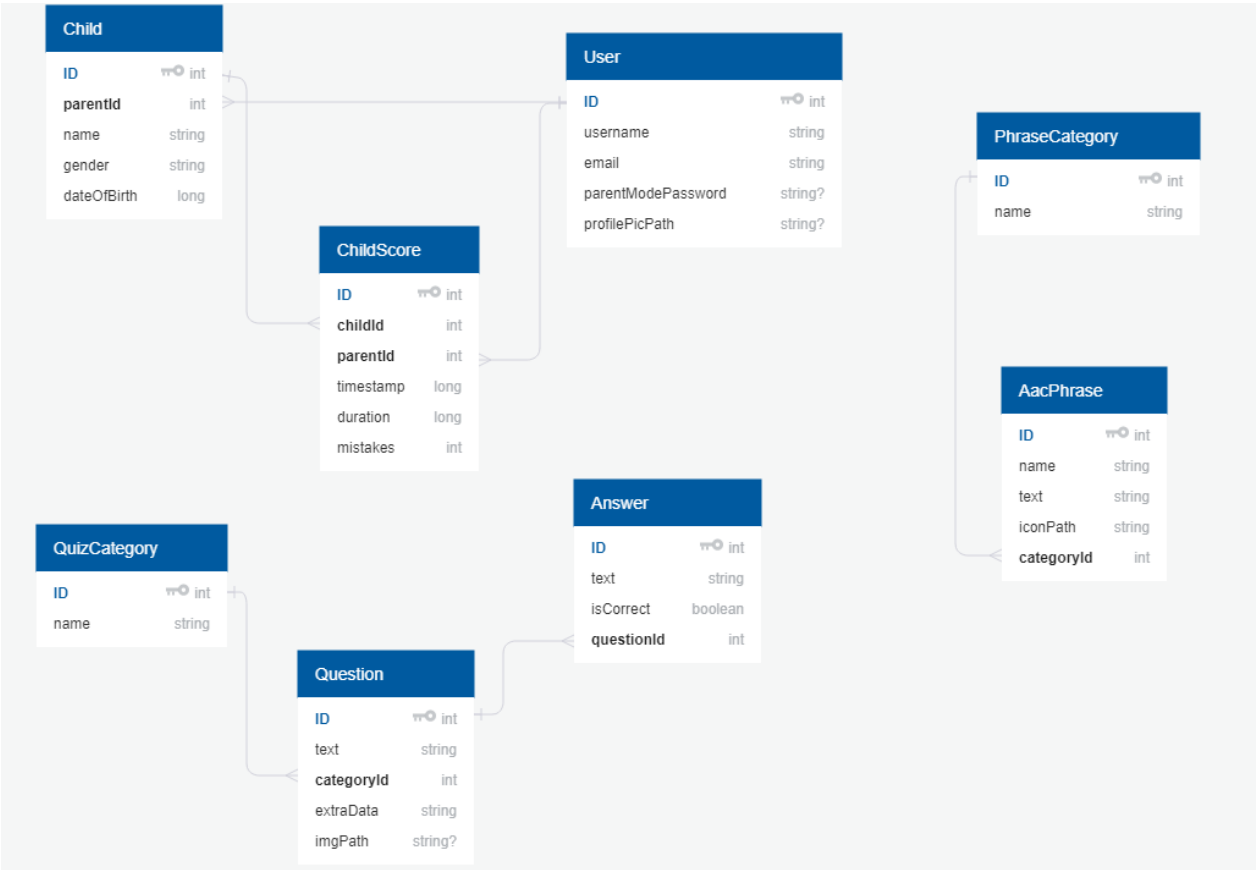
Što se tiče zahtjeva na funkcionalnosti i karakteristika aplikacije potrebno je izvesti slijedeće:

- Pomoć djeci s autizmom u svakodnevnom životu AAC sustavom
- Pomoć pri njihovom mentalnom razvoju elementima ABA terapije
- Privlačan sadržaj i sučelje za poticanje korištenja aplikacije
- Spremanje često korištenih rečenica složenih od fraza
- Unos teksta za fraze i izgovor direktno
- Mogućnost roditeljima da dodaju sadržaj za AAC sustav
- Kreiranje korisničkog profila
- Sinkronizacija korisničkih podataka u lokalnoj i online bazi podataka

- Korištenje na više uređaja bez gubitka podataka
- Mogućnost dodavanja više dječjih profila
- Praćenje i grafički prikaz napretka djeteta
- Autentifikacija za korištenje načina rada za roditelje
- Proširivost sadržaja igara, mogućnost dodavanja na poslužitelju
- Postavljanje korisničkih postavki
- Informiranje roditelja o autizmu i novostima o njemu

3.2. Model podataka

Baza podataka korištena u aplikaciji sastoji se od 8 međusobno povezanih relacijskih tablica. Za lokalnu bazu podataka korištena je *SQLite* baza podataka ugrađena u Android operacijski sustav. Na backend dijelu aplikacije zbog restrikcija sustava koristi se *NoSQL* baza podataka gdje nisu potrebni strani ključevi za relacije, ali upisuju se zbog lakše sinkronizacije s lokalnom *SQLite* bazom. U centru baze se nalazi tablica *User* koja sadrži podatke o korisniku aplikacije, tj. roditelju. Pratit će se korisničko ime, email, lozinka za načina rada za roditelje i putanja do profilne slike. Lozinka za način rada za roditelje i putanja profilne slike isprva ne moraju biti postavljeni. Nadalje od modela koristi se *Child* koji predstavlja dječji profil te se s roditeljem povezuje preko stranog ključa *parentId*. Jedan korisnički profil može imati više dječjih profila. Podaci koje još sadrži su ime djeteta te njegov spol i datum rođenja. S ovim modelom, kao i s *User* modelom povezan je stranim ključem *ChildScore* kojim se prati statistika svakog igranja neke od igara. Pohranjuju se podaci koliko dugo je igra bila pokrenuta, koliko je dijete pogrešaka učinilo pri igri te točno vrijeme kada je igra završena. Dječji profil sadrži više *ChildScore* unosa, a svaki *ChildScore* unos odnosi se samo na jedan dječji profil. Za fraze korištene u AAC sustavu postoje dva modela. *PhraseCategory* služi za razvrstavanje fraza po kategorijama te ima samo primarni ključ ID i naziv kategorije. Stranim ključem *categoryId* kategorije su povezane s tablicom fraza *AacPhrase* u kojoj vezi jedna kategorija sadrži više fraza. Fraza ima svoj naziv, tekst koji se prikazuje i izgovara te putanju do ikonice koja ju predstavlja. Za igre pogađanja koriste se modeli *QuizCategory, Question* i *Answer*. Svaka kategorija sadrži više pitanja, a svako pitanje sadrži po četiri moguća odgovora. Kategorije imaju samo primarni ključ i ime, a pitanja imaju svoj tekst, putanju do slike koju treba prikazati ako postoji te dodatno polje za eventualne potrebne podatke, npr. heksadecimalni broj koji predstavlja boju u igri pogađanja boje. Model podataka i relacije tablica prikazani su na slici 3.2.



Slika 3.2. Model baze podataka

3.3. Korišteni alati i tehnologije

U ovom poglavlju objasnit će se alati i tehnologije potrebni za implementaciju sustava. Opisat će se Android operacijski sustav, Android Studio programsko okruženje, službeni jezik za Android razvoj Kotlin i neki važniji programski okviri, platforme i biblioteke bitne za rad sustava. Od neopisanih alata vrijedne spomena su biblioteke Retrofit2 i OkHttp za sve mrežne operacije te RxKotlin za asinkrono dohvaćanje podataka s mreže preko Retrofita i baze podataka preko Rooma, kombinirajući ih, mapirajući i prosljeđujući u ViewModel. Nadalje, koristi se Dagger2 za ubrizgavanje ovisnosti radi povećanja testabilnosti aplikacije.

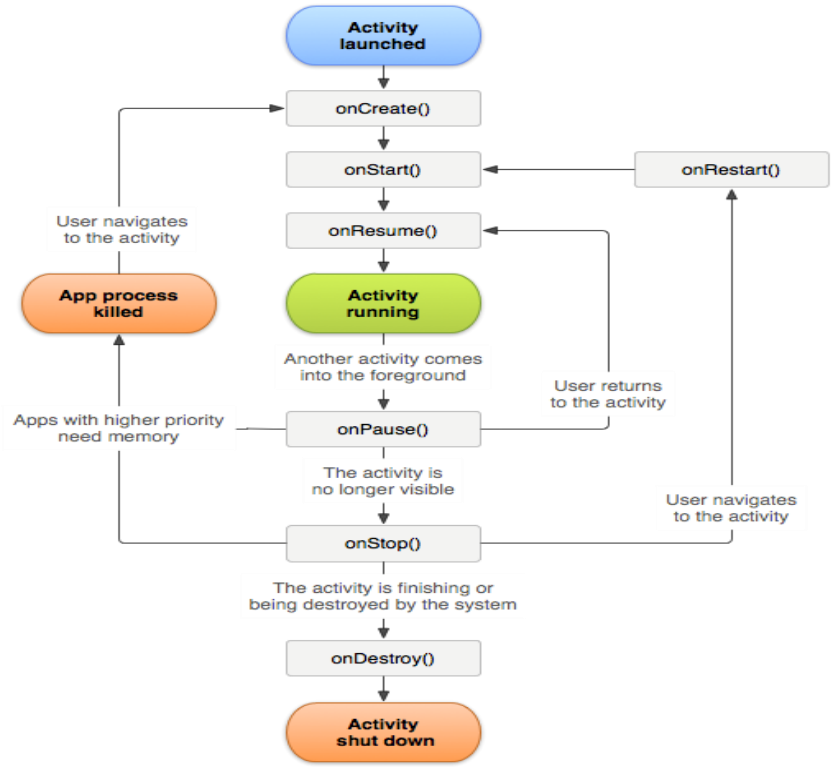
3.3.1. Android OS i Android Studio

Android OS je operacijski sustav namijenjen pametnim mobilnim uređajima. Trenutno ga izdaje i održava Google Inc., ali začet je kao proizvod zasebne kompanije Android Inc. On je potpuno prilagodljiv svim korisničkim zahtjevima i lako ga je pokretati na raznim uređajima. Android je Linux distribucija razvijena za ARM procesore. Otvorenost programskog koda Android sustava olakšava razvoj aplikacija, pa su dostupna mnogobrojna programska rješenja, što privlači

i zadržava korisnike. Trenutno najrasprostranjeniji mobilni operativni sustav na svijetu. Android nastoji najbolje iskoristiti sve značajke uređaja na kojem se pokreće. Svaka aplikacija može pristupiti hardveru uređaja poput GPS-a i kamere što znatno olakšava njihovo upravljanje. Sustav se može prilagoditi zahtjevima svakog korisnika. Za prikaz 2D i 3D grafike koristi grafičke biblioteke zasnovane na OpenGL ES 2.0 specifikacijama. Za lokalnu bazu podataka koristi se SQLite. Uređaji se s povezuju i prenose informacije i datoteke preko UMTS, Bluetooth, Wi-Fi, LTE i drugih tehnologija. Podržava ekran osjetljiv na dodir, GPS, akcelerometar, grafičku 3D akceleraciju i multi-touch. Za pokretanje aplikacija na uređaju koristi se *Android Runtime* temeljen na *Dalvik* virtualnom stroju. Sučelje sustava je temeljeno na izravnom upravljanju, gdje korisnik dodirima i gestama upravlja uređajem i unosi podatke. Android podržava višezadaćnost. Aplikacije se preuzimaju sa servisa *Google Play.* Android uređaji napajani su baterijom pa je važno optimizirati potrošnju električne energije. Android operativni sustav podijeljen je u pet slojeva. Svaki sloj sadrži više programa, dijelova podrške, pogonskih programa i biblioteka. Linux jezgra upravlja memorijom i napajanjem, sadrži sigurnosne postavke, ima podršku za dijeljene biblioteke, mrežni stog i ostale pogonske programe. Nadalje bitna značajka je sloj za apstrakciju hardvera (HAL) koji daje aplikacijama direktan i apstraktan pristup hardveru. Aplikacije ga mogu jednostavno pozvati i upravljati njime. U idućem sloju nalaze se sve biblioteke nužne za rad aplikacija, *Android Runtime* (ART) i Dalvik virtualni stroj. Svaka pokrenuta aplikacija izvodi se u zasebnoj instanci virtualnog stroja. Iznad se nalazi sloj aplikacijskog okvira pružajući servise više razine u obliku Java klasa. Aplikacije se izrađuju korištenjem tih servisa. Najviši sloj je aplikacijski. Korisnici vrše interakciju s uređajem na ovom sloju.

Android aplikacije mogu sadržavati četiri vrste elemenata: *activity, service, content provider i broadcast reciever*. Svaka Android aplikacija ima mogućnost pokretanja komponenti drugih aplikacija. Aktivnost (engl. *activity*) označava svaki pojedini prikaz koji sadrži neko korisničko sučelje. Aplikacija sadrži više aktivnosti u kojima se nalazi logika za upravljanje sučeljem i podacima. Aktivnosti se u memoriju slažu kao stog, te kad se pozove nova postavlja se na vrh tog stoga. Posljednja korištena aktivnost ostaje u stogu ako nije eksplicitno zaustavljena. Aktivnosti imaju četiri stanja: aktivne, pauzirane, zaustavljene i ugašene. Ako je aktivnost trenutno vidljiva na zaslonu tada je aktivna. Kada se preko nje pojavi druga transparentna ili aktivnost nepune veličine, pauzirana je. Informacije koje sadrži su sačuvane, a brišu se samo ako je sistemska memorija na vrlo niskoj razini i treba se očistiti. Ako se aktivnost na punom zaslonu prikaže preko druge, druga je zaustavljena. Potpuno gašenje aktivnosti se događa kada ona neko vrijeme provodi zaustavljena ili pauzirana. Servis (engl. *service*) služi za pokretanje u pozadini i

izvođenje dugotrajnih operacija. On nema korisničko sučelje, a pokreću ga druge komponente aplikacije. Pružatelj sadržaja (engl. *content provider*) upravlja pristupom dijeljenom nizu aplikacijskih podataka. Ako aplikacija želi koristiti i izmjenjivati podatke mora to čini pomoću njega. Prijemnik emitiranja (engl. *broadcast receiver*) je komponenta koja reagira na obavijesti emitirane kroz cijeli sustav. Nema korisničko sučelje kao ni servis. Na slici 3.3. prikazan je „životni ciklus“ aktivnosti.



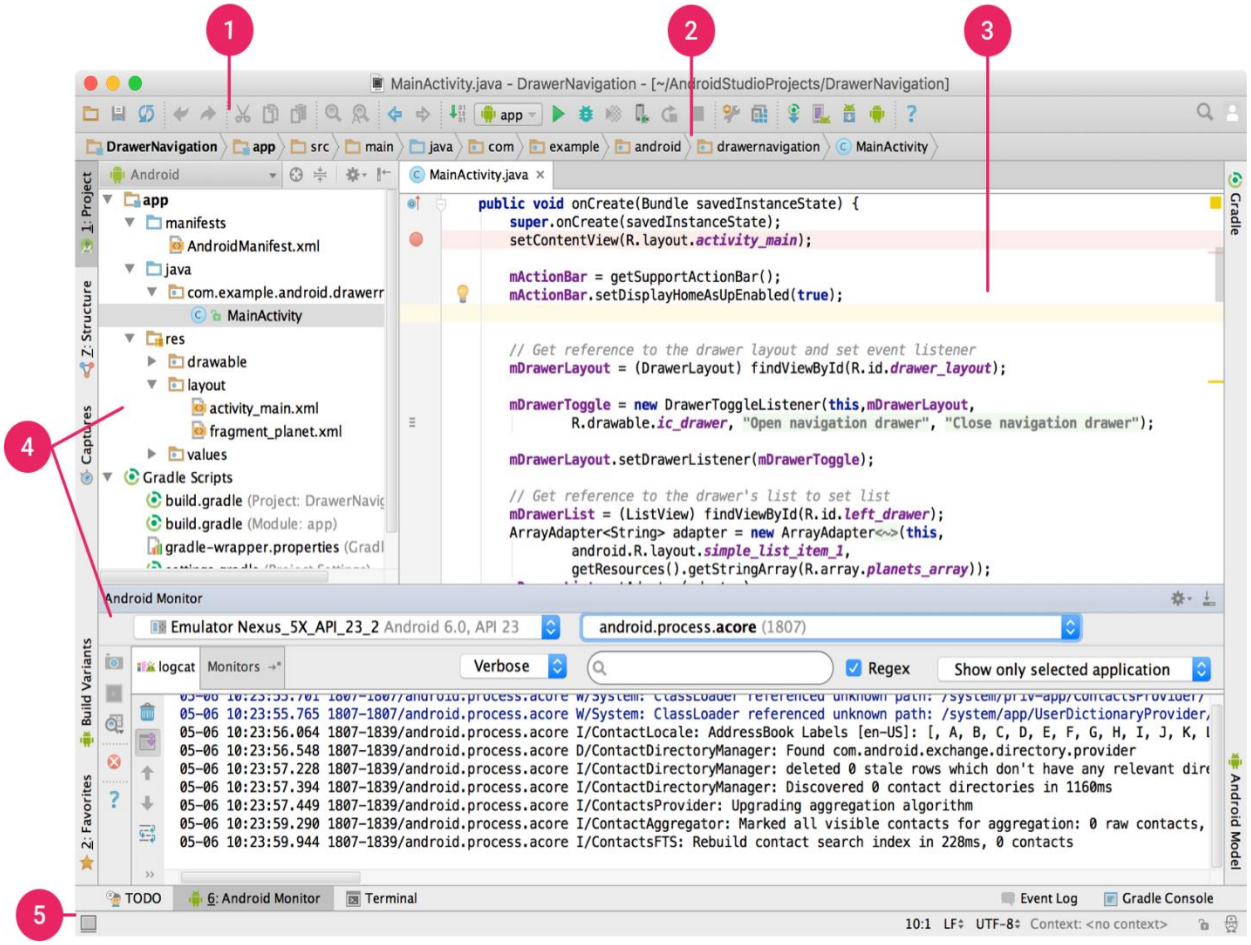
Slika 3.3. Životni ciklus Android aktivnosti [20]

Android Studio je integrirano razvojno okruženje (IDE) za razvoj Android aplikacija. Temelji se na IntelliJ IDEA razvojnom okruženju. Besplatan je i može se preuzeti s Android web stranica. Za kompilaciju se koristi Grade *build system*. Integriran je emulator Android uređaja. Za početak rada potrebno je instalirati i Android SDK te Java razvojne alate. Aplikacije se izrađuju u Java i Kotlin programskim jezicima, a njihovo sučelje se kreira u uređivaču teksta zasnovanom na XML jeziku. Pri izradi rasporeda elemenata sučelja na zaslonu se prikazuje pregled izgleda aplikacije. Kreiranje češćih elemenata koda olakšano je predlošcima. Projekti se sastoje od manifest datoteke s informacijama što sve aplikacija sadrži i zahtijeva, Java klasa te svih resursa koji nisu Java kod poput XML rasporeda elemenata sučelja, resursa i drugog. Testiranje napisanog koda izvodi se pokretanjem u ugrađenom emulatoru ili na fizičkom uređaju. Koristeći Gradle *build system* izmjenjuje se, konfigurira i proširuje proces kompilacije, tj. stvaranja izvršne datoteke. Iz

softvera je moguće stvoriti APK instalacijsku datoteku za direktnu instalaciju na uređaj ili učitavanje na *Google Play* servis. Također, Android Studio sadrži alate za otklanjanje pogrešaka i praćenje korištenja resursa uređaja.

Dijelovi korisničkog sučelja vidljivi su na slici 3.4. i označeni:

1. Alatne traka
2. Navigacijska traka
3. Prozor editora
4. Alatni prozor
5. Statusna traka



Slika 3.4. Korisničko sučelje Android Studio IDE-a

3.3.2. Kotlin

Kotlin je objektno orijentirani programski jezik razvijen od strane JetBrains, kompanije koja je razvila i IntelliJ IDEA razvojno okruženje na kojem je zasnovan Android Studio. Službeni je jezik za razvoj Android aplikacija. Predstavlja na neki način zamjenu za Java programski jezik

pojednostavljujući sintaksu i olakšavajući neke standardne često izvođene operacije. Ovako povećava čitljivost koda, ubrzava razvoj softvera, te smanjuje mogućnost nekih čestih slučajnih pogrešaka poput *NullPointerException* i drugih. Iako je objektno orijentiran jezik, ima mnogo karakteristika funkcionalnog programskog jezika što se vidi po ekstenzivnoj uporabi *lambda* funkcija. Vrlo je ekspresivan te možemo postići više s manje koda. Najčešći programski obrasci pokriveni su u samoj sintaksi. Na prikazu koda 3.3.1. vidljiva je razlika u definiranju najjednostavnije model klase u Kotlinu i Javi.

JAVA:

```java
public class Artist {
   private long id;
   private String name;

   public Artist(long id, String name) {
        this.id = id;
        this.name = name;
   }

   public long getId() {
        return id;
   }

   public void setId(long id) {
        this.id = id;
   }

   public String getName() {
        return name;
   }

   public void setName(String name) {
        this.name = name;
   }
}
```

KOTLIN:

```kotlin
data class Artist(
      var id: Long,
      var name: String,
      var url: String
)
```

Kod 3.3.1. Usporedba model klase u Javi i Kotlinu

Vidljive su i razlike pri deklariranju varijabli. Koriste se *var* za varijable i *val* za konstante. Tip varijable se može izostaviti ako je kompajler sposoban prepoznati ga iz inicijalizacije. Ne postoje primitivi, već su sve varijable objekti. Od Jave se razlikuje i deklaracija funkcija:

```kotlin
fun doSomething(thing: String): String {
   return "$thing done"
}

fun doSomethingShort(thing: String) = "$thing done"
```

Kod 3.3.2. Deklaracija funkcije

Koristi se ključna riječ *fun*, navodi njeno ime, parametriu zagradi te tip koji funkcija vraća iza dvotočke. Ako ne vraća ništa koristi se *Unit*. Moguć je i skraćeni zapis ako se funkcija sastoji od samo jedne linije, tj. izraza. Kao i Java, Kotlin se pretvara u Java Bytecode pokretan u JVM virtualnom stroju i potpuno je interoperabilan s njom te je moguće koristiti skoro sve biblioteke već razvijene za Javu. Vrlo ga je lako naučiti uz već postojeće znanje Jave. Jedna od važnih značajki Kotlina za spomenuti je *null safety*. Kao glavni uzrok rušenja Java aplikacija pokazuje se *runtime* pogreška *NullPointerException*, tj. korištenje objekta koji još nije inicijaliziran i ima vrijednost *null*. Objekti u Kotlinu ovu vrijednost neće uopće moći poprimiti ako kao njegov tip ne navedemo željeni tip s ovom mogućnošću, tj. *optional*. Npr. ako se želi da *String* varijabla može poprimiti i *null* vrijednost deklarirat ćemo je kao *String?*. Kotlin će ovakve pogreške prepoznati već pri kompilaciji ako nije eksplicitno navedeno da se želi koristiti objekt i ako nije inicijaliziran !! operatorom (*non-null assertion*). Uvodi se nekoliko operatora koji olakšavaju *null check* operacije. Ako se poziva samo jedna funkcijana objektu koristi se *safe call* operator dodajući znak ?, a ako ih se poziva više koristi se i *let* operator koji u funkciju prosljeđuje objekt na kojem je pozvana pod nazivom *it* ako nije specificirano drukčije ime. Pozivi izgledaju ovako:

```
private var string1: String? = null

string1?.toDouble()

string1?.let {
    it.length
    it.compareTo("other")
    it.hashCode()
}
```

Kod 3.3.3. Primjer *safe call* i *let* operatora

Nadalje jedna od bitnih značajki je mogućnost deklariranja i prosljeđivanja funkcija kao varijabli. Ovime se može izbjeći korištenje *interfacea* za komunikaciju između klasa, deklarirati neke operatore i prosljeđivati ih u generičku funkciju za računanje i slično. U kodu 3.3.3. vidljiva je deklaracija funkcije koja prima neki broj i vraća njegovu duplu vrijednost te njeno prosljeđivanje i korištenje u drugoj funkciji. Funkcija prima i vraća *Integer* vrijednost.

```
val doubleValue: (Int) -> Int = { a -> a*2 }

fun calculate(value: Int, operator: (Int) -> Int) = operator(value)

val result = calculate(6, doubleValue) // result = 12
```

Kod 3.3.4. Korištenje funkcije kao varijable

Implementirana je mogućnost proširivanja već postojećih klasa funkcijama pod imenom *extension functions*. Te dodatne funkcije definiraju se u zasebnoj datoteci, a ponašaju se kao da su deklarirane u samoj klasi za koju su napisane. Primjer je dan za u kodu 3.3.5. Deklarira se funkcija *repeat* na *String* klasi uzimajući njenu vrijednost i vraćajući je dva puta. Vrijednosti se pristupa ključnom riječi *this*, identično deklaraciji funkcije unutar same klase.

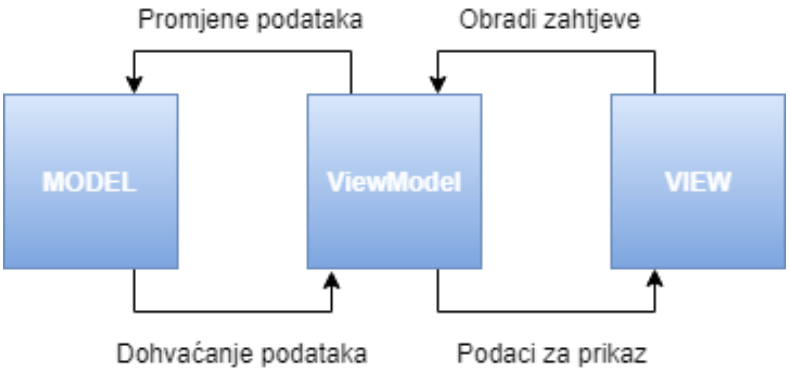
```
fun String.repeat(): String {
   return StringBuilder().append(this).append(this).toString()
}
```

Kod 3.3.5. *Extension function* na klasi *String*

3.3.3. MVVM

Do nedavno kod razvoja Android aplikacija nije postojao službeni arhitekturni obrazac što je rezultiralo velikim razlikama u načinu postavljanja arhitekture aplikacije. Nastavak rada na projektu drugog razvojnog programera bio je mukotrpan, a testabilnost aplikacija varijabilna. Kreirane su mnoge biblioteke i obrasci, ali nedostatak službene podrške i dalje je predstavljao raskol. Radi testabilnosti većinom se slijedio MVP (*Model-View-Presenter*) obrazac, ali i on je imao svoje mane. On odvaja *business* logiku u Model, kod za upravljanje prikazima u View, a Presenter je poveznica između njih. Iako povećava testabilnost koda ima svoje mane, poput gubitka reference Presentera na View kod promjene konfiguracije, tj. orijentacije, jezika sustava i sličnog. Iz ovog razloga prelazi se na MVVM. Sastavnice su mu Model, View i ViewModel. Model i View obavljaju istu ulogu kao kod MVP obrasca, a ViewModel je njihova poveznica. Zajedno s *Modelom* često se koristi *repository* obrazac. Model dobavlja podatke iz raznih izvora podataka te ih šalje u ViewModel koji ih obrađuje i šalje podatak za prikaz u View. Proces može biti i obrnut gdje na primjer korisnička interakcija obavještava ViewModel da izvrši promjene nad podacima u Modelu. Ilustracija ovog procesa dana je na slici 3.5. Razlika između MVVM-a i MVP-a je u tome što ViewModel ne drži referencu na View i ima vlastiti životni ciklus, a podatke mu prenosi preko *Observera* koji se pretplaćuju na podatke u njemu. Važna značajka ViewModela

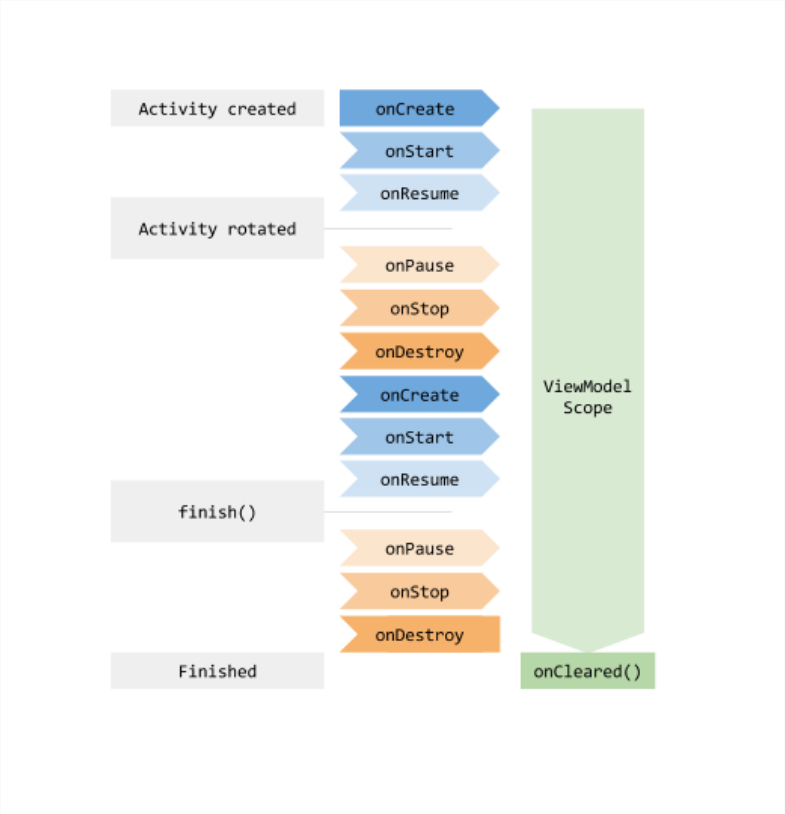
jest to što mu životni ciklus ne ovisi o promjenama konfiguracije kao kod *Activity* klasa, pa se podaci čuvaju i tijekom njih. Google je u Android SDK dodao sve potrebne alate za MVVM arhitekturu i preporuča se njihovo korištenje unutar skupa biblioteka Architecture Components koji će biti opisan u idućem potpoglavlju. MVVM arhitektura je odabrana zbog modularnosti, testabilnosti i službene podrške.



Slika 3.5. Ilustracija interakcije komponenti MVVM arhitekture

3.3.4. Architecture Components

Architecture Components je skupina Android biblioteka koja sadrži alate ključne za razvoj modularnih, testabilnih i održivih aplikacija. U nastavku će se kao pojedine komponente opisati ViewModel, LiveData i Room koji su korišteni pri implementaciji programskog rješenja. Prvotno, ViewModel je izvedenica ViewModel komponente MVVM arhitekture koja ima mogućnost čuvanja podataka i nakon promjene konfiguracija te pružanje podataka View komponenti, u kontekstu Androida aktivnosti ili fragmentu. Dizajniran je tako da nadživi životni ciklus komponente koja ga instancira, što je za aktivnost vidljivo na slici 3.6. Pri obavljanju dužih asinkronih zadataka može doći do gubitka aktivnosti iz memorije. Ako se to dogodi, ViewModel će pričekati da se aktivnost opet pokrene i poveže s njim, i tek onda mu poslati dobivene podatke preko promatrača registriranog u Activityu. ViewModel ne smije držati reference na *View* ili klase koje imaju životni ciklus. Instancira se posebno za svakuaktivnost, a među fragmentima se može dijeliti ViewModel aktivnosti koja ih sadrži. Želi li se kreirati prilagođena klasa koja nasljeđuje ViewModel, a konstruktor joj sadrži dodatne parametre, mora se kreirati i posebna *factory* klasa.



Slika 3.6. Životni ciklus ViewModela u usporedbi s aktivnosti [21]

Nadalje, LiveData je klasa koju možemo protumačiti kao *holder* podataka koji se može promatrati i koji svim promatračima pri promjeni podatka koji sadrži javlja novu vrijednost. LiveData je bitan za MVVM arhitekturu jer olakšava prosljeđivanje podataka iz ViewModela u View korištenjem *Observer* klase, tj. promatrača. Ona pokreće zadanu funkciju svakom promjenom promatranog LiveData objekta. Ovakav pristup olakšava ažuriranje korisničkog sučelja zahtijevajući samo definiranje funkcije koja će se pokretati pri promjenama i u njoj napisati kod za prikaz podataka, a kasnije samo ažuriranje vrijednosti LiveData objekta. Ako želimo ručno mijenjati vrijednost LiveData objekta moramo koristiti njegovu izvedenicu MutableLiveData. Primjer kreiranja i korištenja ViewModela i LiveData dan je kodom 3.3.6. Definirana je klasa *MyViewModel* koja nasljeđuje ViewModel i ima LiveData objekt *userLiveData* i funkciju *loadUserData* za učitavanje podataka o korisniku. U aktivnosti se instanciraViewModel, postavlja *Observer* na *userLiveData* i poziva funkciju za učitavanje podataka. Kada se vrijednost LiveData objekta postavi na novu vrijednost pokrenut će se funkcija *setUserData* i postaviti podatke na sučelje.

```
class MyViewModel: ViewModel() {

   val userLiveData = MutableLiveData<User>()

   fun loadUser() {
       val testUser = User()
       userLiveData.postValue(testUser)
   }

}


class UserActivity : Activity() {

 override fun onCreate(savedInstanceState: Bundle?) {
   super.onCreate(savedInstanceState)
   setContentView(R.layout.activity_user)

   val viewModel = ViewModelProviders.of(this).get(MyViewModel::class.java)

   viewModel.userLiveData.observe(this, Observer { user -> setUserData(user) })

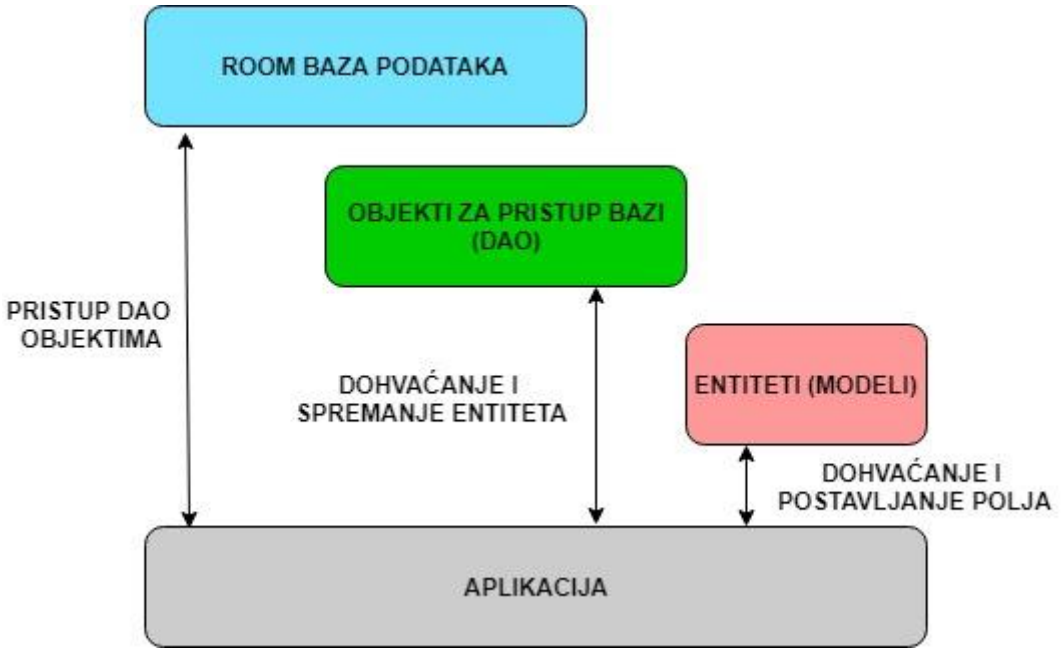
   viewModel.loadUser()
}

 private fun setUserData(user: User?) {
   user?.let {
       tvProfileName.text = it.username
   }
 }

}
```

Kod 3.3.6. Primjer kreiranja i korištenja ViewModel i LiveData klasa

Za komunikaciju sa lokalnom bazom podataka koristi se Room biblioteka. Ona je apstrakcija nad SQLite bazom podataka i omogućava s njom lakšu i bržu interakciju manje sklonu pogreškama. Njena implementacija u aplikaciji sastoji se od *Entity, Dao* i *Database* komponenti vidljivih na slici 3.7. *Entity* predstavlja jednu tablicu u bazi podataka, tj. jednu model klasu sa svim stupcima i informacijama o tablici. Ovdje se navodi primarni ključ, relacije, indeksi i drugo. *Dao* je klasa koja služi za navođenje svih upita prema bazi podataka u obliku funkcija s anotacijama koje određuju upit. Postoje već gotove anotacije za umetanje, ažuriranje i brisanje, a prilagođene upite poput dohvaćanja podataka treba upisati unutar *Query* anotacije. Postoji integracija RxKotlin i LiveData klasa u Room, pa je moguće vratiti podatke u nekim od njihovih tipova. U *Database* komponenti navode se svi *Entity* modeli i *Dao* klase, te funkcije za pristup implementacijama *Dao* klasa koje će Room automatski generirati. Upisuje se i trenutna verzija baze i drugi podaci.



Slika 3.7. Komponente Room implementacije [20]

Primjer korištenja Room biblioteke dan je kodom 3.3.7. Definirana je baza podataka s jednom tablicom *User* i pripadajućom *Dao* klasom. *User* model, tj. tablica sadrži primarni ključ id, korisničko ime i email. *Dao* klasa sadrži funkcije za umetanje, brisanje i dohvat trenutnog korisnika.

```
@Database(entities = [(User::class)],version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
}

@Dao
interface UserDao {

    @Query("SELECT * FROM $TABLE_USER limit 1")
    fun getCurrentUser(): User

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insert(user: User): Long

    @Delete
    fun delete(user: User)
}

@Entity(tableName = TABLE_USER)
data class User(
        @PrimaryKey
        var id: Int,
        var username: String,
        var email: String
)
```

Kod 3.3.7. Implementacija Room biblioteke

3.3.5. Firebase

Firebase je web i mobilna razvojna platforma namijenjena lakšoj integraciji i testiranju funkcionalnosti poput autentifikacije, online baze podataka i pohrane podataka. Besplatna je i omogućava brzu i jednostavnu implementaciju poslužiteljske strane aplikacije bez potrebnog znanja o poslužiteljskim tehnologijama. Postoje i brojni drugi alati poput notifikacija, praćenja događaja u aplikaciji i statistike o korištenju i slično. Dio je Googleove infrastrukture, pa se može računati na kvalitetan i siguran rad. U izvedenom softverskom rješenju koriste se Realtime Database, Authentication i Storage. Realtime Database je NoSQL baza podataka strukturirana kao jedan velik JSON objekt. Može se koristiti kao JSON API ili pomoću pripadajuće biblioteke koja omogućava čitanje i pisanje u bazu u stvarnom vremenu. U ovoj aplikaciji koristi se za spremanje i sinkronizaciju korisničkih podataka. Nadalje Authentication se koristi za registraciju i prijavu, a Storage za učitavanje profilnih slika i ikonica fraza. Firebase je kao poslužiteljski dio aplikacije odabran upravo zbog brze i lake integracije te mogućnosti korištenja bez velikog znanja o poslužiteljskim tehnologijama.

3.4. Način rada i struktura aplikacije

Aplikacija je strukturirana u četiri glavna modula: login, main, parent i child. Login modul obuhvaća sve vezano za prijavu i registraciju, te sinkronizaciju korisničkih podataka. Main je najjednostavniji i nudi samo izbor načina rada. Unutar parent modula nalaze se logika i prikazi za dohvaćanje i uređivanje podataka o dječjim profilima, frazama, postavki, profila, pregledavanje vijesti i korisnih informacija. Child modul sadrži implementacije igara pogađanja i AAC sustava. Svaki od modula sadrži sve relevantne Activitye, Fragmente i pripadajuće ViewModel i Repository klase. Osim ovih modula postoje mape s modelima, deklaracijama baze i mrežnog klijenta, zajedničkim klasama i slično. Za upravljanje stanjem aplikacije, prikazivanje poruka i prijenos nekih osnovnih podataka dohvaćenih u ViewModelu koristi se klasa Resource kao vrijednost LiveData objekta. Deklaracija i upotreba te klase vidljiva je u kodu 3.4.1. Ima parametre Status koji je enum klasa sa svim mogućim statusima, data generičkog tipa i message String. Pokazan je primjer pri registraciji korisnika na Firebase. U LiveData objekt postavlja se prvo status *loading* koji okida *Observer* te prikazuje očitavanje, a nakon toga pokušava registracija. Ako je ona uspješna vraća se *User* objekt koji se postavlja kao vrijednost LiveData objekta koji okida funkciju *handleResource* te odlazi na Main prikaz. Ako je neuspješna prikazuje se poruka greške. Svaki modul ima jedan ili više Repository i ViewModel. Repository klase služe za pristup Room bazi podataka, Retrofit mrežnom klijentu i ostalim izvorima podataka poput *Shared Preferences*.

ViewModel preko Repository klasa postavlja podatke u LiveData objekte te obavještava aktivnost o stanju. Pozivi na bazu i API kombiniraju se i transformiraju pomoću RxKotlin biblioteke. Sve se to čini asinkrono. Tako je moguće na primjer prvo povući svježe podatke s online baze podataka te osvježiti one u lokalnoj bazi, a tek ih zatim učitati, obraditi i prikazati. Svi RxKotlin pozivi vraćaju *Disposable* objekt kojeg trebamo odbaciti poslije korištenja radi izbjegavanja problema s memorijom. Na rezultate se pretplaćuje u pozadinskoj niti da se ne blokira izvršavanje ostalih naredbi, a rezultati se dobivaju u glavnoj *UI* niti. Uključen je i RxKotlin dodatak za Room bazu podataka i Retrofit2 mrežni klijent kako bi se mogli vraćati asinkroni rezultati istog tipa iz ove biblioteke i kombinirati ih. Pristup objektima omogućava se pomoću Dagger2 ubrizgavanja ovisnosti. Stvara se stablo ovisnosti te se na kraju na isti način ViewModel ubacuje u aktivnost. ViewModel aktivnosti dijeli se s pripadajućim fragmentima. U slijedećem dijelu opisat će se tijek i način rada aplikacije prema modulima, te dati neki bitniji algoritmi i kodovi.

```
data class Resource<T>(val status: Status, val data: T?, val message: String?)


VIEWMODEL:

resourceLiveData.value = Resource(Status.LOADING, null, null)
repository.register(signupRequest, object : Listener<User> {
    override fun onSucces(user: User) {
        resourceLiveData.value = Resource(Status.SUCCESS, user, null)
    }

    override fun onFailure(t: Throwable) {
        resourceLiveData.value = Resource(Status.MESSAGE, null, t.message)
    }
})

ACTIVITY:

override fun handleResource(resource: Resource<User>?) {
    resource?.let {
        showLoading(it.status)
        when (it.status){
            Status.SUCCESS -> {
                startMainActivity()
            }

            Status.MESSAGE -> {
                showMessage(it.message)
            }
        }
    }
}
```

Kod 3.4.1. Korištenje *Resource* klase za stanja i podatke

3.4.1. Prikaz prijave - *Login*

Pri pokretanju aplikacije prvi se pokreće *Login* prikaz. Na lokalnu bazu podataka radi se upit koji provjerava postoji li korisnik u bazi, tj. je li već prijavljen. Ako postoji, otvara se *Main* prikaz, a u suprotnom prikazuju se forme za prijavu i registraciju. Korisnik se može prijaviti ranije stvorenim računom, koristeći Google račun ili registrirati novi račun. Kada se korisnik registrira stvara se novi račun na Firebase Authentication servisu koji vraća jedinstveni ID, a zatim se svi ostali podaci spremaju pod tim ID-em u lokalnu bazu podataka i Firebase Realtime Database. Nadalje, korisnik treba dodati dječje profile na svoj račun. Kada je i to obavljeno, prijavljen je i prikazuje se *Main* prikaz. Pri prijavi svi prethodno spremljeni korisnički podaci sinkroniziraju se na lokalnu bazu, kao i kod svakog idućeg pokretanja aplikacije dok je korisnik prijavljen. Kada se odabere *Google Sign In* provjerava se postoji li već u online bazi podataka korisnik s tim ID-em, pa odlučuje hoće li se već postojeći podaci sinkronizirati ili se stvara novi korisnik. Nadalje, moguće je resetirati lozinku pritiskom na *Forgot password,* nakon kojeg korisnik na registrirani email dobiva novu lozinku. Dijagram toka prikazan je slikom 3.8. Za primjer ranije spomenutog ulančavanja poziva na mrežni klijent i bazu podataka dan je kod 3.4.2. u kojem je vidljiva logika za dohvaćanje korisničkih podataka s online baze i spremanje u lokalnu nakon uspješne prijave.

```
RETROFIT API:

@GET("users/{userId}.json")
fun getUser(@Path("userId") userId: String): Single<User>


REPOSITORY:

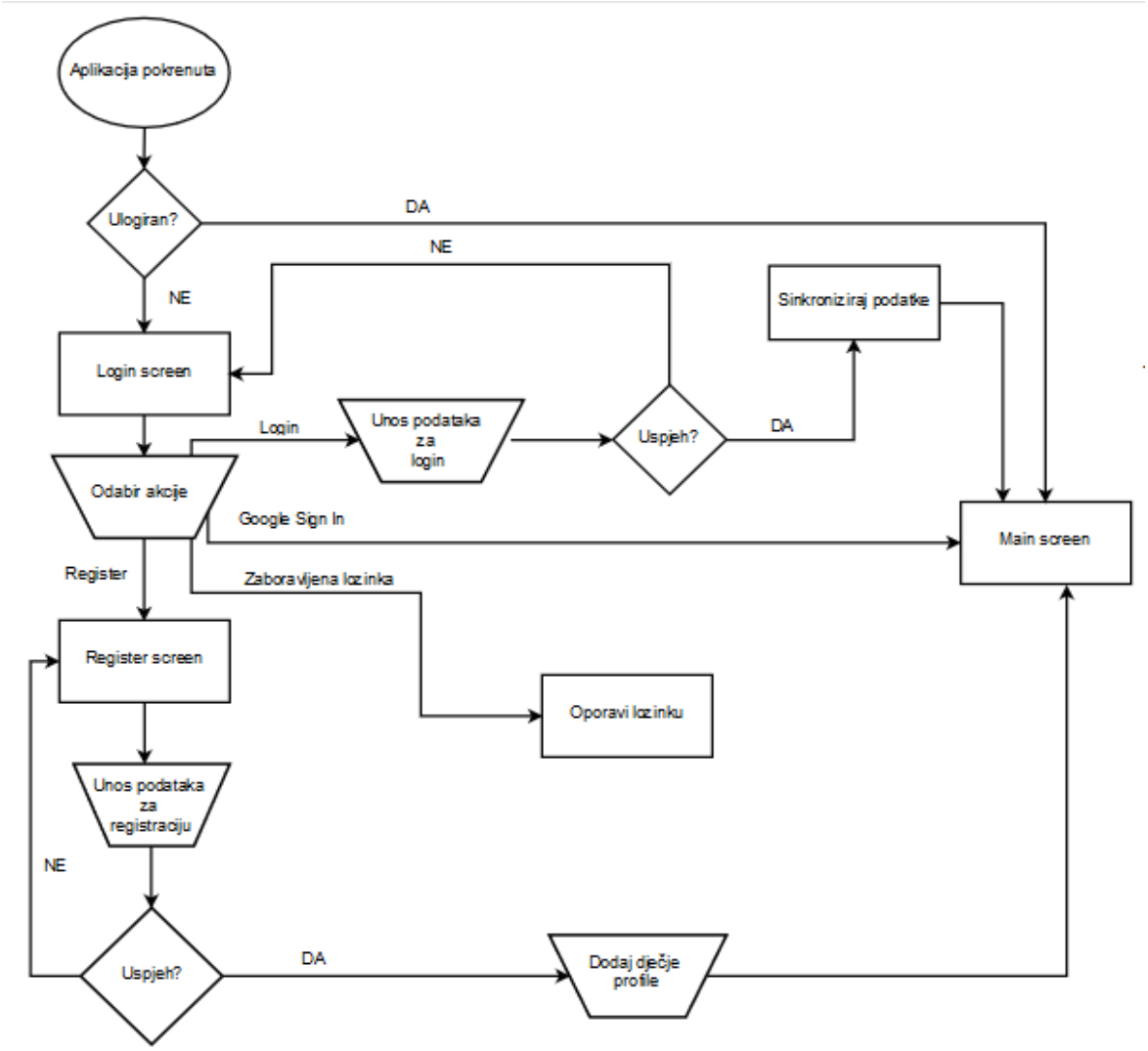
fun fetchAndSaveUser(userId: String): Completable {
   return api.getUser(userId)
            .flatMapCompletable { user -> saveUser(user) }
            .subscribeOn(ioScheduler)
            .observeOn(mainScheduler)
}


VIEWMODEL:

private fun fetchAndSaveUserData(userId: String) {
   compositeDisposable.add(
            repository.fetchAndSaveUser(userId).subscribe(
                    { syncContentData() },
                    { error -> throwErrorAndLogOut(R.string.fetch_error) }
            )
   )
}
```

Kod 3.4.2. Dohvaćanje i spremanje korisničkih podataka

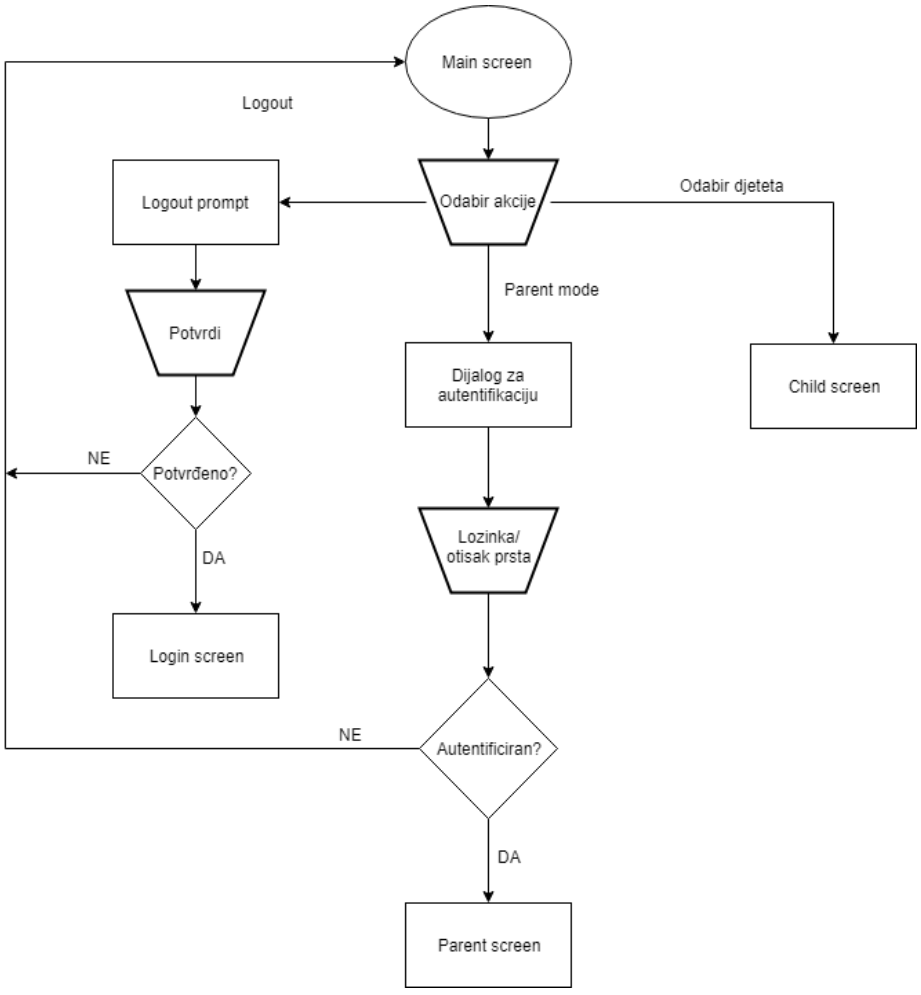
Unutar API klase definiran je GET zahtjev prema ruti s korisnicima koji vraća *Single* RxKotlin objekt koji predstavlja asinkroni poziv s jednim rezultatom. Repository pristupa API-ju preko kojeg dohvaća korisničke podatke, nakon čega se oni spremaju lokalno funkcijom *saveUser*. Vraća se *Completable* RxKotlin objekt na koji se u ViewModelu pretplaćuje te pri uspješnom izvršavanju nastavlja dalje s izvršavanjem, a u suprotnom prikazuje poruka o pogrešci i odjavljuje korisnik. Objekt koji pretplata vraća dodaje se u objekt *compositeDisposable* zbog kasnijeg odbacivanja i oslobađanja memorije.



Slika 3.8. Dijagram toka *Login* prikaza

3.4.2. Glavni prikaz - *Main*

Ovaj prikaz služi za izbor između načina rada za roditelje i djecu. Odabirom načina rada za roditelje potrebno je autentificirati se lozinkom ili otiskom prsta. Ako lozinka nije prethodno postavljena korisnik će se upitati da to napravi te će se ona pohraniti u lokalnu i online bazu podataka. Ako uređaj ima senzor za otisak prsta i već postavljen bar jedan Fingerprint ID u dijalogu za autentifikaciju prikazat će se ikonica otiska prsta kao pokazatelj da se on može koristiti. Pri uspješnoj autentifikaciji otvorit će se *Parent* prikaz. Nasuprot tome, kada dijete želi koristiti aplikaciju može odabrati svoje ime sa prikazane liste te otvoriti *Child* prikaz šaljući mu svoj ID za spremanje statistike. Korisnik se iz *Main* prikaza može odjaviti pritiskom na *Logout* u izborniku na vrhu. Potrebno je dodatno potvrditi ovu akciju. Tijek događaja ovog prikaza opisan je slikom 3.9.



Slika 3.9. Dijagram toka *Main* prikaza

3.4.3. Roditeljski prikaz - *Parent*

U ovom prikazu koristi se *Drawer menu* za navigaciju i sve akcije. Odabirom željenog prikaza izmjenjuju se Fragmenti koji predstavljaju logiku i sučelje za njega. Početni prikaz jest *Children* prikaz na kojem korisnik može pregledati dječje profile i njihove statistike, urediti ih, obrisati i dodati nove. Dječji profili povlače se iz Room baze podataka kao *Flowable* RxJava tip, što omogućuje da se kad god nastanu promjene u bazi nad tablicom dječjih profila prikazani podaci automatski osvježe. Potrebno je samo izvršiti neku izmjenu nad podacima. Na isti način izveden je *Phrases* prikaz na kojemu se mogu pregledati i dodati fraze. Pri dodavanju dječjih profila isti se sprema prvotno na online bazu podataka, te tada na lokalnu. Fraze se spremaju samo lokalno, kako bi pojedini korisnik mogao dodati prilagođene fraze koje bi njegovo dijete moglo zatrebati. Za prikaz elemenata liste korišten je *RecyclerView*. Učitavanje podataka *Flowable* tipa iz baze dano je primjerom učitavanja fraza kodom 3.4.3. Koristi se upit koji izvlači iz baze sve dostupne fraze u obliku *Flowable* liste. Toj funkciji izravno pristupa Repository klasa *Parent* modula te prosljeđuje *Flowable* listu u ViewModel. U njemu se asinkrono pretplaćuje na podatke i ta se funkcija poziva svaki puta kada se dogode promjene nad podacima u bazi. Koristi se ranije spomenuti *resourceLiveData* za status, ali i novi *phraseLiveData* samo za učitavanje fraza zbog dijeljenja ViewModela među fragmentima koji rade s drukčijim tipovima podataka.

DAO KLASA:

```
@Query("SELECT * FROM $TABLE_AAC")
fun getAllPhrases(): Flowable<List<AacPhrase>>
```

REPOSITORY:

```
fun loadPhrases(): Flowable<List<AacPhrase>> {
    return aacDao
            .getAllPhrases()
            .subscribeOn(ioScheduler)
            .observeOn(mainScheduler)
}
```

VIEWMODEL:

```
fun loadPhrases() {
    resourceLiveData.value = Resource.loading()
    compositeDisposable.add(
            repository.loadPhrases().subscribe{
                        phraseLiveData.value = it
                        resourceLiveData.value = Resource.success(null)
            }
    )
}
```

Kod 3.4.3. Asinkrono dohvaćanje svih fraza u *Flowable* obliku

Otvaranjem statistike prikazuje se *Child Scores* prikaz koji sadrži dva grafa, duljinu igranja igara te broj pogrešaka u igrama razvrstane prema datumu. Za prikaz grafova koristi se biblioteka *MPChart.* Korisnik na *Profile* prikazu može mijenjati lozinku, profilnu sliku i korisničko ime. Pri učitavanju profilne slike ona se sprema u lokalnu pohranu aplikacije spremajući putanju do nje u lokalnu bazu podataka i učitava na Firebase Storage za dohvaćanje pri sinkronizaciji. Učitavanje slike na Firebase Storage pokazano je kodom 3.4.4. Prvo se predana *Bitmap* slika kompresira u *ByteArrayOutputStream* te pretvara u *ByteArray* koji se može učitati. Dohvaća se referenca na Firebase Storage te dodaje element naziva istog poput korisničkog ID-ja. Na referenci se poziva funkcija *putBytes* i predaje ranije stvoreni *ByteArray*. Pri uspješnom učitavanju ažuriraju se korisnički podaci na lokalnoj i online bazi. Ostali parametri su zanemareni.

```
fun updateUserData(bitmap: Bitmap) {
   resourceLiveData.value = Resource.loading()

   val baos = ByteArrayOutputStream()
   bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos)
   val data = baos.toByteArray()

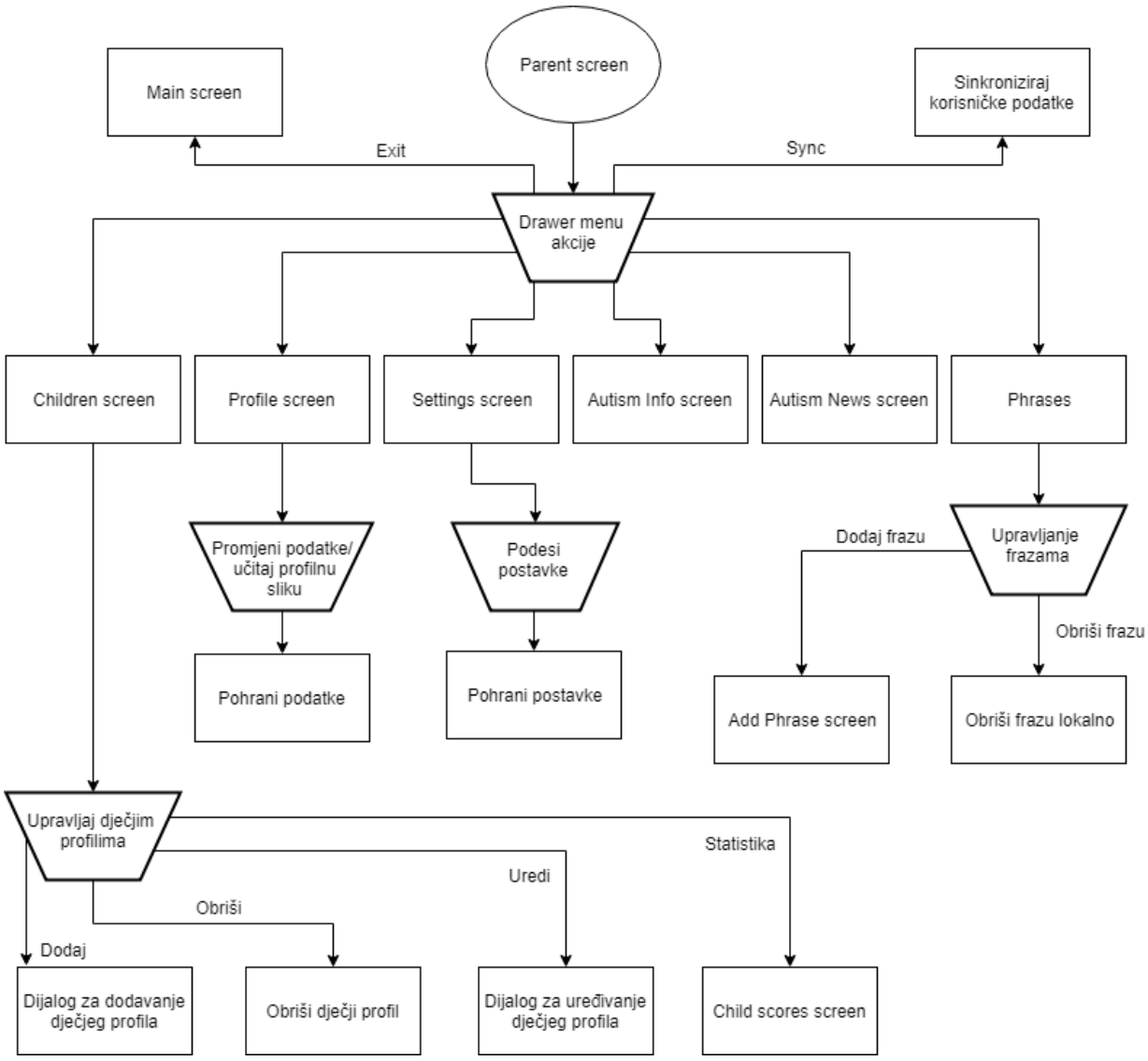
   val storageRef = storageRef.child("$userId.jpg")
   storageRef.putBytes(data)
            .addOnSuccessListener {
                updateUserOnApiAndDb()
            }.addOnFailureListener{
                resourceLiveData.value = Resource.message(R.string.error)
            }
 }
```

Kod 3.4.4. Učitavanje profilne slike na Firebase Storage

Na *Settings* prikazu moguće je mijenjati postavke aplikacije spremajući odgovarajuće vrijednosti u *SharedPreferences*. Ponuđene postavke su omogućavanje zvuka u aplikaciji, te visina i brzina glasa pri izgovaranju rečenica u AAC sustavu. Za *SharedPreferences* koristi se biblioteka Remember. Nadalje na *Autism News* prikazu povlači se *RSS Feed* vijesti o autizmu u XML obliku. Koristi se kao i za ostale mrežne pozive Retrofit. Kada *Autism News* prikaz pokrene pokušava se osvježiti podatke u lokalnoj bazi. Pri uspješnom dohvaćanju podataka oni se spremaju. Nastavlja se neovisno o uspjehu te se učitavaju podaci iz lokalne baze. Ovako su pristupačni najsvježiji podaci kad god je dostupna Internet konekcija. Ulančani poziv vidljiv je u kodu 3.4.5. *Autism Info* prikaz učitava popularnu Web stranicu o autizmu u *WebView* elementu. Korisnik može osim odabira prikaza i izaći iz načina rada za roditelje, sinkronizirati korisničke podatke, te se odjaviti brisanjem svih podataka iz lokalne baze i prijelazom na *Login* prikaz. Cjelokupni tijek načina rada za roditelje vidljiv je na slici 3.10.

```
fun fetchFeeds(): Single<List<FeedItem>> {
   return api.getFeed(RSS_URL)
            .onErrorResumeNext { Single.just(RSS()) }
            .doOnSuccess { feed ->
                feed.channel.feedItems.let {
                    if(it.isNotEmpty()) {
                        feedItemDao.insertMultiple(it)
                    }
                }
            }.flatMap {
                feedItemDao.getItems()
            }.subscribeOn(ioScheduler)
            .observeOn(mainScheduler)
 }
```

Kod 3.4.5. Dohvaćanje RSS vijesti o autizmu



Slika 3.10. Dijagram toka *Parent* prikaza

3.4.4. Dječji prikaz - *Child*

Child modul nudi djetetu igranje igara na pogađanje i korištenje AAC sustava. Mogu se pokrenuti igre pogađanje emocija, boja i brojanja geometrijskih likova. Emocije i boje dijele većinu logike koristeći *ViewPager* za prolazak kroz pitanja, a prebrojavanje geometrijskih likova izvedeno je posebno koristeći *RecyclerView*. Korisniku se prikazuje pitanje, popratna slika koja ga opisuje te četiri moguća odgovora. Odabirom točnog odgovora prelazi se na iduće pitanje ili igra završava, a kod netočnog samo se bilježi jedna pogreška. Kod prebrojavanja geometrijskih likova unosi se broj u *EditText* element te se provjerava njegova točnost. Dana je zvučna povratna informacija o tome je li odgovor bio točan ili ne. Tijekom svake igre bilježi se i njeno trajanje. Po završetku ili izlasku iz nje podaci se spremaju u lokalnu i online bazu. Slučaj da se podaci ne uspiju učitati na online bazu, a spreme se u lokalnu rješava se automatskom sinkronizacijom pri svakom pokretanju aplikacije. Sadržaj kvizova moguće je lako proširiti dodavanjem pitanja na poslužitelju. Kodom 3.4.6. prikazan je način generiranja nasumičnog broja različitih geometrijskih likova za igru. Uvijek se vraća 36 elemenata kojima se nasumično odabire vidljivost, oblik, boja i rotacija. Dostupne boje i oblici su prethodno definirani u listama *colors* i *shapeTypes*. Lista *shapeTypes* sadrži *Drawable* elemente koji se postavljaju kao slika *ImageView* elemenata s nasumičnom bojom i rotacijom. Odabire se određeni oblik koji treba prebrojati i računa se koliko ih je vidljivo radi odlučivanja točnosti odgovora. Ostali dijelovi *RecyclerViewa* izostavljeni su zbog kompleksnosti.

```
override fun getItemCount() = 36

override fun onBindViewHolder(holder: MathShapeVH, position: Int) {
    val random = Random(System.currentTimeMillis())
    val shapeType = random.nextInt(3)
    val color = colors[random.nextInt(colors.size)]

    with(holder.itemView.ivShape) {
        val shape = shapeTypes[shapeType]
        if(random.nextBoolean()) {
            if(shape == pickedShapeType) {
                pickedElements++
            }
            holder.itemView.visibility = View.VISIBLE
        } else {
            holder.itemView.visibility = View.INVISIBLE
        }
        val drawable = context.getDrawable(shape)
        setImageDrawable(drawable)
        drawable.setTint(color)
        rotation = random.nextInt(360).toFloat()
    }
}
```

Kod 3.4.6. Generiranje geometrijskih oblika za igru prebrojavanja

Korištenjem implementiranog sustava korisnik može birati fraze razvrstane po kategorijama i spremljene rečenice te ih postavljati i slagati na prikaz. Najveća duljina složene rečenice može biti 20 fraza. Moguće je i unijeti tekst direktno, prikazujući ga identično kao i dostupne fraze. Rečenice se slažu u obliku liste fraza. Složenu rečenicu moguće je spremiti u lokalnu bazu podataka. Pomoću TTS sustava ugrađenog u Android složena rečenica se izgovara pritiskom na tipku za reprodukciju. Izvlače se tekstovi iz svake pojedine fraze te se pomoću *StringBuilder* klase kombiniraju u jedan *String* koji se predaje TTS sustavu na izgovor. Korištenje TTS sustava sa frazama unutar aktivnosti prikazano je kodom 3.4.7. Prvotno se stvara instanca *TextToSpeech* klase i provjerava podržava li je uređaj. Nakon toga postavljaju se postavke sustava poput jezika, brzine i visine glasa. Za prikaz odabranih fraza koristi se *RecyclerView* s pripadajućim adapterom *aacDisplayAdapter*. Klikom na prikazanu frazu ona se briše iz liste fraza i miče s prikaza. Drugi *RecyclerView* sadrži kategorije, fraze i spremljene rečenice te pritiskom na jednu frazu ili rečenicu poziva se funkcija *addItemToDisplay* dodajući fraze na prikaz. Pritiskom na tipku za reprodukciju poziva se funkcija *speak* koja ako je TTS podržan i zvuk uključen reproducira složenu rečenicu. Dijagram toka *Child* modula dan je slikom 3.11.

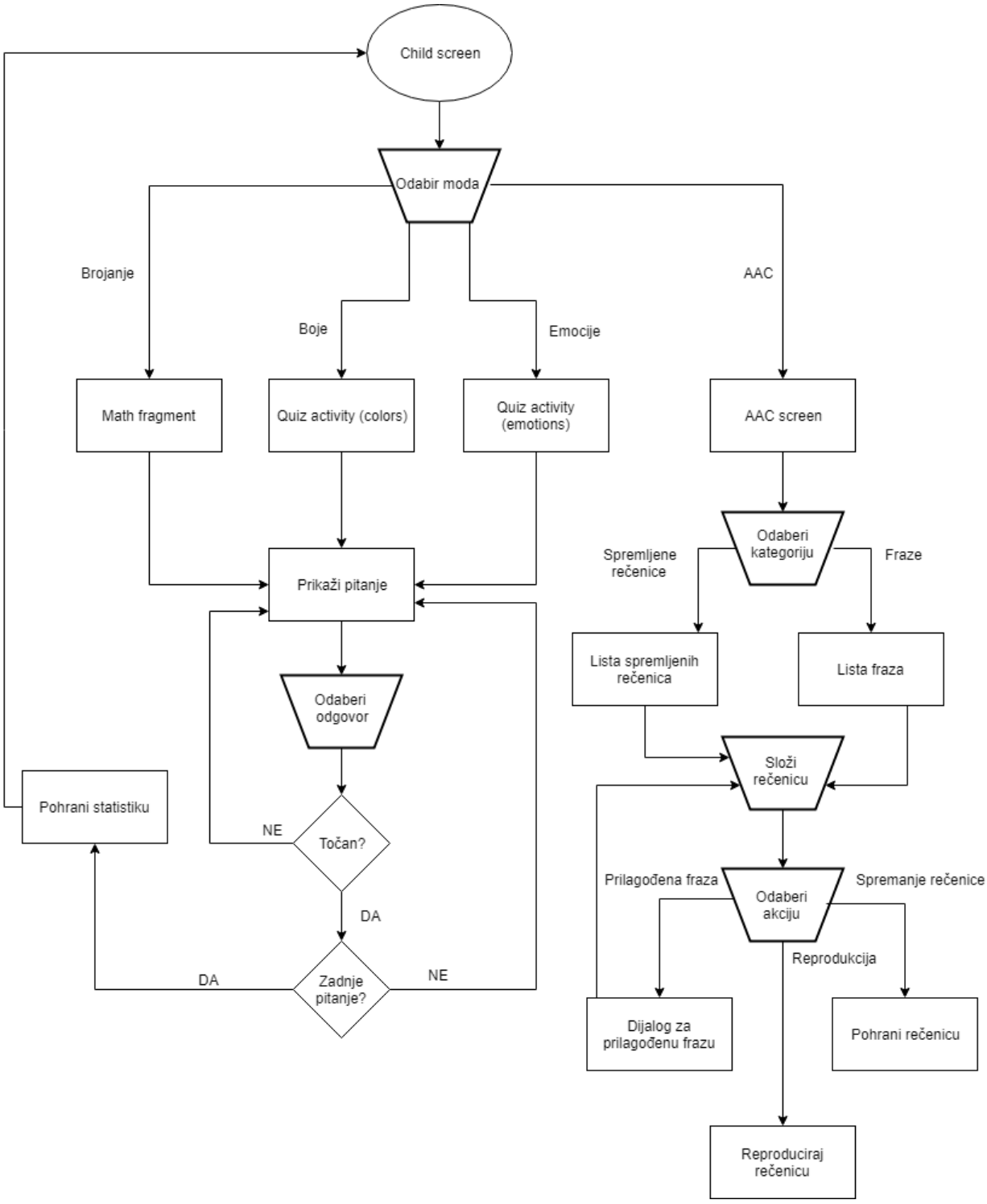
```
val tts = TextToSpeech(this) {
   if (it == TextToSpeech.SUCCESS) {
       ttsSupported = true
       tts.language = Locale.US
       tts.setSpeechRate(viewModel.getTtsSpeed())
       tts.setPitch(viewModel.getTtsPitch())
   }
}

private fun setupAacDisplay() {
     aacDisplayAdapter = AACAdapter(emptyList(), { phrase, position ->
         aacDisplayAdapter?.deleteItem(position)
         ttsWords.removeAt(position)
     })
     rvAacDisplay.adapter = aacDisplayAdapter
     rvAacDisplay.layoutManager = GridLayoutManager(this, 5)
}

fun addItemToDisplay(phrase: AacPhrase) {
    aacDisplayAdapter?.addItem(phrase)
    ttsWords.add(phrase.text)
}

fun speak(toSpeak: String) {
   if (ttsSupported) {
       if(viewModel.isSoundOn()) {
           tts.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, Bundle(), null)
       } else {
           showMessage(R.string.audio_disabled, null)
       }
   } else {
       showMessage(R.string.feature_not_supported, null)
   }
}
```

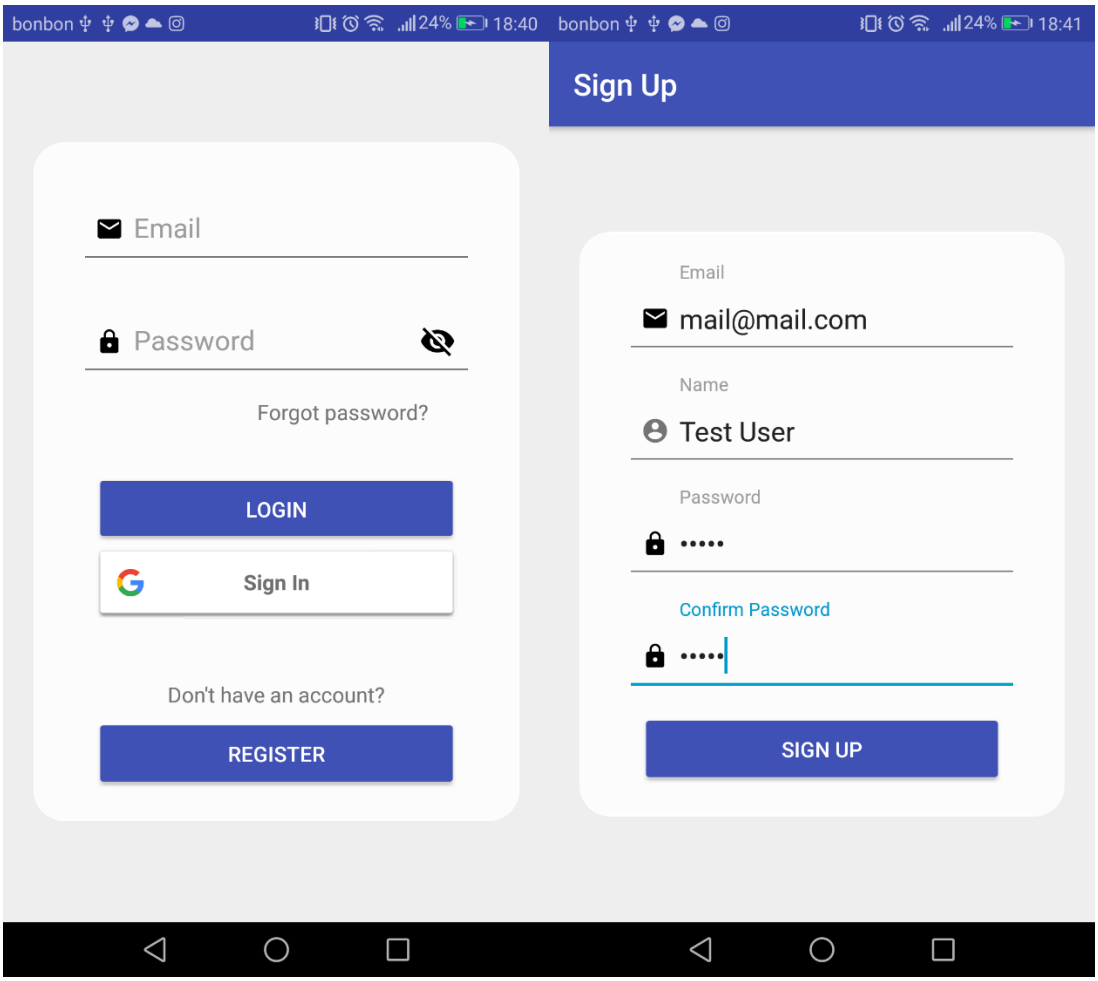
Kod 3.4.7. Korištenje ugrađenog TTS sustava za AAC



Slika 3.11. Dijagram toka *Child* modula

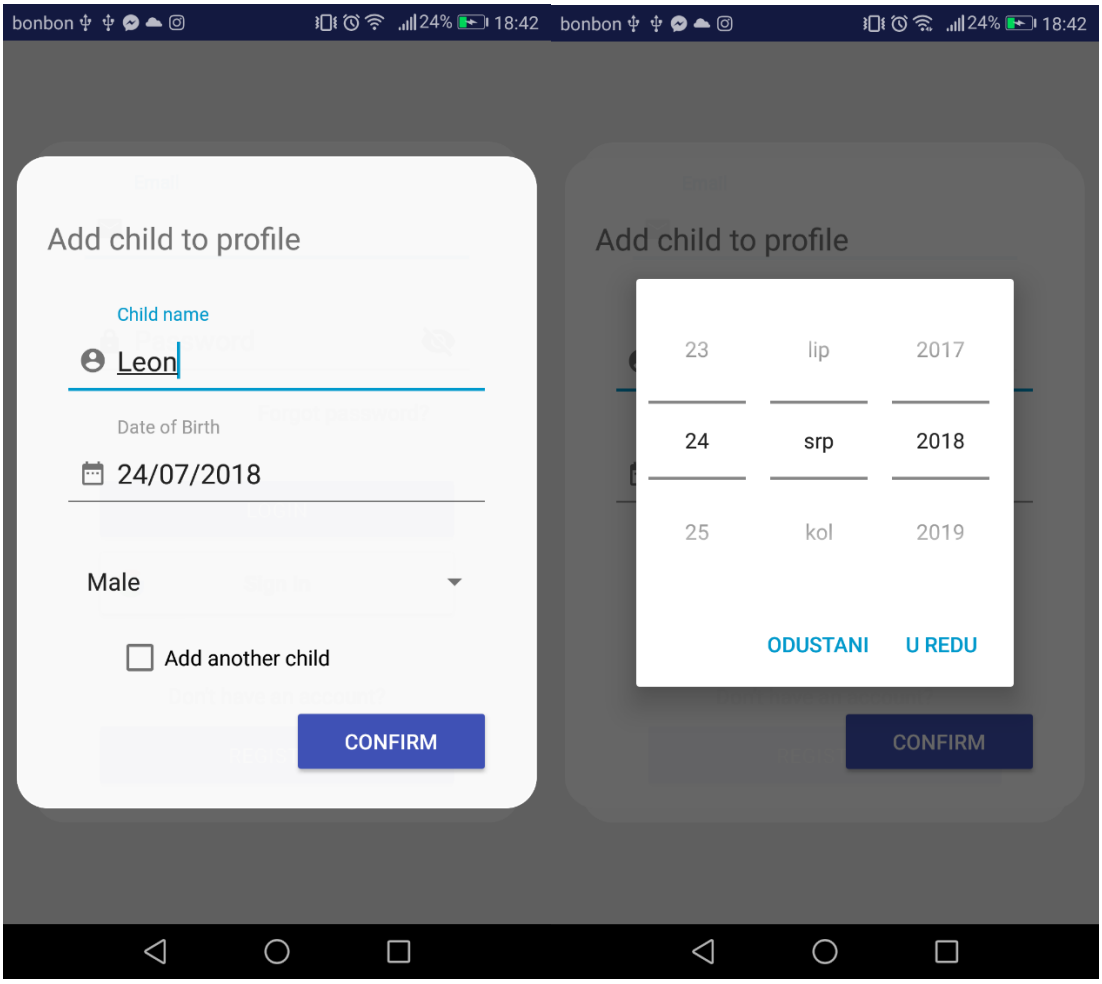
3.5. Upute za korištenje

U slijedećem dijelu bit će dane upute za korištenje aplikacije te opisano sučelje i sve moguće akcije. Odgovarajući dio sučelja bit će prikazan slikom iz aplikacije. Nakon otvaranja aplikacije prvi prikaz koji se pokazuje jest onaj za prijavu korisnika vidljiv na slici 3.12. U slučaju da je korisnik već prijavljen ovaj prikaz se preskače te se može nastaviti s korištenjem aplikacije. Korisnik se može prijaviti upisivanjem emaila i lozinke u polja za unos te pritiskom na *Login* tipku. Upisanu lozinku moguće je vidjeti pritiskom na ikonicu s krajnje desne strane polja za unos lozinke. Također, korisnik se može prijaviti u aplikaciju pomoću Google računa pritiskom na *Sign In* s Google logom. Ako već postoji korisnički račun prijavit će se, a u suprotnom bit će potreban unos dodatnih podataka. Pritiskom na *Forgot Password* otvorit će se dijalog u koji korisnik treba upisati email za resetiranje lozinke. Pritiskom na *Register* tipku otvara se prikaz za registraciju gdje treba upisati email, korisničko ime, lozinku i njenu potvrdu u odgovarajuća polja. Pritiskom na *Sign Up* kreira se račun te traži dodavanje dječjih profila dijalogom na slici 3.13.



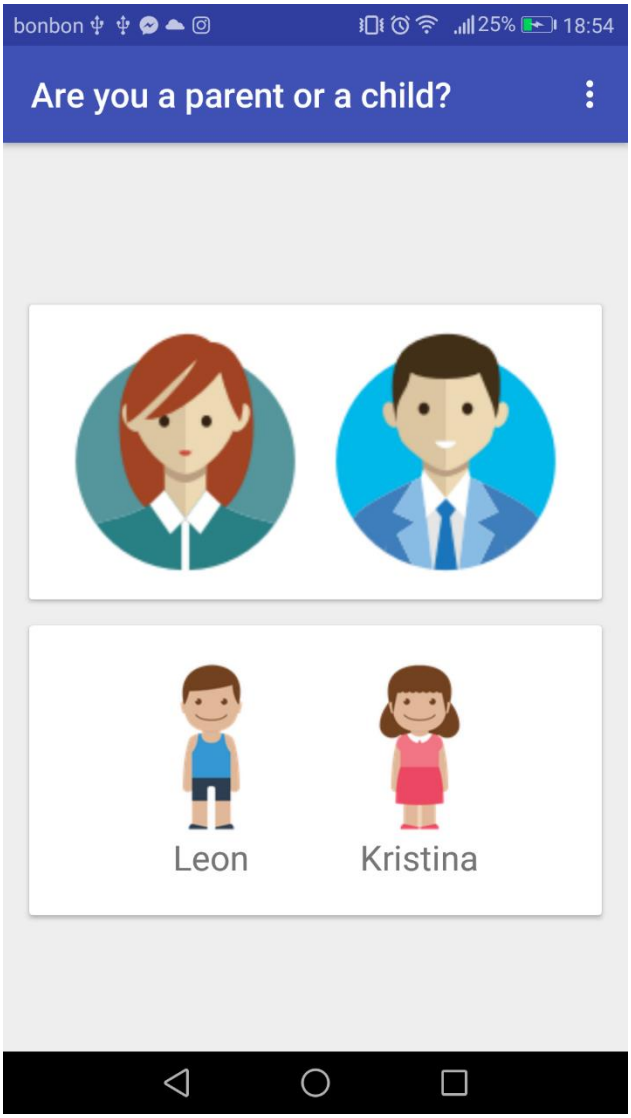
Slika 3.12. Prikazi za prijavu i registraciju

Dijalog za dodavanje dječjih profila zahtijeva unos imena djeteta, datum rođenja i spol u odgovarajuća polja za unos. Pritiskom na polje za unos datuma rođenja otvara se dijalog za odabir datuma. Označavanjem *Add another child* opcije otvara se još jedan dijalog nakon trenutnog za dodavanje još jednog djeteta. Proces se nastavlja sve dok se opcija ne označi te se prelazi na *Main* prikaz.



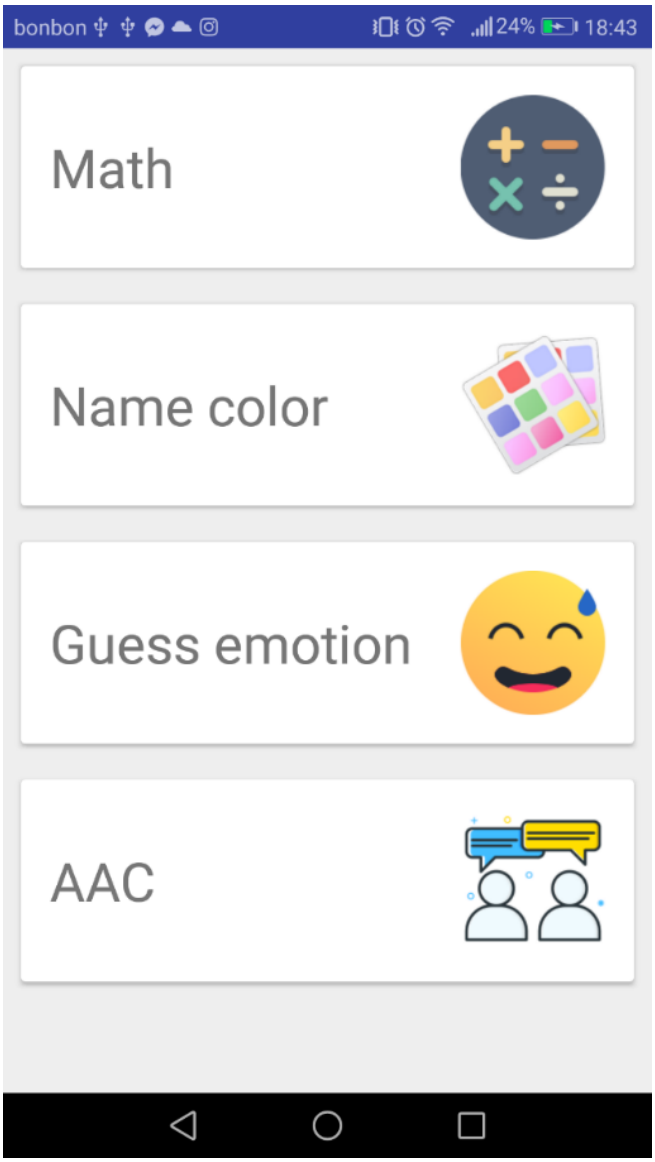
Slika 3.13. Dijalog za dodavanje dječjih profila

Main prikaz nudi odabir načina rada aplikacije, onog za roditelje i za dijete. Pritiskom na ikone roditelja otvara se *Parent* prikaz, a pritiskom na pojedino ime djeteta otvara se *Child* prikaz za to dijete. Pritiskom na *Toolbar menu* nudi se opcija za odjavu korisnika koja ponovno vodi na *Login* prikaz.



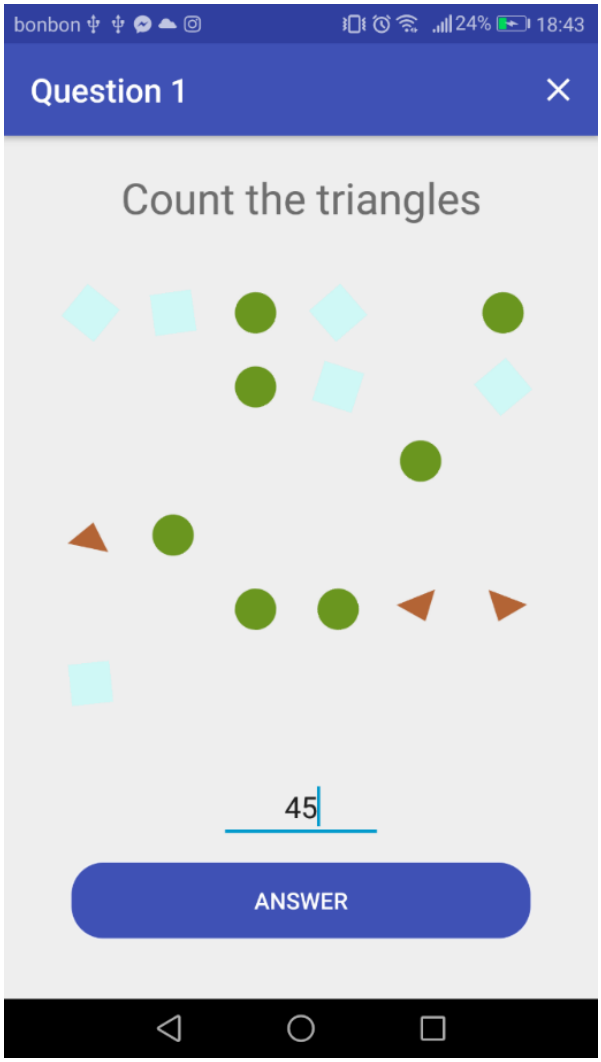
Slika 3.14. *Main* prikaz

Ulaskom u *Child* prikaz korisniku se nudi izbor igara na pogađanje i AAC sustava sa slike. Pritiskom na *Math* otvara se igra prebrojavanja geometrijskih likova, *Guess emotion* vodi do pogađanja emocija, a *Name color* pogađanja boja. *AAC* otvara AAC sustav.



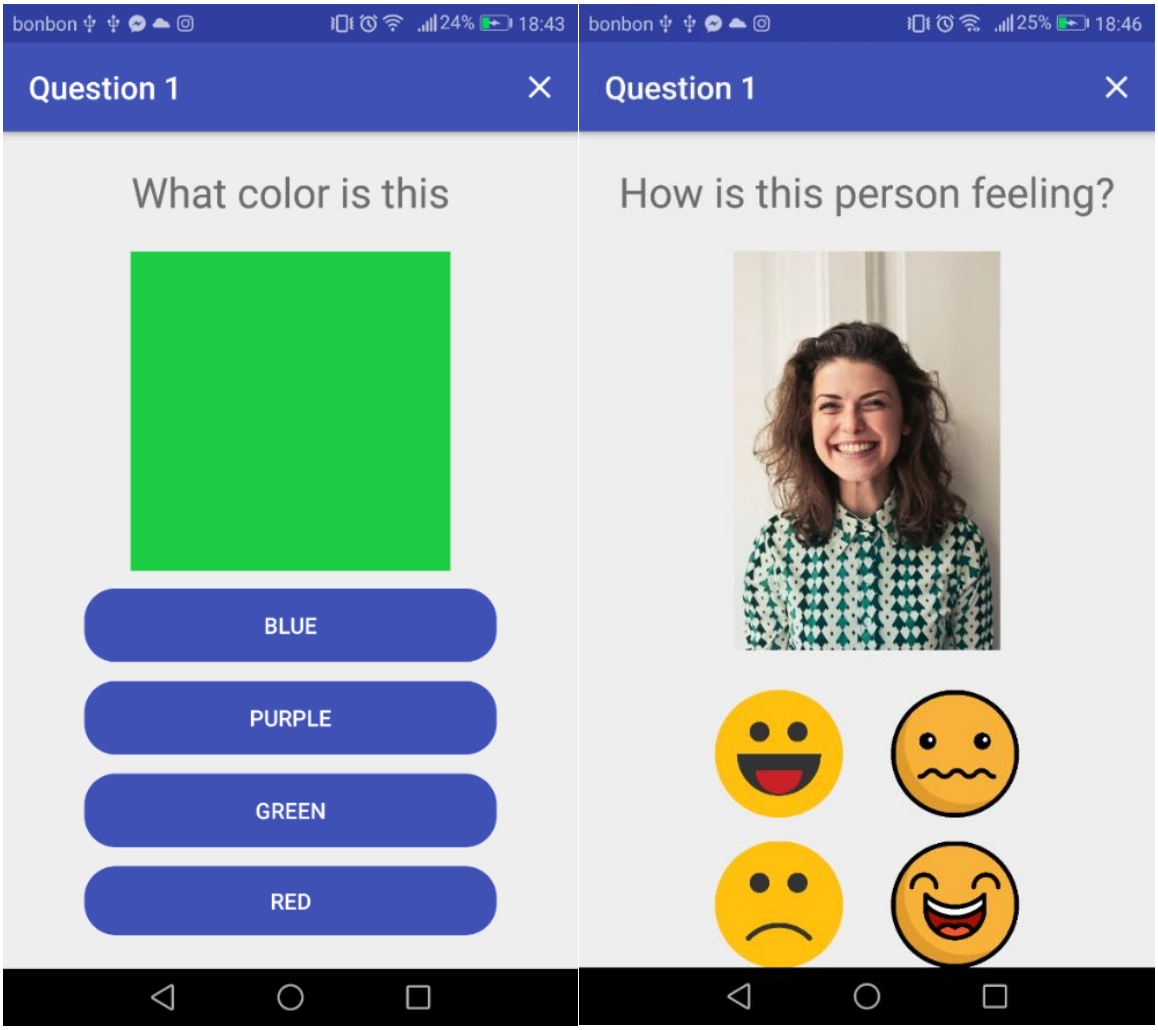
Slika 3.15. Child prikaz

Kod prebrojavanja geometrijskih likova prikazuje se nasumičan broj trokuta, pravokutnika i krugova različitih boja i rotacija. Potrebno je prebrojati zadane likove te upisati točan broj u polje za unos. Pritiskom na *Answer* provjerava se točnost odgovora te nastavlja dalje ako je točan ili samo javlja da je netočan u suprotnom.



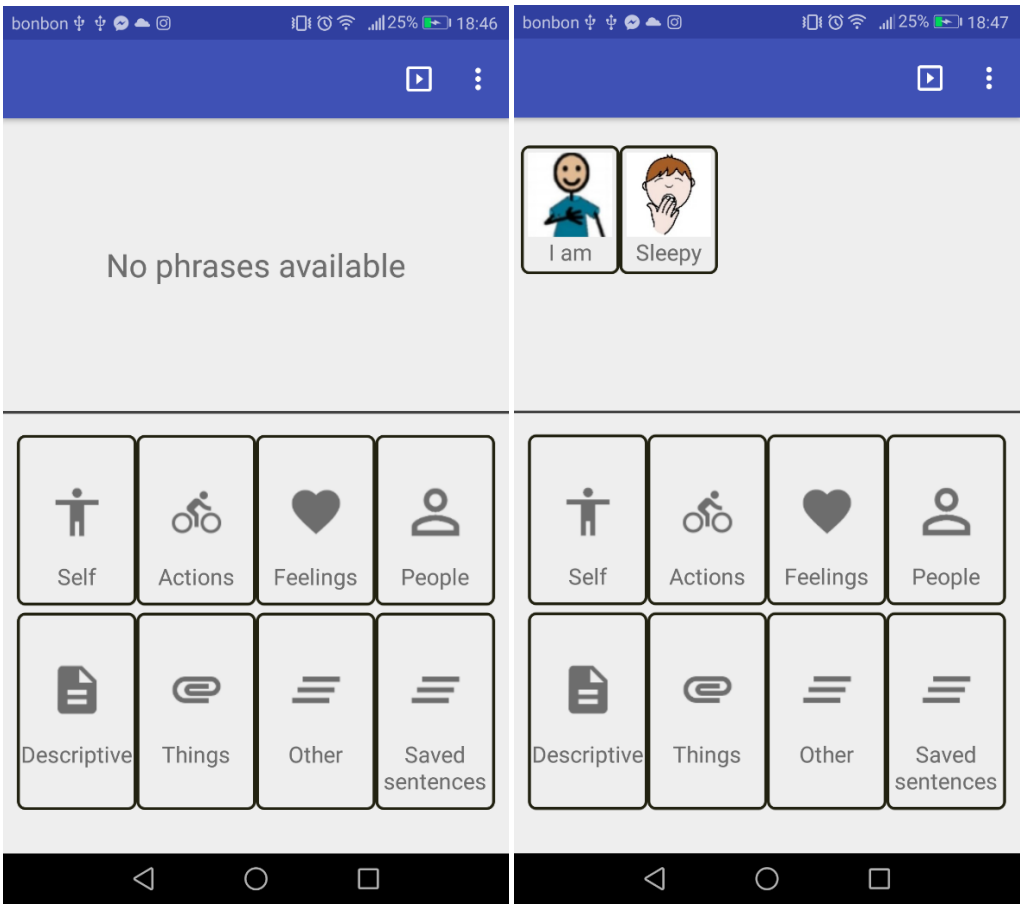
Slika 3.16. Igra prebrojavanja geometrijskih likova

Igre pogađanja boja i emocija imaju slično sučelje, s jedinom razlikom što su za emocije ponuđeni odgovori u obliku ikonica koje predstavljaju određene emocije, a za boje nazivi tih boja. Odabirom odgovora provjerava se je li odgovor točan te nastavlja na iduće pitanje ako je, a u suprotnom javlja greška korisniku.



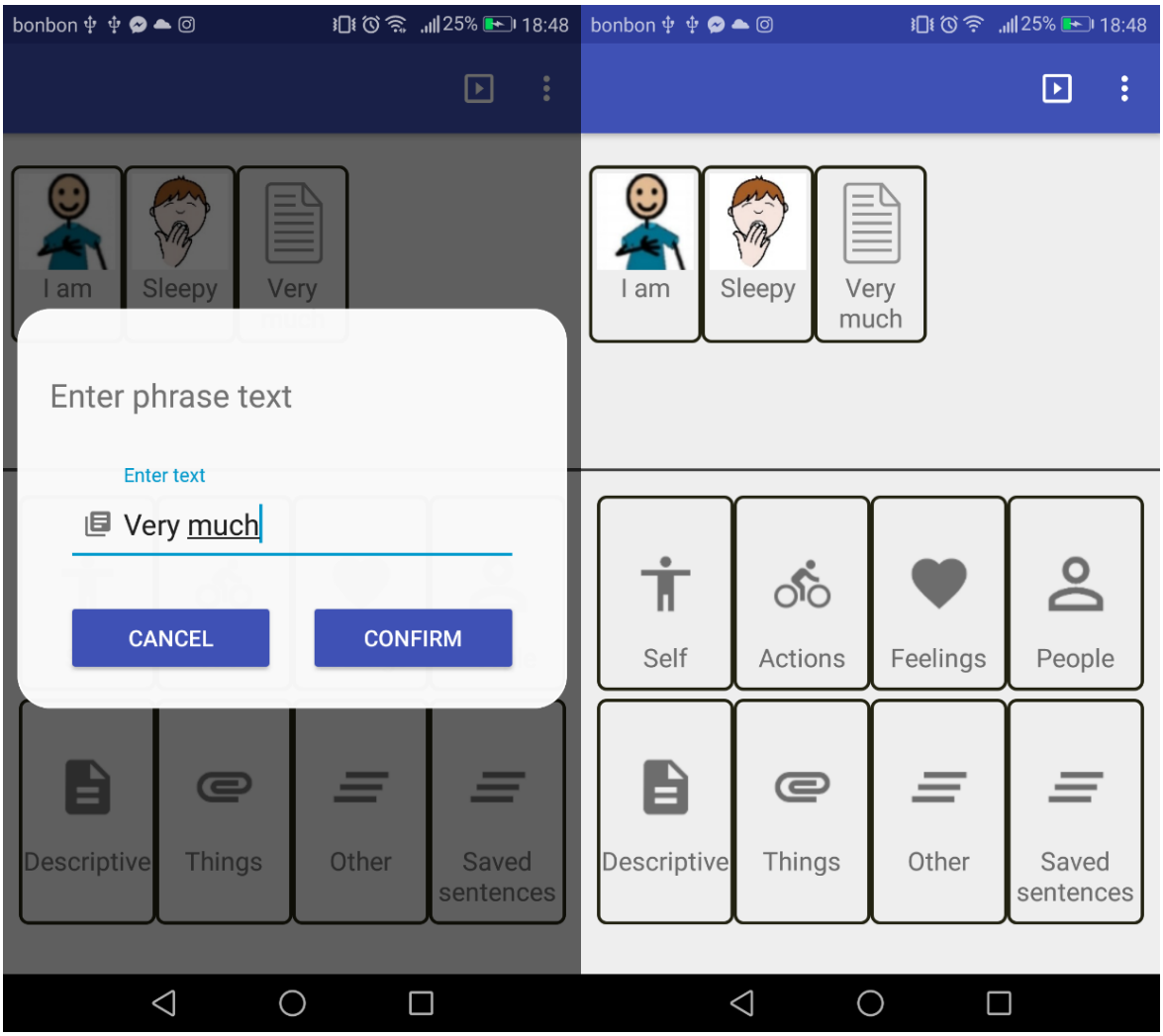
Slika 3.17. Igre pogađanja boja i emocija

Ulaskom u AAC prikaz se dijeli na dva dijela vertikalno. Gornji dio služi za prikaz konstruirane rečenice sastavljene od fraza, a donji dio služi kao izbornik za dodavanje fraza u tu rečenicu. U izborniku fraze su podijeljene prema kategorijama, a moguće je i dodati više fraza odjednom u obliku spremljene rečenice. Dugim pritiskom na frazu prilikom njenog izbora reproducira se njen tekst. Pritiskom na *Play* ikonicu u gornjem izborniku konstruirana se rečenica reproducira. Pritiskom na *Save sentence* iz istog izbornika ona se sprema lokalno i omogućuje dodavanje na prikaz iz kategorije *Saved sentences*.



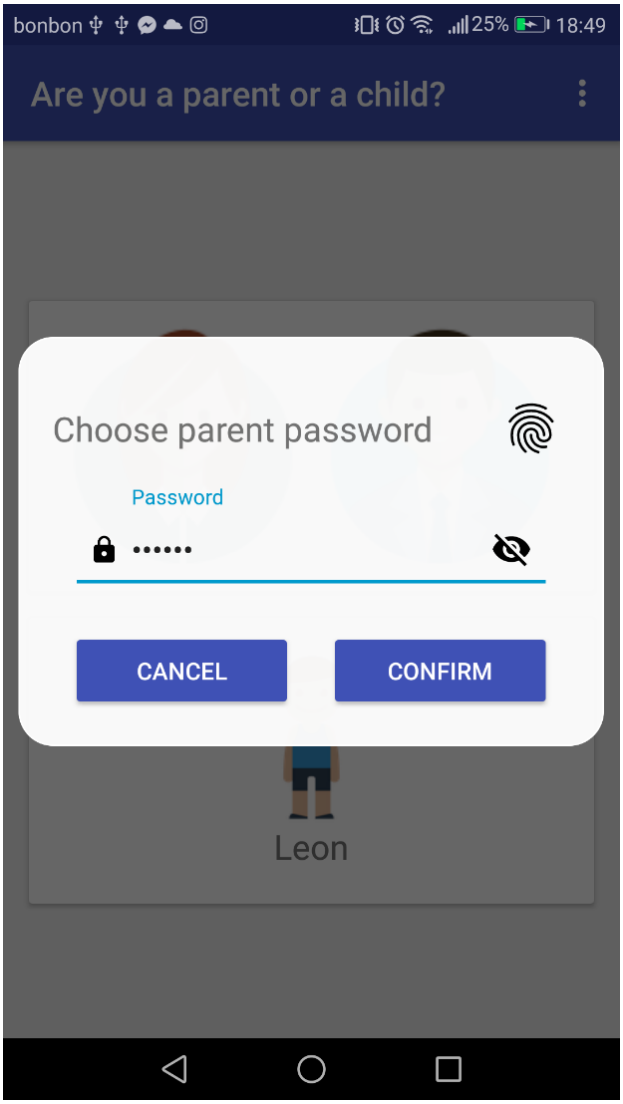
Slika 3.18. AAC sustav

Pritiskom na *Enter text* iz padajućeg izbornika gore desno moguće je direktno unijeti tekst i dodati ga na prikaz te reproducirati zajedno s drugim već postojećim frazama. Otvara se dijalog u koji upisujemo željeni tekst.



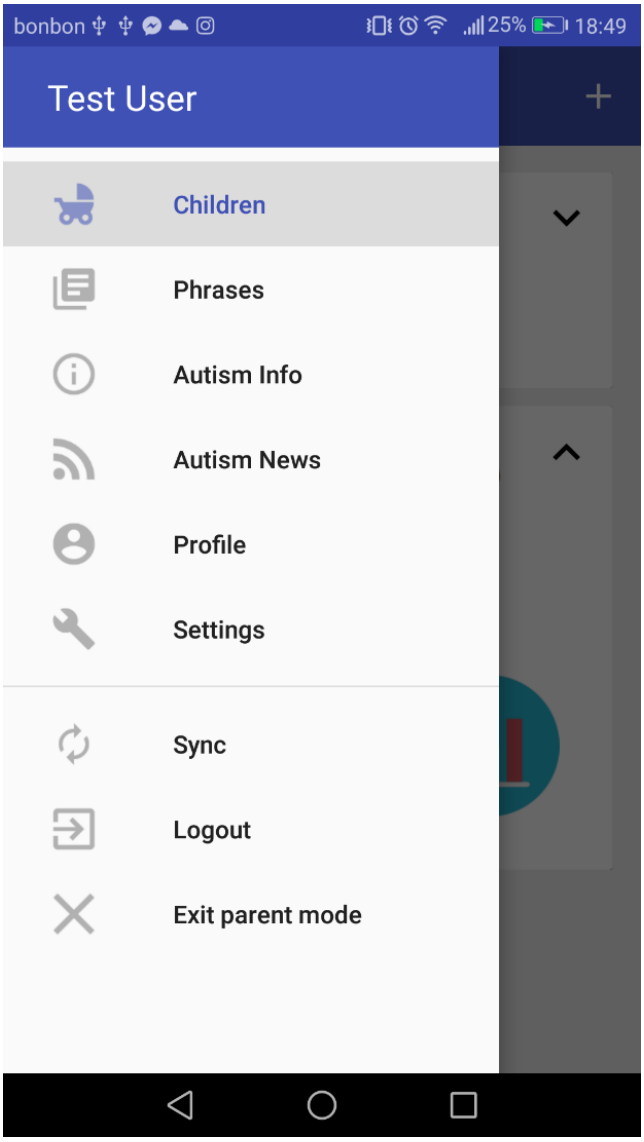
Slika 3.19. Dodavanje teksta direktno

Odabirom načina rada za roditelje iz *Main* prikaza potrebno je prvo autentificirati se kao roditelj. To je moguće upisivanjem prethodno odabrane lozinke ili pomoću otiska prsta. Ako je lozinka već postavljena traži se njen upis, a ako nije njeno postavljanje. Otisak prsta može se koristiti ako uređaj to podržava i već ima postavljen neki *Fingerprint ID*. Ako su ti uvjeti zadovoljeni u gornjem desnom kutu dijaloga prikazuje se odgovarajuća ikona. Kada korisnik postavi prst na senzor otiska prsta ikonica će pozeleniti i otvorit će se način rada za roditelje. Analogno vrijedi i za upis lozinke.



Slika 3.20. Autentifikacija roditelja

Otvaranjem načina rada za roditelje otvara se prikaz sa navigacijskim izbornikom s lijeve strane kojim se obavlja sva navigacija među pripadajućim prikazima. Moguće je vidjeti i upravljati dječjim profilima i frazama, čitati informacije i vijesti o autizmu, uređivati postavke profila i postavke aplikacije odabirom odgovarajućih elemenata izbornika. Nadalje moguće je sinkronizirati korisničke podatke s online bazom podataka, odjaviti se i izaći iz načina rada za roditelje.



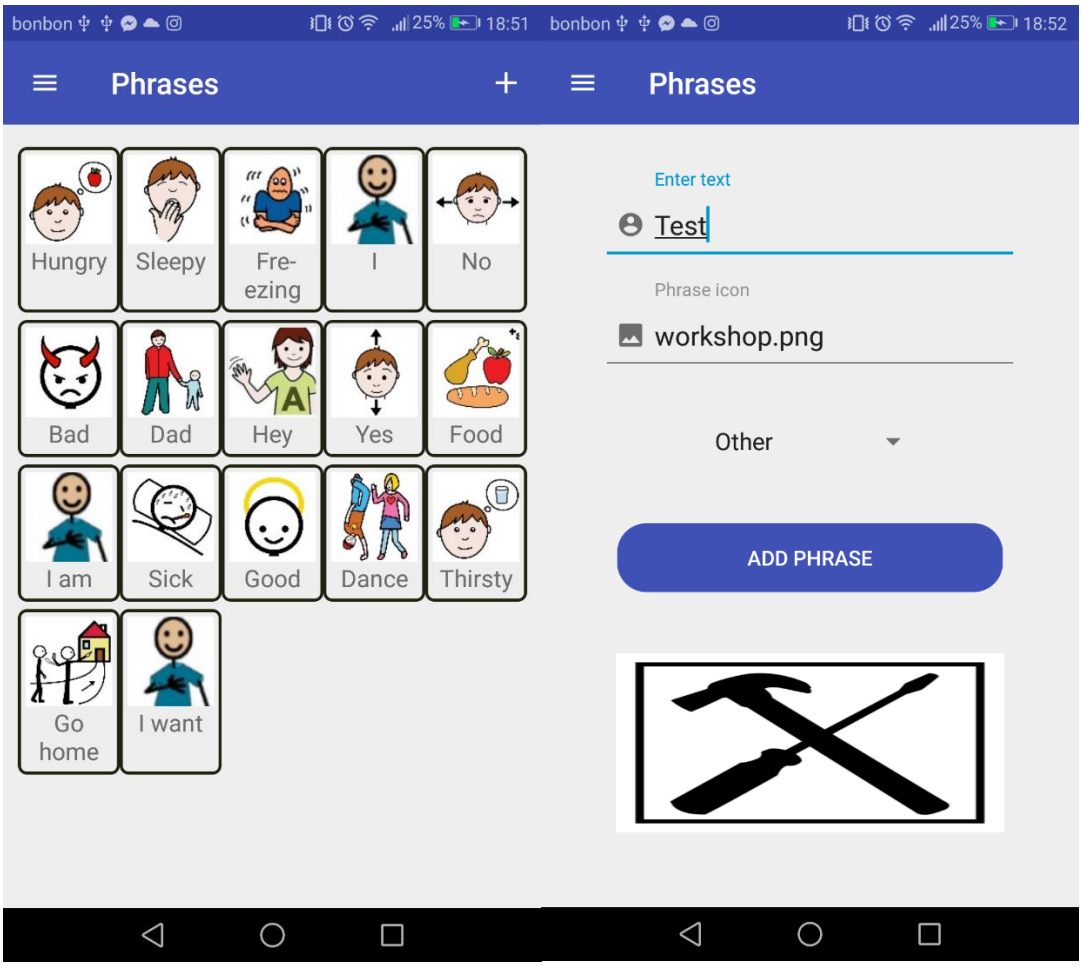
Slika 3.21. Navigacijski izbornik načina rada za roditelje

Prvi prikaz koji se otvara odmah u načinu rada za roditelje jest *Children*. Prikazuje se lista dječjih profila i moguće je proširivanjem elementa liste i odabirom odgovarajuće tipke obrisati profil, urediti ga i pregledati statistike za njega. Statistika prikazuje dva grafa, od kojih gornji prati prosječno vrijeme u igri djeteta, a donji broj pogrešaka po igri. Pritiskom na „+“ tipku u gornjem desnom kutu dodaje se dječji profil istim dijalogom kao i kod registracije. Također se taj dijalog otvara pri uređivanju podataka dječjeg profila s već popunjenim poljima.



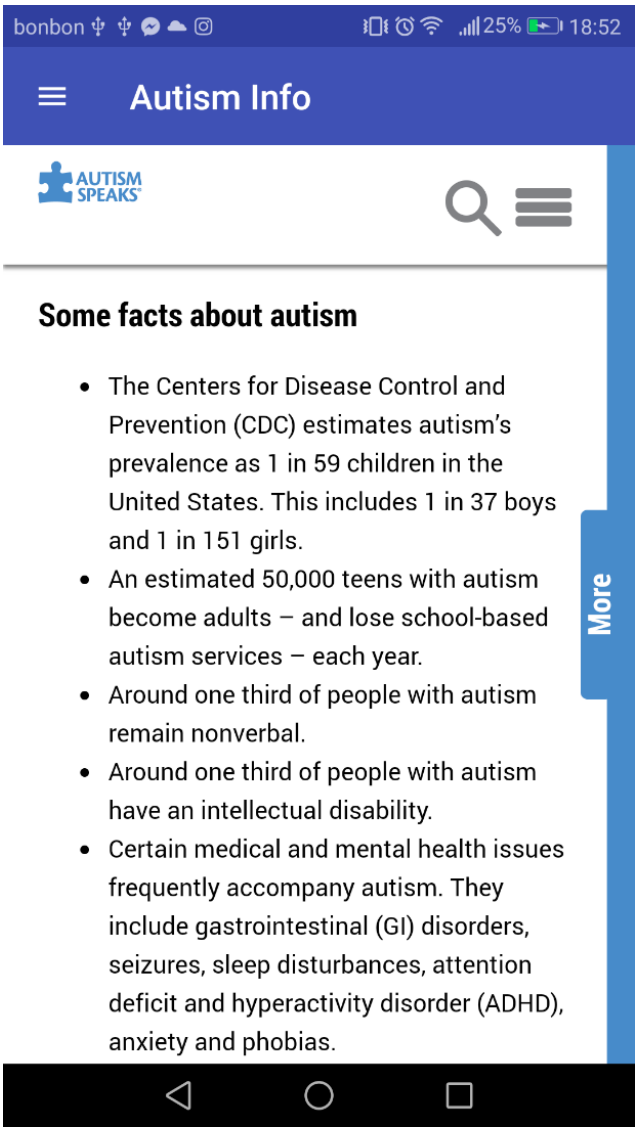
Slika 3.22. Children prikaz i statistika

Nadalje odabirom *Phrases* opcije izbornika otvara se *Phrases* prikaz. Na njemu su prikazane sve dostupne fraze, zajedno s onima dodanim lokalno. Dugim pritiskom na pojedinu frazu ona se može obrisati. Pritiskom na „+“ tipku u gornjem desnom kutu otvara se prikaz gdje treba ispuniti polja za unos teksta fraze, učitati odgovarajuću ikonu i odabrati kategoriju. Pritiskom na *Add Phrase* tipku fraza se dodaje u lokalnu bazu podataka, a ikona u lokalnu pohranu aplikacije.



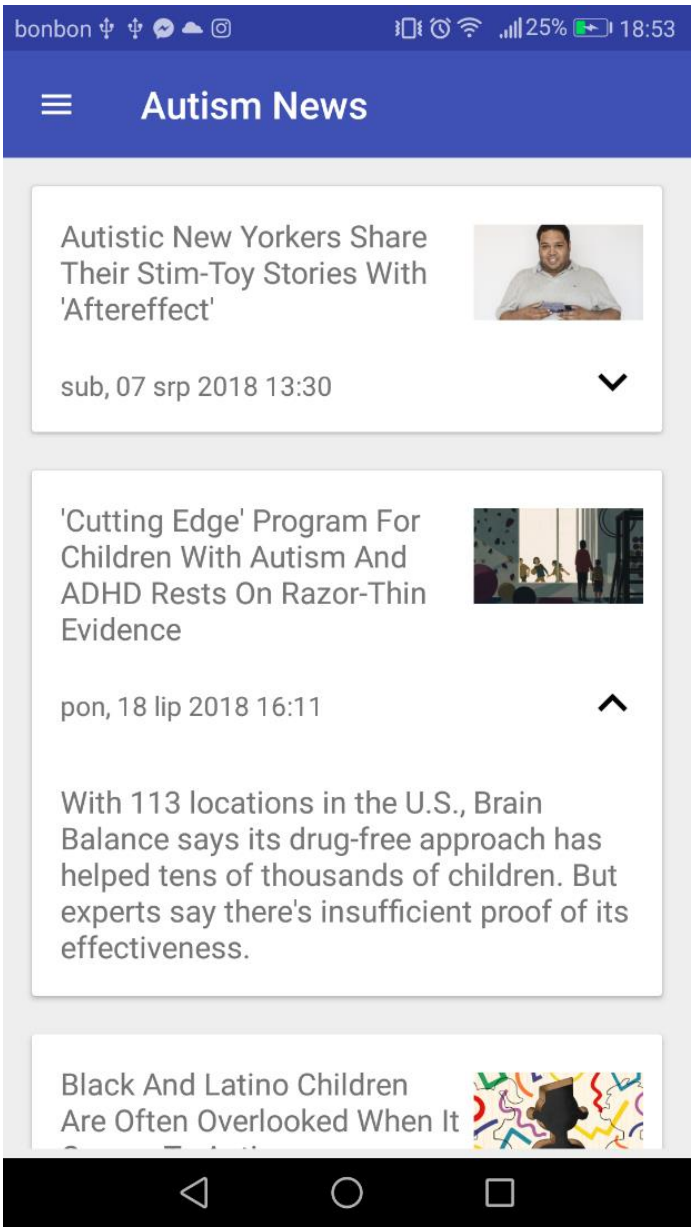
Slika 3.23. Phrases prikaz i dodavanje fraze

Odabirom *Autism Info* opcije u izborniku otvara se prikaz koji otvara Web stranicu *Autism Speaks* organizacije sa svim informacijama o autizmu i povezanim bolestima. Web stranica se otvara unutar aplikacije.



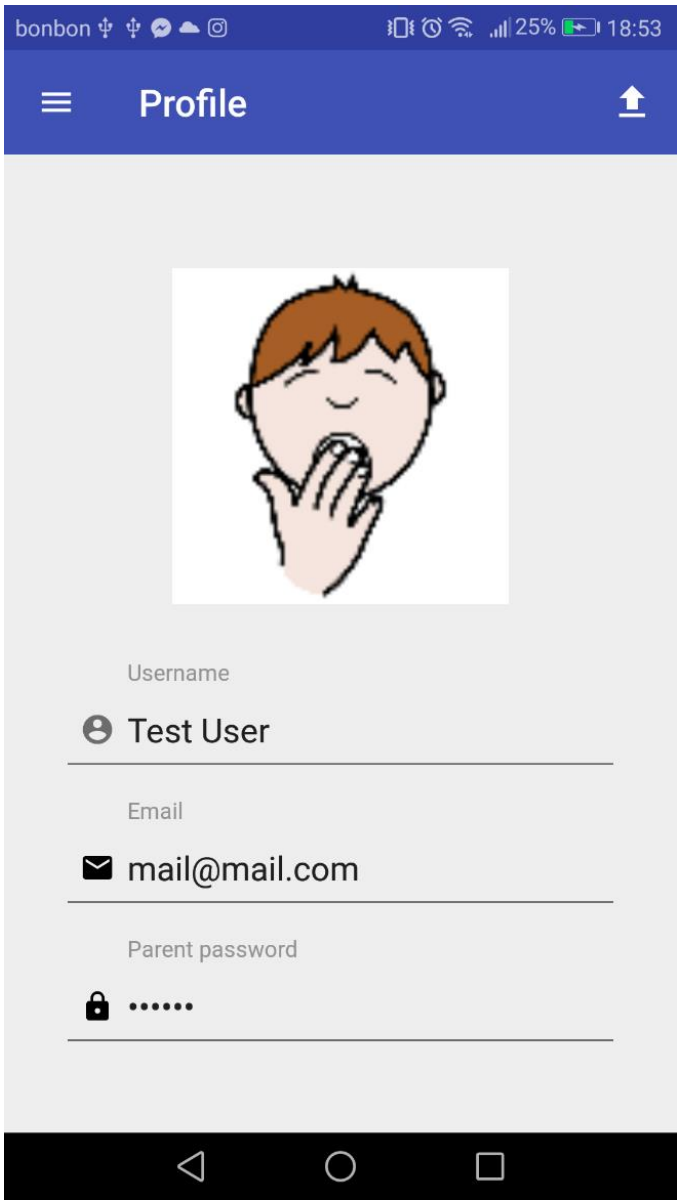
Slika 3.24. *Autism Info* prikaz

Autism News prikaz sadrži listu vijesti vezanih za autizam sa popularnog RSS servisa. Prikazani su naslov artikla, slika i datum objave. Proširivanjem elementa liste prikazan je detaljniji opis artikla a pritiskom na njega otvara se artikl u Web pregledniku.



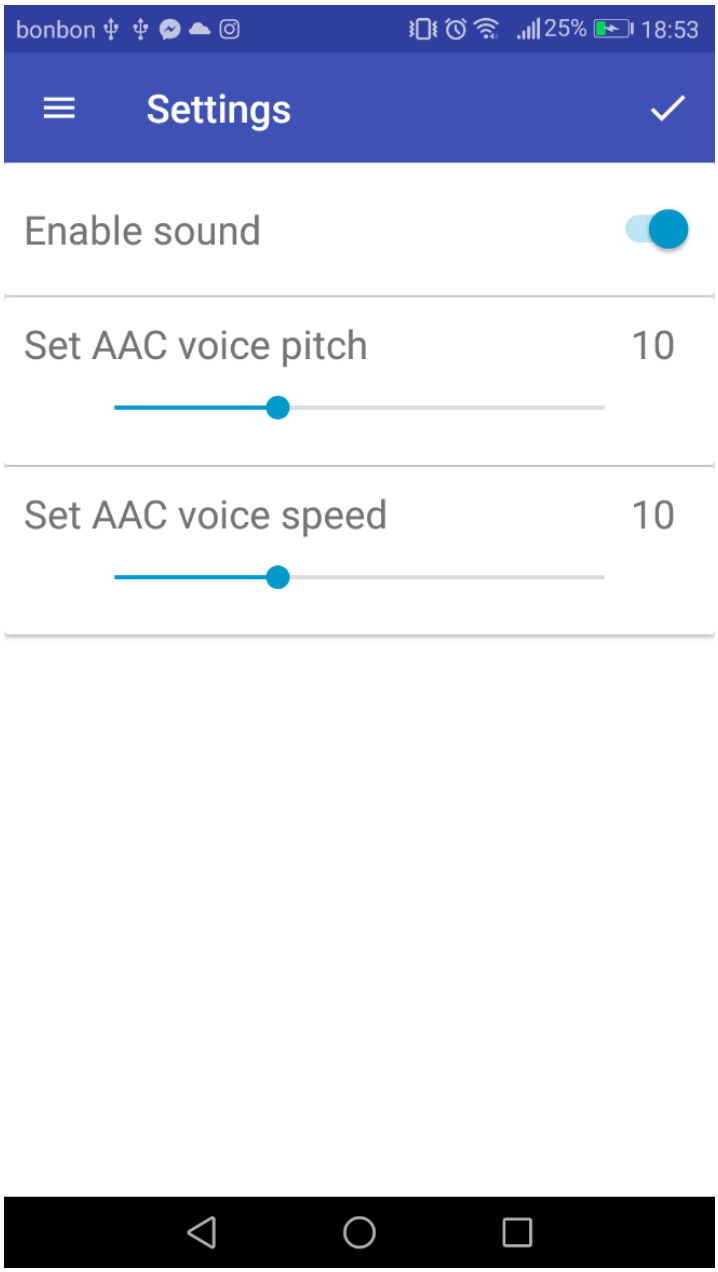
Slika 3.25. *Autism News* prikaz

Profile prikaz nudi korisniku uređivanje svog korisničkog imena, lozinke za način rada za roditelje i učitavanje profilne slike. Korisničko ime i lozinka upisuju se u odgovarajuća polja, a profilnu sliku moguće je učitati pritiskom na postojeću ili mjesto namijenjeno za nju. Kada se naprave nekakve promjene u gornjem desnom kutu pokazat će se ikonica za učitavanje promjena. Promjene će se spremiti u online i lokalnu bazu, a profilna slika učitati na lokalnu i online pohranu.



Slika 3.26. *Profile* prikaz

Postavke aplikacije mijenjaju se na *Settings* prikazu. Nudi se mogućnost uključivanja i isključivanja zvukova aplikacije te postavljanja brzine i visine glasa TTS sustava. Pritiskom na kvačicu u gornjem desnom kutu postavke se spremaju.



Slika 3.27. Settings prikaz

3.6. Testiranje implementacije

Kako bi završna softverska implementacija bila što kvalitetnija i kako bi se izbjegle pogreške pri budućim izmjenama koda, potrebno je tu implementaciju testirati. Jedan od najčešćih načina testiranja koje razvojni programeri mogu sami provesti jest pisanje *Unit* testova. To su testovi koji provjeravaju ponašaju li se pojedini dijelovi koda kako trebaju, tj. ima li logičkih pogrešaka. Također provjerava se utječe li dodavanje novog koda na izvođenje starih dijelova. Pisanje testova često dovodi do ranog otkrivanja pogrešaka te tako i smanjenjem troškova koje bi uzrokovali naknadni popravci. Potrebno je testovima pokriti što veći dio napisanog koda te logički odvojiti pojedine cjeline prema funkcionalnostima. Još neke vrste testova koje je moguće provoditi su integracijski i instrumentacijski testovi. Integracijski testovi provjeravaju kako se cjeline ponašaju u međusobnoj interakciji, a instrumentacijski su oni koji se moraju pokretati na uređaju. Nasuprot tome *Unit* testovi mogu se pokretati lokalno u instanci Java virtualnog stroja. U implementaciji ove aplikacije pisani su *Unit* testovi za Repository klase i ViewModele, a instrumentacijski za Dao klase Room baze podataka. Zbog testabilne i modularne arhitekture aplikacije pisanje testova bilo je jednostavno i nije zahtijevalo mnogo koda. Pri pisanju testova korištene su biblioteke JUnit 4 i Mockito te dodaci za testiranje nekih korištenih biblioteka. Primjeri testova dani su kodovima ispod. U kodu 3.6.1. prikazan je dio *UserDao* klase za koji je dan primjer testa, u kodu 3.6.2. instrumentacijski test za te funkcije. Kako bi se Room baza podataka mogla testirati stvara se implementacija u memoriji koja ne utječe na stvarnu implementaciju. Nakon toga se na generiranoj Dao klasi vrše definirani upiti te testiraju vrijednosti vraćene vrijednosti. U ovom slučaju testirano je umetanje *User* objekta u bazu te provjera nalazi li se isti takav jedinstven objekt u njoj. Pošto su Dao klase pisane tako da vraćaju RxJava objekte, možemo koristiti funkciju *test* te nad njima izvršiti *assertion* funkcije, tj. zahtijevati da dobivene vrijednosti zadovoljavaju postavljene uvjete za uspješnost testa.

```
@Dao
interface UserDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insert(user: User): Long

    @Query("SELECT * FROM $TABLE_USER limit 1")
    fun getCurrentUser(): Single<User>

}
```

Kod 3.6.1. Funkcije testirane Dao klase

```
@RunWith(AndroidJUnit4::class)
class UserDaoTest {

    private lateinit var db: AppDatabase
    private lateinit var userDao: UserDao

    private val testUser = TestDataGenerator.testUser()

    @Before
    override fun setUp() {
        db = Room
          .inMemoryDatabaseBuilder(
             InstrumentationRegistry.getContext(),
             AppDatabase::class.java
          ).allowMainThreadQueries()
           .build()

        userDao = db.userDao()
    }

    @Test
    fun userInsertsSuccessfully() {
         userDao.insert(testUser)

        userDao.getCurrentUser()
                .test()
                .assertValue(testUser)
    }
}
```

Kod 3.6.2. Primjer testa Dao klase Room baze podataka

Slijedeća kategorija klasa koje je bitno testirati jest ViewModel. Primjer je dan kodom 3.6.3. za *LoginViewModel*. Bitno je definirati *Rule* kako bi se asinkrone operacije izvele odmah i sinkrono kako bi se mogle testirati. Instancirajući *LoginViewModel* objekt potrebno je kao parametre predati testne implementacije korištenih klasa. Za to se koristi *Mockito* biblioteka. Deklariraju se potrebni objekti s *@Mock* anotacijom te se inicijaliziraju funkcijom *initMocks*. Koristi se *MockitoJUnitRunner* za pokretanje testova. Kada se stvore testne implementacije klasa, može se odrediti što će one vraćati kada se pojedina funkcija pozove nad njima. U ovom slučaju postavlja se da kada *LoginRepo* klasa poziva funkciju *isLoggedIn* vraća *User* objekt koji je stvoren unutar testa. Analogno *syncUserData* klase *SyncRepository* odmah vraća potvrdu o uspješnom završetku izvođenja. Na poslijetku, potrebno je pretplatiti testni *Observer* na LiveData objekt ViewModela. Nakon definiranja svih radnji pokreće se funkcija *checkLoggedIn* ViewModela te se provjerava poprimaju li LiveData objekti željene vrijednosti ispravnim redoslijedom.

```
@RunWith(MockitoJUnitRunner::class)
class LoginViewModelTest {

    @get:Rule
    var instantTaskExecutorRule = InstantTaskExecutorRule()

    @Mock private lateinit var loginRepo: LoginRepository
    @Mock private lateinit var dataRepo: DataRepository
    @Mock private lateinit var observer: Observer<Resource<User>>

    private lateinit var loginViewModel: LoginViewModel
    private val user: User = TestDataGenerator.testUser()

    @Before
    override fun setUp() {
         MockitoAnnotations.initMocks(this)

        loginViewModel = LoginViewModel(loginRepo, dataRepo)
        loginViewModel.resourceLiveData.observeForever(observer)
    }

    @Test
    fun loginCheckSuccessTest() {
        whenever(loginRepo.isLoggedIn()).thenReturn(Maybe.just(user))
        whenever(dataRepo.syncUserData()).thenReturn(Completable.complete())

        loginViewModel.checkLoggedIn()

        verify(observer).onChanged(Resource.loading())
        verify(observer).onChanged(Resource.success(user))
    }
}
```

Kod 3.6.3. Primjer testa ViewModela

Nadalje, bitno je testirati Repository klase jer su one direktna poveznica sa izvorima podataka za aplikaciju. Analogno s prethodnim primjerom potrebno je stvoriti testne implementacije korištenih klasa, postaviti *Rule* za sinkrono i instantno izvođenje te zadati akcije izvedene na pozive određenih funkcija. U ovom slučaju potrebno je stvoriti i testne *Scheduler* klase kako bi se izvođenje RxJava poziva u pozadinskoj niti moglo testirati. U ovom testu *Observer* pretplaćuje se na rezultat poziva. Pomoću *TestScheduler* klase okida se izvršenje svih naredbi, a *Observer* čeka završetak izvedbe pa provjerava ispravnost dobivenih podataka.

```
@RunWith(MockitoJUnitRunner::class)
class LoginRepoTest {

    @get:Rule
    var instantTaskExecutorRule = InstantTaskExecutorRule()

    @Mock private lateinit var auth: FirebaseAuth
    @Mock private lateinit var api: API
    @Mock private lateinit var db: AppDatabase
    @Mock private lateinit var prefsHelper: PrefsHelper
    @Mock private lateinit var userDao: UserDao

    private var testScheduler = TestScheduler()
    private var testObserver = TestObserver<User>()
    private val user = TestDataGenerator.testUser()

    private lateinit var loginRepo: LoginRepository

    override fun before() {
         MockitoAnnotations.initMocks(this)
        loginRepo = LoginRepository(auth, db, api, prefsHelper,
                                    testScheduler, testScheduler)
    }

    @Test
    fun dbReturnsUserLoggedInSuccess() {
        whenever(db.userDao()).thenReturn(userDao)
        whenever(db.userDao().userLoggedIn()).thenReturn(Maybe.just(user))

        loginRepo
                .isLoggedIn()
                .subscribe(testObserver)

        testScheduler.triggerActions()

        testObserver.awaitTerminalEvent()
        testObserver.assertValue(user)
    }
}
```

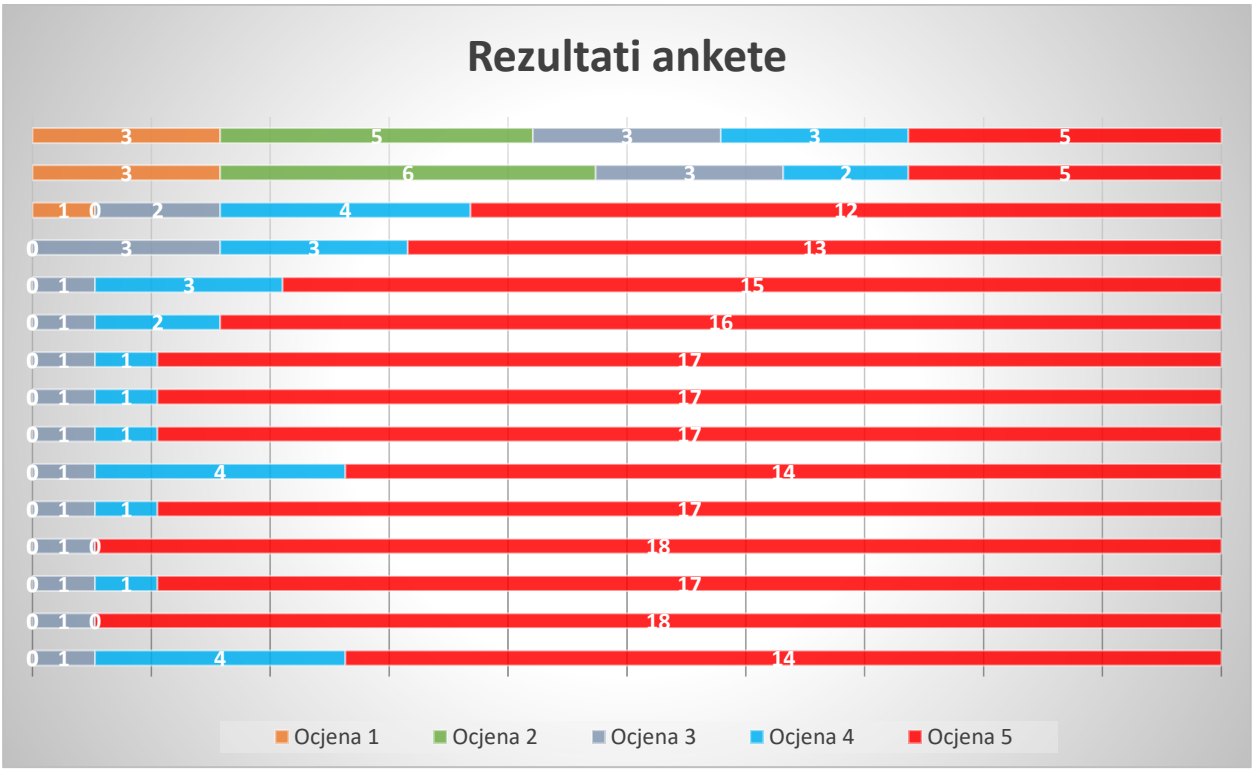
Kod 3.6.4 Primjer testa Repository klase

3.7. Anketa o uspješnosti implementacije

Kako bi se provjerila uspješnost softverske implementacija prema nekoj ideji potrebno je doznati mišljenja ciljnih skupina korisnika o njenoj korisnosti i kvaliteti. Iz ovog razloga aplikacija je dana na korištenje nekolicini osoba koje su naposljetku popunile anketu i izrazile svoje mišljenje. Anketa sadrži tvrdnje o kvaliteti implementacije te su osobe koje su je testirale odabirale razinu slaganja s navedenim tvrdnjama. Popis pitanja i rezultati ankete vidljivi su i komentirani u slijedećem dijelu. Provedena anketa imala je pitanja u tipu Likertove ljestvice, navodeći tvrdnje i tražeći od anketiranih osoba stupanj slaganja s tom tvrdnjom. Tvdnje su bile:

1. Potrebna su poboljšanja sučelja aplikacije.
2. Potrebna su logička poboljšanja.
3. Aplikacija radi stabilno i na očekivan način.
4. Aplikacija koristi dobre principe, ali postoji dosta mjesta za napredak.
5. Roditeljima su dostupne informacije o autizmu.
6. Roditelji mogu lako pratiti korištenje aplikacije djece.
7. Sadržaj AAC sustava je proširiv i prilagodljiv.
8. AAC sustav je lako koristiti i omogućava negovornu komunikaciju.
9. Sadržaj igara je zanimljiv djeci i pomaže mentalnom razvoju.
10. Sučelje je privlačno djeci i potiče ih na igru.
11. Načini rada za roditelje i djecu su dobro odvojeni i ne dopuštaju djeci promjene podataka.
12. Korištenje aplikacije je intuitivno i jednostavno.
13. Aplikacija može pomoći roditeljima s djecom oboljelom od autizma.
14. Aplikacija može pomoći djeci s autizmom u svakodnevnom životu.
15. Aplikacija je uspješna implementacija prvotne ideje.

Rezultati su dani za svako pitanje istim redoslijedom kojim su ona prethodno navedena i vidljivi su na slici 3.28.



Slika 3.28. Rezultati ankete o uspješnosti implementacije

Ispitano je 19 ljudi, od kojih nijedan nije član obitelji djeteta s autizmom. Zbog ovoga su rezultati ankete nepouzdani i ne mogu se primijeniti na situaciju korištenja aplikacije u stvarnom životu. Ipak, daju nekakav uvid u mišljenja o izvedenoj aplikaciji. Ocjene od 1 do 5 predstavljaju razine slaganja s navedenom tvrdnjom, gdje 5 predstavlja potpuno slaganje. Prema grafu može se zaključiti da su ispitani korisnici općenito zadovoljni izvedenom implementacijom aplikacije, ali potrebna su poboljšanja. Najviše se slažu s tvrdnjama da je aplikaciju intuitivno i lako koristiti te da ona zapravo može pomoći djeci s autizmom u svakodnevnom životu. S ovim tvrdnjama potpuno se slaže čak 94.7% ispitanika. Nadalje s nešto manjim postotkom od 89.5% slijede tvrdnje o iskoristivosti aplikacije za roditelje, sadržaju i odvojenosti načina rada za roditelje i djecu. U rasponu od 63.2% do 84.2% nalaze se tvrdnje o uspješnosti implementacije prvotne ideje, lakoći praćenja napretka djece i pristupu informacijama o autizmu, principima aplikacije, njenoj stabilnosti i sučelju. Najveća raznolikost mišljenja prisutna je kod tvrdnji o potrebnim poboljšanjima logike i sučelja. S ovim tvrdnjama potpuno se slaže 26.3% ispitanika, a potpuno ne slaže 15.8% njih.

4. ZAKLJUČAK

U radu je prvotno iznesen problem i postavljeni su ciljevi koje treba postići da bi se rješenje smatralo uspješnim. Dana je teorijska podloga o autizmu i dodatno pojašnjenje zašto je potrebno ovako nešto izvesti. Nadalje su opisane mogućnosti mobilne okoline za rješenje postavljenog problema i već postojeći alati koji ga već rješavaju. Na kraju su opisana struktura i način rada izvedenog rješenja i opisani alati koji su se koristili tijekom izvedbe.

Izrađena je Android aplikacija koja koristi principe ABA terapije za unaprjeđenje mentalnog učenja te AAC sustav za olakšavanje komunikacije s drugim osobama. Namijenjena je djeci mlađeg uzrasta oboljeloj od autizma i njihovim roditeljima. Korisnik, tj. roditelj se može prijaviti u aplikaciju i sinkronizirati svoje podatke. Sastoji se od načina rada za roditelje i za djecu. Roditelji mogu kreirati dječje profile, pratiti statistiku korištenja aplikacije tih dječjih profila, informirati se o autizmu, uređivati sadržaj i postavke. Djeca mogu igrati igre pogađanja boja i emocija, prebrojavati geometrijske likove i koristiti AAC sustav. Aplikacija je testirana softverski pisanjem testova te je o kvaliteti njene izvedbe anketirana skupina ljudi. Rezultati ankete pokazali su da općenito zadovoljstvo korisnika i potrebu za poboljšanjem logike i sučelja. Naime, nitko od korisnika nije u stvarnosti bio član obitelji djeteta s autizmom te je uzorak anketiranih ljudi malen pa se rezultati ne mogu primijeniti na situaciju korištenja aplikacije u realnim uvjetima.

Implementirana mobilna aplikacija dobar je temelj za razvoj ozbiljnijeg projekta koji će iskoristiti iste ili slične principe, ali proširiti sadržaj i mogućnosti. Moguća su razna unaprjeđenja poput dodavanja više kategorija igara i povećanja interakcije pri samom korištenju. Valjalo bi izvesti igre koje će prepoznati napredak djeteta i prilagoditi svoju težinu. Dobar prostor za unaprjeđenje bio bi dodavanje implementacija i drugih terapijskih pristupa. Sadržaj bi se trebao često ažurirati i ponovno privući korisnike. Sučelje treba biti vizualno atraktivnije i sadržavati više animacija, zvučnih povratnih informacija i sličnog.

LITERATURA

[1] Autism Society, *What is Autism*, Autism Society, (http://www.autism-society.org/what-is/), pristupljeno: 21.5.2018.

[2] The MNT Editorial Team, *What is Autism*, Medical News Today, (https://www.medicalnewstoday.com/info/autism), pristupljeno: 24.5.2018.

[3] Autizam SUZAH, *Općenito o autizmu,* Autizam SUZAH, (http://www.autizam-suzah.hr/ /index.php/autizam), pristupljeno: 24.5.2018.

[4] C. Bonnello, *I ran an autism survey. The results? Quite revealing.*, Autistic not weird, (http://autisticnotweird.com/survey/), pristupljeno: 24.5.2018.

[5] Autism Europe, *Prevalence Rate of Autism*, Autism Europe, (http://www.autismeurope.org/about-autism/prevalence-rate-of-autism/), pristupljeno: 26.5.2018.

[6] Centers for Disease Control and Prevention, *ASD Data & Statistics*, Centers for Disease Control and Prevention, (https://www.cdc.gov/ncbddd/autism/data.html), pristupljeno: 26.5.2018.

[7] T. Shimabukuro, S. Grosse, C. Rice, *Medical expenditures for children with an autism spectrum disorder in a privately insured population*, Journal of Autism and Developmental Disorders, br. 38, sv. 3, str. 546-552, 2008.

[8] L.A. Croen, J.K. Grether, C.K. Joshida, *Maternal Autoimmune Diseases, Asthma and Allergies, and Childhood Autism Spectrum Disorder,* Arch Pediatr Adolesc Med, br. 159, str. 151-157, 2005.

[9] Applied Behavior Analysis Edu, *What are the Health Problems that Co-Occur with Autism?*, Applied Behavior Analysis Edu, (https://www.appliedbehavioranalysisedu.org/what-are-the-health-problems-that-co-occur-with-autism/), 21.5.2018.

[10] Epilepsy Foundation, *Ketogenic Diet,* Epilepsy Foundation, (https://epilepsychicago.org/epilepsy/treatment/ketogenic-diet/), pristupljeno: 4.6.2018.

[11] C. Lamm, *Sleep,* Autism Speaks, (https://www.autismspeaks.org/family-services/health-and-wellness/sleep), pristupljeno: 4.6.2018.

[12] P. Gorrido, *Gastrointestinal Dysfunction in Autism: Parental Report, Clinical Evaluation, and Associated Factors,* Autism Research, br. 5, sv. 2, str. 101-108, 2012.

[13] Car Autism Roadmap, *Feeding Disorders,* Car Autism Roadmap, (https://www.carautismroadmap.org/feeding-disorders/), 6.6.2018.

[14] S. Hyman, *Autism Spectrum Disorder and Migraines*, Sydney Cognitive Development Centre, (https://scdcentre.com/autism-spectrum-disorder-migraines/), pristupljeno: 6.6.2018.

[15] S. Myers, C. Johnson, *Management of Children With Autism Spectrum Disorders*, Pediatrics Official Journal of the American Academy of Pediatrics, br. 120, sv. 5, str. 1163, 2007.

[16] AHRC New York, *What Is TEACHH?*, AHRC New York, (https://schools.ahrcnyc.org/teacch/), pristupljeno: 6.6.2018.

[17] J.J. Wood, A. Drahota, K. Sze, K. Har, D. A. Langer, *Cognitive behavioral therapy for anxiety in children with autism spectrum disorders*, br. 50, sv. 3, str. 224-234, J Child Psychol Psychiatry, 2009.

[18] J. Weiss, J. Baker, E. Butter, *Mental health treatment for people with autism spectrum disorder*, American Psychological Association, (http://www.apa.org/pi/disability/resources/publications/newsletter/2016/09/autism-spectrum-disorder.aspx), pristupljeno: 6.6.2018.

[19] K. Lofland, *The Use of Technology in Treatment of Autism Spectrum Disorders,* Indiana Resource Center for Autism, (https://www.iidc.indiana.edu/pages/the-use-of-technology-in-treatment-of-autism-spectrum-disorders), pristupljeno: 27.6.2018.

[20] Android Developers, *Understand the Activity Lifecycle,* Android, (https://developer.android.com/guide/components/activities/activity-lifecycle), pristupljeno: 30.6.2018.

[21] Android Developers, *Android Architecture Components*, Android, (https://developer.android.com/topic/libraries/architecture/), pristupljeno: 30.6.2018.

SAŽETAK

Tema rada bila je dati teorijsku podlogu o autizmu te iznijeti potrebu za implementacijom neke od terapijskih metoda pomoću mobilne okoline. Zadatak je bio izraditi vlastito rješenje ovog problema. Iznesene su mogućnosti implementacije tih terapijskih metoda mobilnom okolinom i opisani već postojeći alati iz ove kategorije. Postavljena je ideja i cilj za izradu vlastitog rješenja. Izvedena je Android aplikacija koristeći Kotlin programski jezik. Ona koristi elemente ABA terapije za igre pogađanja emocija, boja i prebrojavanje geometrijskih oblika. Ugrađen je i AAC sustav koji prikazuje fraze i reproducira ih. Aplikacija je podijeljena u načine rada za roditelje i djecu. Roditelji se mogu prijaviti i sinkronizirati korisničke podatke, pratiti statistiku djece, informirati se o autizmu, postavljati postavke aplikacije i dodavati sadržaj. Djeca mogu igrati gore spomenute igre te koristiti AAC sustav. U radu su dane upute za korištenje aplikacije te njena struktura i način rada. Vidljivi su dijagrami toka pojedinih modula aplikacija, a bitniji algoritmi pokazani su u obliku programskog koda. Opisano je testiranje aplikacije i anketiranje manjeg uzorka osoba o kvaliteti izvedbe.

Ključne riječi: Autizam, Android, Kotlin, mobilna okolina, terapijske metode

ABSTRACT

This thesis' theme was giving a theoretical background about autism and to bring up the need for mobile implementation of some autism therapy methods. The goal was to implement a custom solution to this problem. Mobile environment's possibilites to implement some of the therapy methods were given and already existing tools from this category described. The idea and goals were set for custom implementation. As a solution an Android application was made using Kotlin programming language. It uses elements of ABA therapy to implement emotion and color guessing, as well as geometry shape counting. Also, a custom AAC system was built in. It displays and reproduces certain phrases. The application was split into parent and child modes. Parents can log in and synchronize user dana, track child statistics, inform themselves about autism, set application preferences and add content. Children can play aforementioned games and use the AAC system. The thesis gives application usage manual and describes it's structure and way of functioning. The flow charts are given for the application's modules and more important algorithms are shown in the programming code form. Application testing and surveying a small sample of people about it's quality was described.

Keywords: Autism, Android, Kotlin, mobile environment, therapy methods

ŽIVOTOPIS

Marijan Novak rođen je 19.7.1994. godine u Virovitici. Upisao je Osnovnu Školu Josipa Kozarca u Slatini 2001. godine i završio 2009. godine. Iste godine upisuje smjer elektrotehnika u Srednjoj Školi Marka Marulića u Slatini. Završetkom srednje škole upisuje Elektrotehnički Fakultet u Osijeku 2013. godine. Opredjeljuje se za smjer Komunikacije i Informatika. Akademske godine 2015./2016. stječe zvanje sveučilišnog prvostupnika elektrotehnike.

MARIJAN NOVAK

_____

PRILOZI

1. „Primjena mobilne okoline u podršci pacijentima s autizmom“ u .docx formatu
2. „Primjena mobilne okoline u podršci pacijentima s autizmom“ u .pdf formatu
3. Izvorni kod programskog rješenja