Универзитет у Београду
Електротехнички факултет

Развој мобилне апликације за заказивање фризерских услуга

Дипломски рад

Ментор:
проф. др Захарије Радивојевић

Кандидат:
Павле Шаренац 2020/0359

Београд, октобар 2025.

САДРЖАЈ

1. Увод

Циљ овог рада је да понуди решење у виду *Android* мобилне апликације *BarberBooker,* која олакшава резервацију термина клијентима и фризерским салонима. Поред унапређења начина уговарања времена термина, апликација омогућује клијентима да ефикасно пронађу најбоље оцењене бербернице на жељеном подручју. Идеја за развој оваквог софтвера је проистекла из личног искуства са непрактичним начинима заказивања и проналаска одговарајућих фризерских услуга.

Сродне *Android* мобилне апликације које настоје да реше сличне проблеме већ постоје како за домаће (Србија и остале земље из региона попут Хрватске, Босне и Херцеговине и Црне Горе), тако и за глобално тржиште. Апликације намењене искључиво за домаће тржиште углавном нису достигле велики успех из више разлога: географска ограниченост, лош квалитет софтвера, као и недостатак жеље власника фризерских салона за променом начина свог пословања. Насупрот њима, постоје успешне апликације са више милиона преузимања, а намењене су за глобално тржиште. [1] [2] [3]

Допринос овог рада се може описати као успешно реализован минимално одржив производ, односно MVP (*Minimum Viable Product*). MVP представља почетну верзију производа за чији развој је потребно уложити најмање труда и времена тако да буду имплементиране функционалности које су најважније за основно коришћење. Овакав производ се испоручује првим клијентима како би се што раније добиле повратне информације на основу којих се итеративно ради на даљим унапређењима. Један од познатих примера производа који је продаван већ као MVP јесте *iPhone*. *Apple* је продавао ове телефоне иако они нису имали многе тривијалне функционалности: копирање и лепљење текста, 3G (*Third Generation*) интернет и електронску пошту. Овај приступ се много пута показао у пракси као далеко супериорнији од традиционалног перфекционистичког начина развоја који подразумева веома дуг период имплементације током ког клијенти немају прилику да пруже конструктивне критике произвођачима, већ им се производ испоручује тек када се уради све што је оригинално замишљено. [4]

У наставку ће бити обрађене следеће теме:

- Преглед постојећих решења – биће детаљно анализиране две *Android* апликације са сличном наменом као *BarberBooker*, при чему је једна за домаће тржиште, а друга за глобално.
- Преглед коришћених технологија – објаснiће се различити софтверски алати и концепти који су били коришћени при развоју апликације *BarberBooker*.
- Имплементација – биће обрађено како су претходно наведене технологије конкретно искоришћене при имплементацији апликације.
- Функционалности – описаће се све што развијена апликација пружа корисницима, а поменуће се и постојећа ограничења уз могућа побољшања.
- Закључак – биће резимирани доприноси овог рада уз навођење смерница за даља истраживања.

2. ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА

У овом поглављу биће изложене анализе две *Android* апликације које настоје да реше сличне проблеме као *BarberBooker*. Биће обрађена једна апликација која је за домаће тржиште (*SrediMe*), а затим и једна апликација за глобално тржиште (*Fresha*). Главни фокус у овим анализама ће бити на апликације које се односе на клијенте, док ће апликације за салоне бити укратко описане. Детаљно проучавање већ постојећих решења за конкретан проблем је изузетно важно за избегавање уобичајених грешака, а и за смишљање иновативних идеја које могу учинити нови производ јаком конкуренцијом на тржишту.

Обе апликације ће бити систематично обрађене кроз следеће теме:

- Функционалности – биће побројане и укратко описане све главне функционалности апликације.
- Корисничко искуство и интерфејс – биће анализирани UX (*User Experience*) и UI (*User Interface*), односно корисничко искуство и интерфејс у погледу дизајна, једноставности коришћења и слично.
- Предности апликације – биће закључено на основу описаних функционалности и рецензија са *Google Play* шта су главне карактеристике које позитивно издвајају апликацију у односу на остале.
- Недостаци апликације – биће закључено такође на основу описаних функционалности и рецензија са *Google Play* која су највећа ограничења апликације.
- Место на тржишту – биће закључено каква је генерална успешност апликације на циљном тржишту у односу на остала доступна решења.

На крају поглавља ће бити приказан резиме карактеристика обе апликације у виду табеле у којој ће сваки ред бити посвећен одређеној особини, а свака колона одговарајућој апликацији. Тако ће бити могуће ефикасно поређење апликација по било којој особини од интереса.

2.1. Мобилна апликација *SrediMe*

SrediMe је *Android* апликација која омогућава корисницима проналазак великог броја различитих услуга које обично пружају салони лепоте, као и ефикасно заказивање термина. Ова апликација је намењена за домаће тржиште и то конкретно за Србију, Хрватску, Црну Гору и Босну и Херцеговину. *SrediMe* тренутно има преко 50 000 преузимања и просечну оцену 4.4/5.0 за 388 рецензија. Прва верзија апликације је објављена на *Google Play* 29.5.2017. [1]

2.1.1. Функционалности

Главне функционалности апликације *SrediMe* су:

- Регистрација – могуће је направити налог искључиво уносом одговарајућих података.
- Пријава – могуће је пријавити се на свој налог уносом електронске поште и лозинке.
- Преглед сопственог профила – корисник може да прегледа податке које је унео при регистрацији.
- Ажурирање сопственог профила – корисник може да измени податке које је унео при регистрацији.
- Избор државе и града – пре почетка претраге салона и услуга потребно је да корисник одабере једну од четири понуђене државе (Србија, Хрватска, Црна Гора или Босна и Херцеговина), а затим и град у оквиру те државе. Овако је осигурано да корисник надаље претражује само салоне који се налазе у релевантном граду у којем се клијент налази.
- Претрага салона по називу – могуће је унети назив или почетни део назива салона на основу ког ће се излистати сви салони чија имена почињу таквим текстом.
- Претрага салона по услугама – могуће је одабрати неку од многобројних понуђених услуга, а и унети назив или почетни део назива услуге како би се пронашли сви салони који пружају такву услугу.
- Претрага салона за одабрану услугу на мапи – након одабира услуге, постоји могућност да се на *Google* мапи виде сви салони који пружају ту услугу. Ово је врло практичан начин да се брзо пронађе одговарајући салон на жељеној локацији.
- Филтрирање резултата претраге – могуће је извршити прецизнију претрагу по разним додатним критеријума као што су део града, датум и време жељеног термина, максимална цена услуге и слично.
- Сортирање резултата претраге – могуће је сортирати резултате претраге салона по просечној оцени, времену регистрације салона, удаљености од тренутне локације корисника и по препорукама од стране власника апликације које се додељују најбољим локалима.
- Преглед недавно посећених налога салона – на почетној страници су излистани профили салона које је корисник скоро посетио, па тако клијент може ефикасније да поново погледа салоне које је већ истраживао.
- Преглед слика исхода различитих услуга – могуће је прегледати слике које салони постављају као резултате својих услуга како би клијенти имали бољи увид у сам рад салона. Такође, корисник може да сачува слике које му се допадну у посебну листу која је названа *LookBook* у апликацији.
- Преглед свих актуелних попуста – могуће је прегледати све попусте који важе у датом тренутку за услуге у свим салонима.
- Преглед профила салона – могу се видети многобројне важне информације о сваком салону. Излистане су све услуге које салон пружа уз њихове описе, а може се видети и цена сваке услуге. Приказане су слике салона, радно време, локација, као и начин плаћања.
- Преглед свих рецензија салона – приказане су просечне оцене за сваку услугу коју салон пружа, као и појединачне оцене и коментари које су корисници остављали.

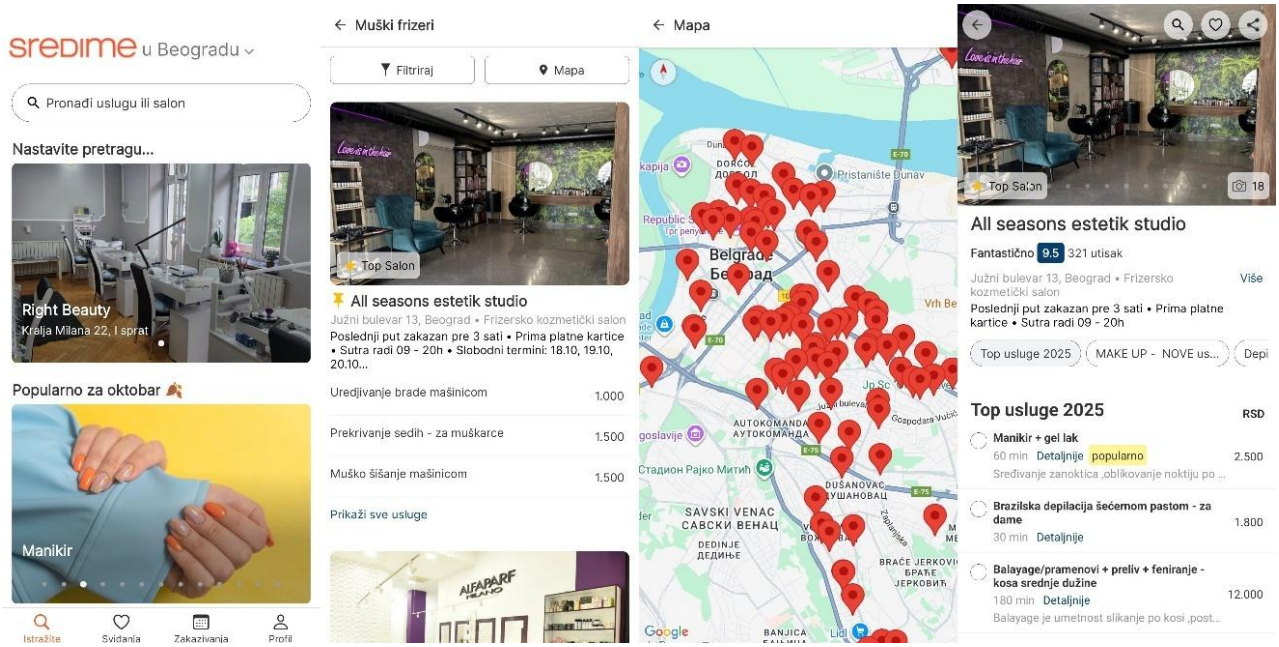
- Регуларно заказивање термина за једну или више услуга одједном – при прегледу профила салона, могуће је одабрати једну или више услуга за које се може одабрати неки од слободних термина и креирати захтев за резервацијом који салон мора да потврди да би термин био заиста резервисан.
- Инстант заказивање термина за једну или више услуга одједном – слична функционалност као и регуларно заказивање термина, при чему клијент не мора да чека салон да потврди захтев за резервацијом.
- Могућност бирања специфичног радника при резервацији – корисници могу да одаберу конкретног радника који ће им пружити услугу.
- Отказивање или промена времена термина – клијент може да потпуно откаже термин или само да промени време све до 24 часа пре почетка оригинално заказаног термина.
- Преглед свих заказаних термина у будућности – корисник може да види листу свих заказаних термина.
- Преглед архиве термина из прошлости – корисник може да види листу свих термина на којима је био.
- Остављање рецензија за салоне – корисник може оценити салоне у којима је био на бар једном третману.
- Додавање салона у листу омиљених – корисник при прегледу профила салона има опцију да дода салон у своју листу салона који му се највише допадају.

Треба истаћи да нерегистровани корисници имају могућност претраге салона. На овај начин се могу заинтересовати потенцијални корисници да направе налог након што виде понуду салона у апликацији. Главни разлог за прављење налога је коришћење функционалности попут заказивања термина и оцењивања салона.

2.1.2. Корисничко искуство и интерфејс

Корисничко искуство (UX) апликације је углавном добро зато што је апликација једноставна за коришћење. Корисник брзо може да пронађе салон на одговарајућој локацији са услугом која му је потребна. Постоји и много додатних функционалности које доприносе бољем корисничком искуству.

Кориснички интерфејс (UI) апликације није на завидном нивоу. Распоред елемената на страницама понекад изгледа неуредно и превише збијено. Одабрана палета боја при дизајну није претерано пријатна за око. Не постоји конзистентан стил који је искоришћен на свим страницама. На слици испод се може видети изглед четири странице из апликације.



Слика 2.1.3.1. Део корисничког интерфејса мобилне апликације *SrediMe*

2.1.3. Предности апликације

Позитивне карактеристике апликације *SrediMe* које је издвајају у односу на остале сличне апликације на домаћем тржишту су:

- Једноставност коришћења – интуитивно је пронаћи одговарајући салон и резервисати термин. Такође, коришћење осталих функционалности углавном није компликовано, тако да је целокупно корисничко искуство добро.
- Велики број корисних функционалности – корисничком искуству знатно доприноси велики број могућности које корисници имају у оквиру апликације. Поред основних функционалности попут претраге салона, заказивања термина и остављања рецензија, постоји много додатних опција као што се може видети у поглављу 2.1.2.
- Бесплатно коришћење – не наплаћује се нити преузимање апликације нити коришћење било којих функционалности. На овај начин је много већа шанса да се привуку нови корисници, као и да тренутни клијенти не престану да користе апликацију.
- Подршка за више држава – апликација покрива чак четири државе из региона – Србију, Хрватску, Црну Гору и Босну и Херцеговину.

2.1.4. Недостаци апликације

Неке од мана апликације *SrediMe* су:

- Недовољно добар кориснички интерфејс (UI) – као што је већ објашњено у поглављу 2.1.3, дизајн корисничког интерфејса није на нивоу квалитета светски познатих мобилних апликација.
- Немогућност онлајн плаћања кроз апликацију – нажалост, не постоји опција да корисник плати за свој термин у апликацији користећи на пример *Apple Pay*, *Google Pay* и сличне начине онлајн плаћања.

- Нетачност информација о салонима у апликацији – као што се може видети у рецензијама на *Google Play*, корисници су се често сусретали са проблемом тачности информација доступних у апликацији. Догађало се више пута да подаци које салони приказују у апликацији, попут листе слободних термина, нису истинити. [1]
- Немогућност пријаве преко *Google Sign-In* – клијенти су приморани да се региструју у самој апликацији ручним уносом својих података како би могли да користе апликацију као пријављени корисници. Када би био омогућен *Google Sign-In*, корисници би могли да се пријаве једним кликом у апликацији користећи свој већ постојећи *Google* налог, што би значајно допринело корисничком искуству.
- Ограниченост на домаће тржиште – апликација знатно ограничава број својих корисника зато што послује у само четири државе.
- Непостојање адекватне заштите салона од отказивања резервација – једини вид заштите је деактивација налога клијента уколико три пута откаже резервацију за исти салон. Боље би било када би се на пример захтевало од сваког клијента да унесе податке о бар једној својој кредитној картици како би могао новац аутоматски да се скине са ње уколико клијент не откаже резервацију у дозвољеном року. [1]

2.1.5. Место на тржишту

Са преко 50 000 преузимања и просечном оценом 4.4/5.0 за 388 рецензија на *Google Play*, апликација *SrediMe* је међу најпопуларнијим домаћим апликацијама за проналажење и заказивање услуга које нуде салони лепоте. Позитивни аспекти апликације истакнути у поглављу 2.1.4 очигледно остављају већи утисак на кориснике од недостатака описаних у поглављу 2.1.5, па се корисници из региона углавном одлучују баш за ову апликацију уколико желе да закажу себи термин за услугу која им је потребна. Такође, нема озбиљних конкурената који су искључиво фокусирани на домаће тржиште као ова апликација, тако да је то још један од разлога зашто људи са ових простора користе апликацију *SrediMe*. [1]

2.1.6. SrediMe PRO

SrediMe PRO је одвојена апликација од *SrediMe* која је намењена власницима салона и њиховим запосленима. Кроз ову апликацију салони уносе све потребне информације о њиховим услугама и слободним терминима. Такође, радници из салона могу видети свој распоред заказаних термина, а постоје и додатне функционалности које неће бити детаљно обрађене. [5]

2.2. Мобилна апликација *Fresha*

Fresha је *Android* апликација која омогућава клијентима широм света проналазак разноврсних услуга које пружају салони лепоте, као и једноставно заказивање термина. Ова апликација је намењена за глобално тржиште и тренутно се користи у преко 120 држава. [6] *Fresha* тренутно има преко 1 000 000 преузимања и просечну оцену 5.0/5.0 за 69 505 рецензија. Прва верзија апликације је објављена на *Google Play* 1.12.2017. [2]

2.2.1. Функционалности

Главне функционалности апликације *Fresha* су:

- Регистрација – могуће је направити налог уносом одговарајућих података.
- Пријава – могуће је пријавити се на свој налог уносом електронске поште и лозинке. Додатно, могуће је пријавити се користећи постојећи *Facebook* или *Google* налог, чиме корисници могу да избегну заморан поступак регистрације, а затим и пријаве ручним уносом својих података.
- Преглед сопственог профила – корисник може да прегледа податке које је унео при регистрацији.
- Ажурирање сопственог профила – корисник може да измени податке које је унео при регистрацији.
- Претрага салона за одабрану услугу на мапи – на страници за претрагу салона одмах након учитавања странице кориснику се аутоматски на горњој половини екрана на мапи приказују локали који су у близини његове тренутне локације, а на доњој половини екрана су такође излистани такви локали. Корисник може ефикасно да претражује локале на оба начина.
- Претрага салона по називу – могуће је унети назив или почетни део назива салона на основу ког ће се излистати сви салони чија имена почињу таквим текстом.
- Претрага салона по услугама – могуће је одабрати неку од многобројних понуђених услуга, а и унети назив или почетни део назива услуге како би се пронашли сви салони који пружају такву услугу.
- Филтрирање резултата претраге – могуће је извршити прецизнију претрагу по разним додатним критеријума као што су максимална цена, тип салона, могућност прихватања *Fresha* поклон картица и други параметри.
- Сортирање резултата претраге – могуће је сортирати резултате претраге салона по просечној оцени, близини и релевантности.
- Преглед недавно посећених налога салона – на почетној страници су излистани профили салона које је корисник скоро посетио, па тако клијент може ефикасније да поново погледа салоне које је већ истраживао.
- Преглед профила салона – могу се видети многобројне важне информације о сваком салону. Излистане су све услуге које салон пружа уз њихове описе, а може се видети и цена сваке услуге. Приказане су слике салона, радници уз њихову просечну оцену, рецензије, опис салона, радно време, локација, локали у близини тренутно посматраног салона, као и додатне информације.
- Остваривање попуста уз *Fresha* поклон картицу – корисници који заслуже Fresha поклон картицу могу унети у апликацији код са картице, па на тај начин могу платити мање за неку услугу за онолико новца колико има на поклон картици.
- Регуларно заказивање термина за једну или више услуга одједном – при прегледу профила салона, могуће је одабрати једну или више услуга за које се може одабрати неки од слободних термина и креирати захтев за резервацијом који салон мора да потврди да би термин био заиста резервисан.
- Инстант заказивање термина за једну или више услуга одједном – слична функционалност као и регуларно заказивање термина, при чему клијент не мора да чека салон да потврди захтев за резервацијом.
- Групне резервације – могуће је да се резервише групни термин за одређене услуге који подразумева пружање услуге више клијената у исто време.

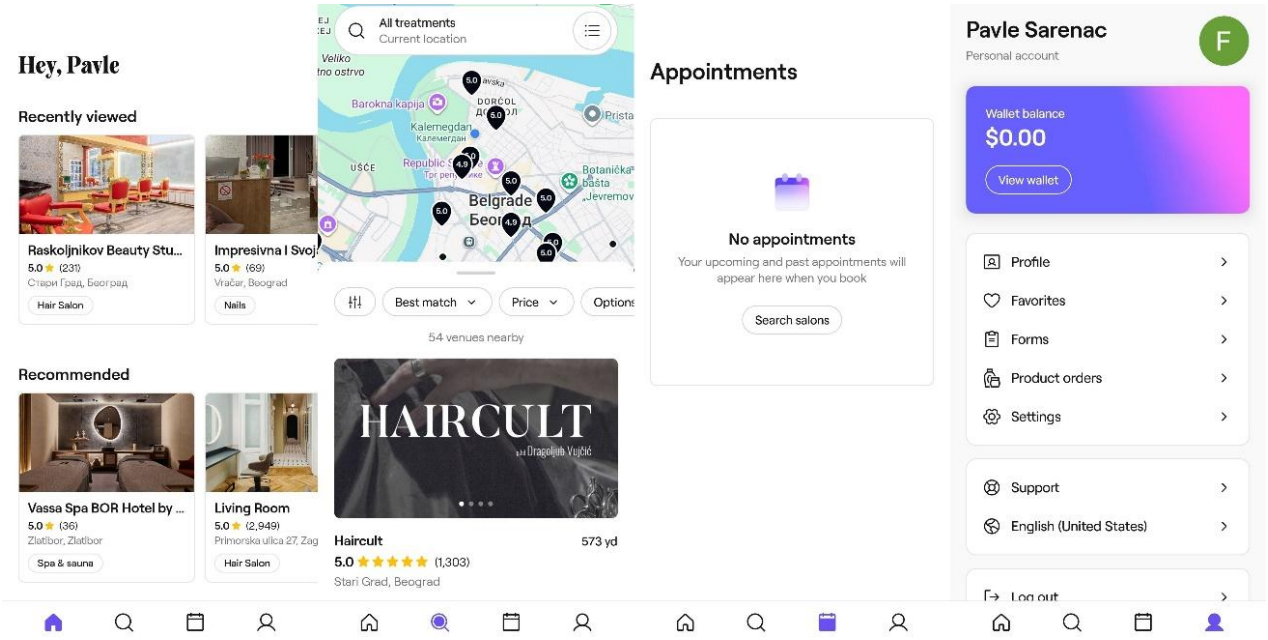
- Понављајуће резервације – клијент може да одједном резервише више термина уколико жели да редовно долази на исто место за исту услугу више пута у будућности.
- Могућност бирања специфичног радника при резервацији – корисници могу да одаберу конкретног радника који ће им пружити услугу.
- Отказивање резервације – клијент може да потпуно откаже термин, при чему ће у зависности од времена отказивања или проћи некажњено или ће му се са картице скинути одговарајућа количина новца. Салони могу дефинисати до када може да се откаже резервација без последица, као и да ли захтевају да клијенти унесу податке о бар једној својој картици у систем пре прављења резервације. На овај начин локали могу да се заштите од нерегуларних отказивања термина.
- Ажурирање детаља резервације – клијент може да ажурира разне детаље везане за већ потврђену резервацију: време, одабраног радника, тип услуге и друге.
- Могућност онлајн плаћања – корисник може унети податке о својим кредитним картицама како би их искористио за онлајн плаћање термина локалима који то дозволе.
- Преглед свих заказаних термина у будућности – корисник може да види листу свих заказаних термина.
- Преглед архиве термина из прошлости – корисник може да види листу свих термина на којима је био.
- Остављање рецензија за салоне – могуће оценити салоне у којима је био на бар једном третману.
- Оцењивање радника салона – клијент може да оцени и конкретног радника након пружене услуге.
- Додавање салона у листу омиљених – корисник при прегледу профила салона има опцију да дода салон у своју листу салона који му се највише допадају.

Треба нагласити да нерегистровани корисници немају могућност коришћења било које функционалности апликације. Обавезно је да се корисник или пријави користећи свој постојећи *Google* или *Facebook* налог, или да се региструје ручним уносом одговарајућих података, а затим пријави уносом своје електронске поште и лозинке. Тек након успешне пријаве клијент може користити све функционалности апликације.

2.2.2. Корисничко искуство и интерфејс

Корисничко искуство (UX) апликације је изузетно добро зато што је апликација врло једноставна и интуитивна за коришћење. Постоји много функционалности које позитивно утичу на општи утисак о апликацији. Најважније од свега, корисници могу веома брзо да нађу адекватан салон у својој тренутној близини и закажу термин.

Кориснички интерфејс (UI) апликације је на врхунском нивоу. Постоји конзистентан стил који је примењен на свим страницама. Свака страница изгледа уредно. Искоришћена комбинација боја је врло пријатна за око. На слици испод се може видети изглед четири странице из апликације.



Слика 2.2.3.1. Део корисничког интерфејса мобилне апликације *Fresha*

2.2.3. *Предности апликације*

Главне позитивне особине апликације *Fresha* су:

- Апликација *Fresha* поседује сличне предности, описане у поглављу 2.1.4, као и апликација *SrediMe*, при чему треба нагласити да *Fresha* није ограничена на четири земље из региона.
- Врхунски кориснички интерфејс – дизајн апликације је одлично одрађен, што доприноси пријатном осећају током коришћења.
- Могућност онлајн плаћања – клијенти могу да плате салонима директно у апликацији, што значајно унапређује корисничко искуство.
- Постојање заштите салона од отказивања резервација – у случају неадекватних отказивања резервација од стране клијената, одговарајућа количина новца ће бити скинута са његове картице и пребачена салону.
- Глобална присутност – апликација се користи у преко 120 држава широм света. [6]
- Могућност пријаве помоћу *Google/Faceboook Sign-In* – корисници могу да буду поштеђени ручног уноса података и регистрације тако што ће се пријавити на систем помоћу свог већ постојећег *Google* или *Facebook* налога.

2.2.4. *Недостаци апликације*

Главне мане апликације *Fresha* су:

- Присутност грешака у много кључних функционалности апликације – упркос већински позитивним искуствима људи, постоји известан број људи који је оставио веома негативне рецензије на *Google Play*. Такви људи су писали да су имали проблема при коришћењу основних функционалности попут онлајн плаћања, промене лозинке, отказивања резервација и других. [2]

- Вишеструке примедбе на брзину апликације – више људи је пријављивало у рецензијама на *Google Play* да је навигација кроз апликацију спора и да учитавање разних података често траје неприхватљиво дуго. [2]

2.2.5. Место на тржишту

Са преко 1 000 000 преузимања и просечном оценом 5.0/5.0 за 69 505 рецензија на *Google Play*, апликација *Fresha* је међу најпопуларнијим апликацијама на свету за проналажење и заказивање услуга које нуде салони лепоте. Један од главних глобалних конкурената који се такође истиче по броју преузимања и рецензија на *Google Play* јесте апликација *Booksy* која има преко 10 000 000 преузимања и просечну оцену 4.3/5.0 за 836 557 рецензија. [3] Што се тиче глобалне распрострањености, *Fresha* је далеко испред апликације *SrediMe*. Међутим, на домаћем тржишту *SrediMe* се више користи. Много више локала у Србији се може пронаћи у апликацији *SrediMe* него у апликацији *Fresha*.

2.2.6. Fresha for business

Fresha for business је одвојена апликација од *Fresha for customers* која је намењена власницима салона и њиховим запосленима. Кроз ову апликацију салони уносе све потребне информације о њиховим услугама и слободним терминима. Такође, радници из салона могу видети свој распоред заказаних термина, а постоје и додатне функционалности, као на пример аутоматско уметање заказаних термина у *Google Calendar* у виду догађаја. Детаљнија анализа ове апликације неће бити урађена. [7]

2.3. Табеларни преглед карактеристика апликација *SrediMe* и *Fresha*

Испод се може видети табела у којој су за апликације *SrediMe* и *Fresha* резимиране неке од најважнијих карактеристика ради лакшег поређења.

Табела 2.3.1. Упоредни преглед главних особина апликација *SrediMe* и *Fresha*

	SrediMe	***Fresha***
тржиште	домаће	глобално
број држава	4 [1]	преко 120 [6]
број преузимања	преко 50 000 [1]	преко 1 000 000 [2]
просечна оцена	4.4/5.0 [1]	5.0/5.0 [2]
број рецензија	388 [1]	69 505 [2]
број салона у Србији	велики	средњи
UX/UI	задовољавајуће	изузетно
регистрација/пријава	ручно	ручно и *Google/Facebook Sign-In*
функционалности нерегистрованог корисника	претрага салона	нема
претрага салона	по називу, услугама и на мапи	по називу, услугама и на мапи
операције над резултатима претраге	филтрирање и сортирање	филтрирање и сортирање
заказивање термина	регуларно и инстант	регуларно и инстант
типови резервација	за једног клијента	за једног или више клијената, као и

		понављајуће резервације
избор специфичног радника	да	да
отказивање резервације	да	да
заштита салона	није адекватна	добра заштита
ажурирање резервације	да	да
преглед будућих и прошлих термина	да	да
преглед профила клијента и салона	да	да
оцењивање салона	да	да
чување листе омиљених салона	да	да
онлајн плаћање	не	да
попусти/промотивне понуде	да	да
бесплатно коришћење	да	да
језик апликације	подржан само српски	подржано преко 20 језика
одвојена апликација за салоне	да, *SrediMe PRO* [5]	да, *Fresha for business* [7]

3. ПРЕГЛЕД КОРИШЋЕНИХ ТЕХНОЛОГИЈА

Једна од најважнијих пројектних одлука коју треба донети пре него што се крене у саму имплементацију софтвера јесте одабир одговарајућих технологија. Много различитих фактора треба узети у обзир, као што су: предзнање програмера, рок за израду апликације, сигурност, скалабилност, ефикасност и многи други. У овом поглављу прво ће бити укратко поменута развојна окружења коришћена за развој *Android* мобилне апликације *BarberBooker*. Након тога, биће детаљно описане кључне технологије коришћене за развој клијентског дела апликације, а затим и серверског.

3.1. Развојна окружења

У наставку ће се навести и укратко објаснити два интегрисана развојна окружења (на енглеском IDE – *Integrated Development Environment*) која су била коришћена при имплементацији рада: *Android Studio* (верзија *Android Studio Iguana*, 2023.2.1 *Patch* 1) за развој клијентског дела апликације и *IntelliJ IDEA* (верзија 2025.2.2, *Community Edition*) за развој серверског дела.

3.1.1. Android Studio

Android Studio је званично интегрисано развојно окружење за развој *Android* апликација. Заснован је на другом моћном интегрисаном развојном окружењу које се зове *IntelliJ IDEA*, а нуди и додатне функционалности које повећавају продуктивност приликом развоја *Android* апликација, као што су:

- флексибилан систем превођења програма заснован на *Gradle* алату
- ефикасан емулатор обогаћен многим функционалностима
- јединствено окружење за развој апликација за све *Android* уређаје
- *Live Edit* функционалност која омогућава ажурирање корисничког интерфејса у емулаторима и на физичким уређајима у реалном времену
- шаблони програмског кода за имплементирање уобичајених функционалности у апликацијама
- *GitHub* интеграција која олакшава увоз различитих програмских кодова
- исрпни алати за тестирање и радни оквири
- *Lint* алати за аутоматску детекцију проблема везаних за перформансе, компатибилност верзија библиотека и многих других
- подршка за *C++* програмски језик као и подршка за NDK (*Native Development Kit*) [8]

3.1.2. IntelliJ IDEA

IntelliJ IDEA је интегрисано развојно окружење које је примарно намењено за развој апликација у програмским језицима који се преводе у JVM (*Java Virtual Machine*) бајткод,

као што су *Java*, *Kotlin, Scala* и *Groovy*. Обезбеђена је висока програмерска продуктивност уз нагласак на приватност и безбедност података. Функционалности попут интелигентног предлагања кода, статичке анализе програма и аутоматизованог рефакторисања значајно доприносе корисничком искуству софтверских инжењера. [9]

3.2. *Android* клијент

У овом поглављу ће бити обрађене главне технологије које су коришћене при развоју клијентског дела мобилне апликације *BarberBooker*. Биће објашњен *Kotlin* програмски језик, асинхроно програмирање уз *Kotlin* корутине, различите *Android Jetpack* технологије, концепт *Dependency Injection* примењен уз *Hilt*, као и мрежна комуникација примењена уз *Retrofit*.

3.2.1. Kotlin програмски језик

Kotlin је статички типизиран програмски језик који има подршку како за објектно оријентисано програмирање, тако и за функционално програмирање. *Kotlin* има сличну синтаксу и концепте као и други програмски језици попут *C#*, *Java*, *Scala* и многи други. Постоји више варијанти Kotlin програмског језика које циљају различита окружења: JVM (*Kotlin*/JVM), *JavaScript* (*Kotlin*/JS (*JavaScript*)) и изворни машински код (*Kotlin*/*Native*). [10]

Kotlin програмски језик је одабран за овај рад зато што је *Google* још 2019. године објавио да ће све будуће *Android* библиотеке и докуменатција примарно бити намењене за овај језик. [11] Коришћена је верзија 1.9.0 *Kotlin* програмског језика.

3.2.2. Асинхроно програмирање уз Kotlin корутине

Већ деценијама програмери су суочени са проблемом блокирајућих операција унутар софтвера. Увек је жеља да се избегне ситуација у којој корисник не може да настави са коришћењем апликације док се неки процес не заврши, као и да се избегне стварање уских грла која би знатно ограничила скалабилност апликације. Ови проблеми се решавају асинхроним програмирањем које омогућава конкурентно извршавање више операција, што значајно побољшава корисничко искуство. [12]

Најпознатији начин конкурентног програмирања јесте коришћењем нити, које представљају независне токове контроле који се извршавају унутар процеса којим управља оперативни систем. Међутим, нити захтевају већу количину ресурса, па креирање великог броја нити може проузроковати озбиљне проблеме са перформансама. [13]

Програмски језик *Kotlin* има одличну подршку за ефикасно конкурентно програмирање. Користе се корутине које омогућавају писање асинхроног програмског кода на исти начин као што се пише и секвенцијални код. Једина разлика је што је потребно користити суспендујуће функције које омогућавају да се извршење текуће функције заустави како би се извршила нека друга функција, по чијем завршетку се наставља извршење оригиналне функције од истог места где је претходно била заустављена. Функција постаје суспендујућа једноставним додатком кључне речи *suspend* испред потписа функције. [13]

Насупрот нитима, корутине не захтевају много ресурса, због чега оне представљају лагану алтернативу нитима. Суспензија корутина не блокира системске ресурсе, па су зато корутине веома ефикасан вид конкурентног програмирања. Такође, промену контекста код корутина програмер врши експлицитно у коду позивом друге суспендујуће функције. С друге стране, промену контекста код нити врши оперативни систем кроз одговарајући системски позив, што је знатно спорије. [13]

Из свих наведених разлога, корутине су коришћене при имплементацији апликације *BarberBooker*. Коришћена је верзија 1.7.3 библиотеке *org.jetbrains.kotlinx:kotlinx-coroutines-android*.

3.2.3. Android Jetpack технологије

У овом делу ће бити наведене неке од најважнијих *Android Jetpack* технологија које су коришћене при изради мобилне апликације *BarberBooker*.

i) Jetpack Compose

Jetpack Compose је модеран скуп алата за имплементацију корисничких интерфејса у *Android* апликацијама. Ови алати омогућавају ефикасну изградњу комплексних апликација са мало кода. Неке од *Jetpack Compose* библиотека које су коришћене у овом раду су:

- *androidx.activity:activity-compose* (верзија 1.8.2)
- *androidx.compose.ui:ui* (верзија одређена преко *Compose* BOM (*Bill of Materials*) 2023.08.00)
- *androidx.compose.ui:ui-graphics* (верзија одређена преко *Compose* BOM (*Bill of Materials*) 2023.08.00)
- *androidx.compose.ui:ui-tooling* (верзија одређена преко *Compose* BOM (*Bill of Materials*) 2023.08.00)
- *androidx.compose.ui:ui-tooling-preview* (верзија одређена преко *Compose* BOM (*Bill of Materials*) 2023.08.00)
- *androidx.compose.material3:material3* (верзија одређена преко *Compose* BOM (*Bill of Materials*) 2023.08.00)
- *androidx.compose.material:material-icons-extended* (верзија 1.7.0-*beta*04)

Compose BOM омогућава да програмери мање брину о међусобној компатибилности између различитих верзија *Compose* библиотека. Уместо да прецизирају конкретне верзије сваке библиотеке у свом пројекту, софтверски инжењери могу само да назначе верзију *Compose* BOM коју желе да користе и то ће аутоматски обезбедити да све библиотеке које користе исту верзију *Compose* BOM буду компатибилне. [14]

ii) Lifecycle

Lifecycle представља класу која садржи информације о животу неке компоненте и омогућава осталим објектима да сазнају тренутно стање те компоненте и реагују у складу с тим. Овим се обезбеђује синхронизација између извршавања операција различитих објеката. Такође, код је боље организован, више читљив и лакши за одржавање уколико се исправно користе овакве компоненте које су свесне стања других компоненти. [15] У овом раду коришћена је верзија 2.7.0 библиотеке *androidx.lifecycle:lifecycle-runtime-ktx*.

iii) ViewModel

ViewModel је класа која је намењена за енкапсулацију кода који се тиче логике за приказ података, као и за чување стања на нивоу једног екрана у апликацији. Главна предност ове класе јесте то што чува стање екрана чак и након конфигурационе измене попут ротирања екрана. Ово је важно јер би значајно погоршало перформансе апликације уколико би морали да се нпр. поново дохватају подаци из базе података у оваквим ситуацијама. [16] У овом раду коришћена је верзија 2.8.3 библиотеке *androidx.lifecycle:lifecycle-viewmodel*.

iv) Навигација

Навигација се односи на акције којима корисници могу да мењају тренутно приказани садржај апликације. Могући су разни начини навигације унутар *Android* апликације: притиском на дугме, коришћењем менија који се може појавити извлачењем попут фиоке, као и притиском на траке које могу бити на врху или дну екрана. [17] У овом раду коришћена је верзија 2.8.0-*beta*04 библиотеке *androidx.navigation:navigation-compose*.

3.2.4. Dependency Injection уз Hilt

Dependency Injection, тј. аутоматско инстанцирање зависности, је добро познат шаблон који се користи у многим апликацијама. Овај концепт подразумева да, уколико је класа зависна од објекта неке друге класе, *Dependency Injection* механизам ће обезбедити да се на одговарајуће место достави потребан објекат у време извршавања програма, што ослобађа програмера од ручног инстанцирања објеката. Програмски код у ком се исправно користи *Dependency Injection* је реупотребљив, једноставан за рефакторисање, као и једноставан за тестирање. [18]

Hilt је библиотека која омогућава *Dependency Injection* у *Android* апликацијама. *Hilt* библиотека је изграђена на основу друге популарне библиотеке која се зове *Dagger*. То је случај зато што *Dagger* обезбеђује тачност у време превођења програма, добре перформансе током извршавања програма, скалабилност, као и подршку за *Android Studio*. [19] Библиотеке коришћене у овом раду су *hilt-android* (верзија 2.51.1), *hilt-compiler* (верзија 2.51.1), *androidx-hilt-navigation-compose* (верзија 1.2.0).

3.2.5. Мрежна комуникација уз Retrofit

Retrofit је библиотека која омогућава да *Android* апликације остваре комуникацију преко мреже слањем HTTP (*HyperText Transfer Protocol*) захтева, а затим и пријемом HTTP одговора. Замишљено је да се API (*Application Programming Interface*) *Android* апликације дефинише у виду интерфејса у којима се наводе потписи метода заједно са одговарајућим анотацијама. Могуће је слати како синхроне, тако и асинхроне HTTP захтеве. У овом раду је коришћена верзија 2.9.0 библиотеке *com.squareup.retrofit2:retrofit*.

3.3. *Ktor* сервер

У овом поглављу ће бити обрађене главне технологије које су коришћене при развоју серверског дела мобилне апликације *BarberBooker*. Биће објашњен *Ktor* радни оквир, технологије везане за базу података, аутентикација уз JWT, интеграције са спољним сервисима, као и контејнеризација уз *Docker*.

3.3.1. Ktor радни оквир

Ktor и *Spring Boot* су два најчешће коришћена радна оквира за имплементацију серверског дела *Android* апликација. Оба радна оквира подржавају програмски језик *Kotlin*. Зато се природно намеће питање, који радни оквир употребити за развој апликације попут *BarberBooker*?

Прво, *Ktor* је у потпуности имплементиран у програмском језику *Kotlin*, због чега природно има подршку за *Kotlin* корутине за разлику од *Spring Boot* радног оквира. Такође, *Ktor* радни оквир има доста мање подразумеваних конфигурација и зависности због чега је лаганији за покретање у односу на *Spring Boot*. С једне стране, ово пружа програмерима

бољи увид у подешавање своје апликације јер су у обавези да изврше већи број ручних конфигурација. С друге стране, ово програмерима ствара додатан посао при иницијалној конфигурацији сервера. Што се тиче зрелости радних оквира, *Spring Boot* ту дефинитивно предњачи зато што је много распрострањенији од *Ktor* радног оквира, те самим тим има и доста више документације, туторијала и других видова подршке на интернету. Обично је препоручљивије да се искористи *Ktor* радни оквир уколико је неопходно развити серверску страну некакве мање апликације, а *Spring Boot* уколико се развија велика и комплексна апликација. Из свих ових разлога, у овом раду је искоришћен *Ktor* радни оквир, и то верзија 2.2.1. [20]

3.3.2. База података

У овом делу ће бити објашњене све технологије које се у коришћене за перзистенцију података у апликацији *BarberBooker*. Биће обрађени *MySQL* сервер базе података, *MySQL Workbench* алат за визуелни преглед података, JDBC (*Java Database Connectivity*) технологија за конекцију сервера са базом, као и *HikariCP* библиотека за побољшање перформанси операција над базом података.

i) MySQL сервер базе података

MySQL је систем за управљање релационим базама података (на енглеском RDBMS (*Relational Database Management System*)) који омогућава чување и ефикасно организовање података. Овај систем се веома често користи због своје поузданости, перформанси, скалабилности и лакоће коришћења. Многе добро познате апликације са огромним бројем корисника користе баш *MySQL*: *Facebook*, *Netflix*, *Uber*, *Airbnb*, *Shopify*, *Booking.com* и друге. [21]

Широка распрострањеност и подршка за *MySQL*, као и лакоћа коришћења су били довољни разлози да се прибегне коришћењу баш ове базе података за апликацију *BarberBooker*. Коришћена је верзија 8.0 *MySQL* сервера.

ii) MySQL Workbench

MySQL Workbench је алат који представља графички кориснички интерфејс (на енглеском GUI (*Graphical User Interface*)) који се може искористити за једноставно повезивање на *MySQL* сервер и извршавање SQL (*Structured Query Language*) упита над базама података које су на том серверу. [22] При развоју мобилне апликације *BarberBooker* коришћена је верзија 8.0.34 *MySQL Workbench* алата.

iii) JDBC

JDBC API је индустријски стандард за повезивање апликација написаних у *Java* програмском језику са различитим SQL базама података, као и за извршавање SQL упита из апликација над тим базама. [23] С обзиром да је серверски део апликације *BarberBooker* написан у програмском језику *Kotlin* који је потпуно компатибилан са програмским језиком *Java*, JDBC је био очигледан избор за повезивање сервера са базом података у овом раду. Коришћена је класа *com.mysql.cj.jdbc.Driver* из верзије 8.0.30 библиотеке *mysql:mysql-connector-java*.

iv) HikariCP

HikariCP је библиотека која додатно побољшава перформансе операција над базама података. Ова библиотека садржи велики број микро-оптимизација које су појединачно посматрано занемарљиве. Међутим, све ове оптимизације заједно значајно доприносе перформансама при раду са базама података. Имплементација *HikariCP* библиотеке

представља напор да се смањи број бајткод инструкција у свим критичним деловима кода како би свака нит могла што више посла да стигне да обави у току свог додељеног процесорског времена које одређује распоређивач нити на датом оперативном систему. Ово је важно зато што се овако избегавају будући промашаји при приступу кеш меморији који су практично неминовни ако распоређивач преотме процесор од нити, а да она није завршила целу започету операцију. [24]

Коришћењем ове библиотеке одржава се скуп отворених конекција путем JDBC са базом података, при чему се може дефинисати тачан број жељених конекција. У овом раду коришћена је верзија 5.0.1 библиотеке *com.zaxxer:HikariCP*.

3.3.3. Аутентикација уз JWT (JSON Web Tokens)

JWT (JSON *(JavaScript Object Notation) Web Token*) је отворени стандард, прописан документом RFC (*Request for Comments*) 7519, који представља компактан и самодовољан начин да се безбедно размене информације између две стране у виду JSON објеката. [25]

JWT токени могу да пруже безбедност у комуникацији пружањем аутентикације корисника, тј. ако једна страна прими некакву поруку од друге стране, она може да провери да ли је заиста очекивани корисник послао ту поруку. Аутентикација је могућа захваљујући дигиталним потписима које је могуће верификовати. [25]

Један начин је да пошиљалац потпише JWT тако што ће уз своју поруку послати и вредност која се добије када се израчуна HMAC (*Hash-based Message Authentication Code*) SHA256 (*Secure Hash Algorithm 256-bit*) хеш функција за улаз који се добије комбиновањем садржаја JWT токена и тајног кључа који знају само пошиљалац и прималац. Прималац онда може верификовати овај потпис тако што ће и он израчунати вредност која је излаз HMAC SHA256 хеш функције за исти улаз који је искористио и прималац. Овај начин аутентикације је безбедан уколико тајни кључ знају искључиво пошиљалац и прималац. Баш овај алгоритам је искоришћен при имплементацији JWT аутентикације у апликацији *BarberBooker*. [25]

Други вид дигиталног потписа је могућ уз коришћење асиметричних криптографских алгоритама попут RSA (*Rivest-Shamir-Adleman*) или ECDSA (*Elliptic Curve Digital Signature Algorithm*). [25] Ови алгоритми су асиметрични зато што се користе различити кључеви при потпису и верификацији. За креирање дигиталног потписа користи се приватни кључ који зна само пошиљалац, а за верификацију тог потписа се користи јавни кључ који може свако да сазна.

Приватни и јавни кључеви се генеришу у пару и циљ је генерално да се искористи одговарајући математички апарат како би се осигурало да нападачу буде тешко или немогуће да сазна приватни кључ на основу јавног. RSA алгоритам се ослања на чињеницу да је, с једне стране, веома лако помножити два велика проста броја, док је с друге стране веома тешко сазнати на основу добијеног броја која су то два проста броја његови чиниоци. Безбедност ECDSA алгоритма се заснива на елиптичким кривама и чињеници да је веома тешко решити дискретни логаритам на датој кривој.

3.3.4. Интеграције са спољним сервисима

i) Push нотификације уз FCM (Firebase Cloud Messaging)

FCM је мулти-платформско решење које омогућава поуздано слање порука. Углавном се користи за слање нотификација одговарајућем уређају. [26] У овом раду FCM је искоришћен за слање *push* нотификација у одговарајућим ситуацијама, нпр. када клијент

остави рецензију салону, *push* нотификација ће стићи уређају који користи тај салон под условом да је фризер пријављен у апликацији. Коришћена је верзија 33.1.2 библиотеке *com.google.firebase:firebase-bom*.

ii) Google OAuth (Open Authorization) 2.0

Google OAuth 2.0 протокол се користи за ауторизацију корисника при *Google Sign-In* процесу. *Google Sign-In* је предуслов за приступање *Google Calendar* API, што је неопходно како би се резервације аутоматски уносиле у *Google* календаре клијента и салона. У овом раду је за *Google Sign-In* на клијентској страни апликације коришћена верзија 21.4.0 библиотеке *com.google.android.gms:play-services-auth.*

3.3.5. Контејнеризација уз Docker

Docker је отворена платформа за развој, испоруку и покретање апликација. *Docker* омогућава раздвајање програмског кода апликације од инфраструктуре, као и убрзање испоруке софтвера клијентима. Уз коришћење *Docker* технологије могуће је управљати инфраструктуром у којој ће се извршавати апликација на сличан начин као што се пише програмски код саме апликације, кроз дефинисање одговарајућих конфигурационих фајлова. [27]

Docker омогућава да се апликација покрене у изолованом окружењу које се зове *Docker* контејнер. *Docker* контејнер се практично састоји од програмског кода апликације заједно са библиотекама неопходним за извршавање апликације. Захваљујући томе, није важно на којој машини се контејнер извршава – на сваком рачунару ће се извршавати на исти начин. [27] У овом раду коришћена је верзија 28.4.0 *Docker* технологије.

4. Имплементација

У овом поглављу биће детаљно описана имплементација мобилне апликације *BarberBooker*. Обрадиће се редом архитектура система на високом нивоу, дизајн базе података, организација програмског кода пројекта, најважнији процеси у апликацији, као и контејнеризација уз *Docker*. Кроз ове теме биће објашњено функционисање целе *BarberBooker* платформе од највишег нивоа, па све до различитих техничких детаља који су кључни за потпуно разумевање система.

4.1. Архитектура система

У наставку ће бити представљен технички дизајн *BarberBooker* платформе. Прво ће бити објашњена општа архитектура целокупног система, а затим ће посебно бити обрађене појединачне архитектуре како клијентског, тако и серверског дела система.

4.1.1. Општа архитектура целог система

Главне компоненте система су *Android* клијент, *Ktor* сервер и *MySQL* база података. Важно је истаћи да се *Ktor* сервер и *MySQL* сервер базе података покрећу унутар *Docker* контејнера. Апликација интерагује и са спољним сервисима попут *Firebase Cloud Messaging*, *Google Sign-In* API и *Google Calendar* API.

Комуникација између клијента и сервера се одвија преко REST (*REpresentational State Transfer*) API интерфејса користећи HTTP протокол, при чему се подаци размењују у JSON формату. REST представља архитектурални стил у ком се подаци и функционалности третирају као ресурси, при чему им се приступа преко одговарајућег идентификатора ресурса (на енглеском URI (*Uniform Resource Identifier*)). Такође, ресурси се иначе могу дохватити у разноврсним форматима JSON, HTML (*HyperText Markup Language*), XML (*eXtensible Markup Language*), PDF (*Portable Document Format*) и многим другим. [28] Треба нагласити такође да су сви ресурси на серверу заштићени обавезном JWT аутентикацијом која се спроводи при опслуживању сваког клијентског захтева. Уколико клијент нема одговарајући токен приступа, ресурс му неће бити пружен и вратиће му се HTTP грешка са кодом 401 који је управо намењен за овакве ситуације.

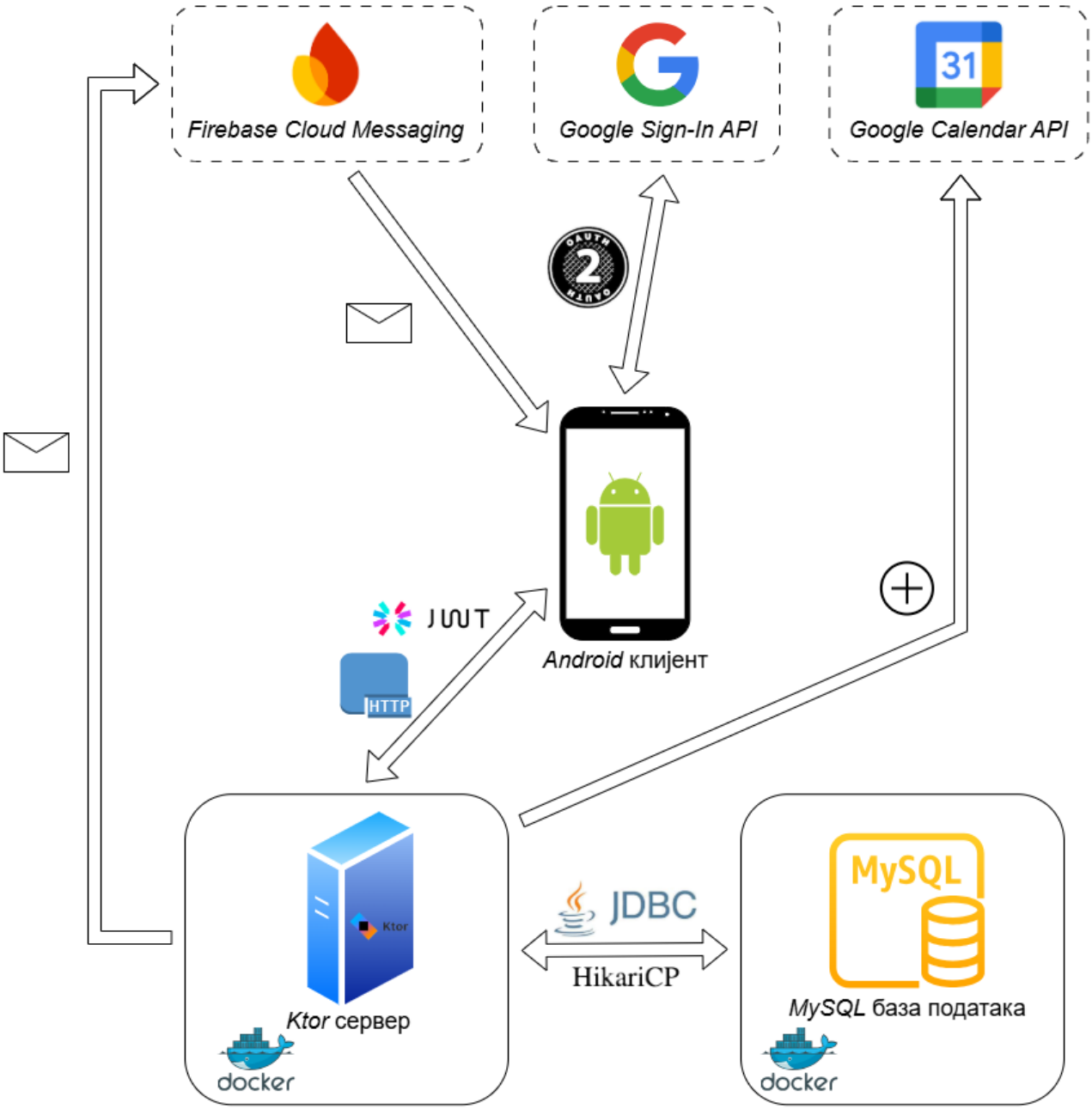
Сервер је повезан са базом података користећи JDBC уз *HikariCP* базен конекција ради бољих перформанси. Комуникација између *Ktor* сервера и сервера базе података је двосмерна, као што је и комуникација између *Android* клијента и *Ktor* сервера двосмерна. Као што *Android* клијент шаље захтев *Ktor* серверу и *Ktor* сервер враћа одговор клијенту, тако и *Ktor* сервер може да захтева податке од сервера базе података које он треба да му врати.

Push нотификације се шаљу захваљујући *Google* сервису који се зове *Firebase Cloud Messaging*. Физички уређаји се идентификују помоћу одговарајућих токена које *Google* издаје за сваки уређај, а ови токени се чувају у бази података за сваки налог у апликацији. Када је потребно послати нотификацију неком уређају, *Ktor* сервер шаље захтев са токеном

дестинационог уређаја овом *Google* сервису, а онда *Firebase Cloud Messaging* сервис на основу њега шаље *push* нотификацију.

Пријава на *Google* налог се обавља уз помоћ *OAuth* 2.0 ауторизационог протокола. Уколико *Android* клијент пошаље адекватне креденцијале, добиће назад ауторизациони код од *Google* сервера са којим после *Ktor* сервер може да тражи од *Google* сервера токене са којима ће моћи да креира догађаје у *Google Calendar* преко *Google Calendar* API.

На слици испод се може видети слободни дијаграм којим је обухваћено функционисање целог система на високом нивоу.



Слика 4.1.1.1 Општа архитектура целог система

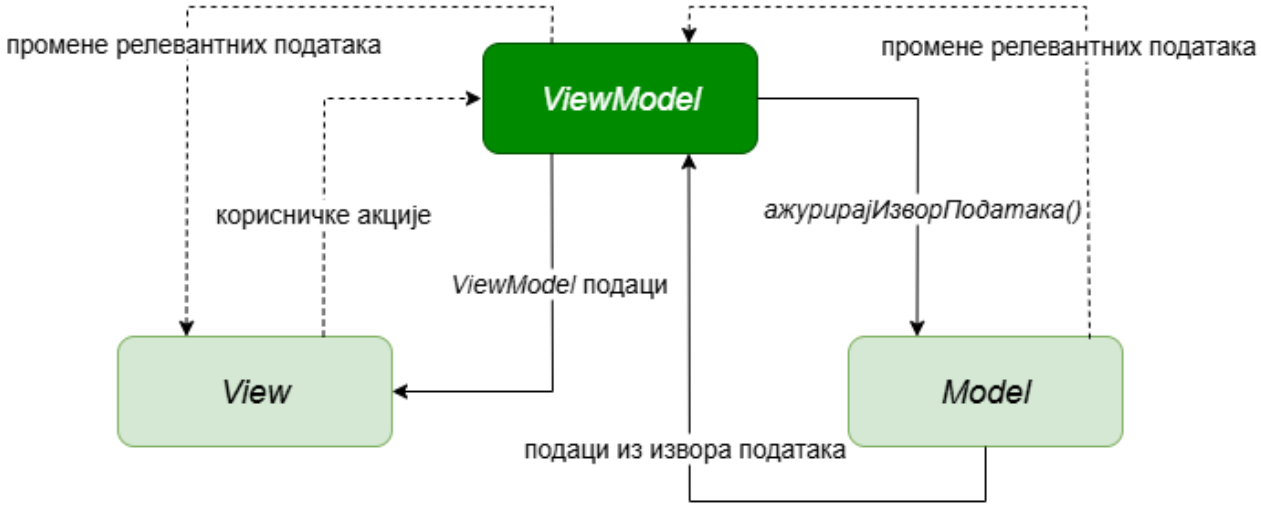
4.1.2. Архитектура Android клијентске апликације

Главни пројектни шаблон који је коришћен при развоју *Android* клијентске апликације јесте MVVM (*Model-View-ViewModel*). Сматра се да је овај шаблон супериорнији у односу на MVP (*Model-View-Presenter*) и MVC (*Model-View-Controller*) пројектне узорке, те је оваква архитектура индустријски стандард при развоју *Android* апликација. [29]

MVVM се састоји из три слоја:

- *Model* – сврха овог слоја јесте апстракција извора података. *ViewModel* комуницира са *Model* слојем ради дохватања података потребних апликацији, као и због ажурирања података у изворима података из којих *Model* дохвата податке.
- *View* – сврха овог слоја јесте да обавести *ViewModel* слој о корисничким акцијама на корисничком интерфејсу, али исто тако и да ослушкује промене у *ViewModel* слоју како би кориснику увек били приказани одговарајући подаци.
- *ViewModel* – сврха овог слоја јесте да достави *View* слоју податке који су му потребни тако што ће проследити податке које добије од *Model* слоја. На овај начин, *ViewModel* представља везу између *View* и *Model* слојева. [29]

На слици испод се може видети шематски приказ MVVM пројектног узорка. [29]



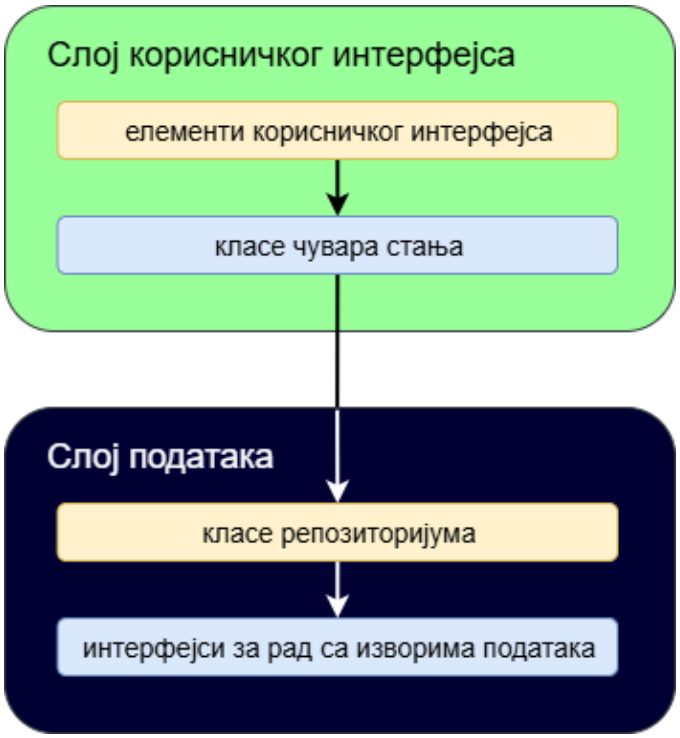
Слика 4.1.2.1 MVVM архитектура

У званичној *Android* документацији је изложена препоручена модерна архитектура заснована на MVVM шаблону, коју би требало пратити при развоју *Android* апликација. Управо ови савети су испоштовани при развоју апликације *BarberBooker* како би написани програмски код био разумљив, лак за тестирање, проширљив и скалабилан. Главна два обавезна слоја у модерној *Android* архитектури су слој корисничког интерфејса и слој података.

Слој корисничког интерфејса се састоји из два дела: елементи корисничког интерфејса и чувари стања. Елементи корисничког интерфејса служе за приказ података кориснику. У овом раду они су имплементирани помоћу *Jetpack Compose* библиотеке. Чувари стања су задужени за чување података потребних за приказ кориснику, као и за имплементацију логике везане за дохватање и ажурирање одговарајућих података у извору података преко комуникације са слојем података. У овом раду чувари стања су одговарајуће *ViewModel* класе.

Слој података се такође састоји из два дела: репозиторијума и класа за рад са изворима података. Репозиторијуми су задужени за достављање података слоју корисничког интерфејса, централизацију промена података у изворима података, апстракцију извора података остатку апликације, као и за имплементацију главне бизнис логике. [30] С обзиром на малу комплексност бизнис логике у апликацији *BarberBooker*, у овом раду је већина логике имплементирана у *ViewModel* класама. У случају даљег развоја ове апликације требало би тежити имплементирању нове бизнис логике у репозиторијумима. Свака класа за рад са извором података треба да буде одговорна за рад са тачно једним извором података. Извори података могу бити различити: фајл, локална база података, удаљена база података и други. [30] У овом раду извор података за *Android* апликацију је удаљена база података из које се дохватају подаци преко мреже користећи *Retrofit*. За дохватање података преко мреже се користе интерфејси кроз које се дефинише REST API *Android* клијентске апликације.

На слици испод се може видети шематски приказ модерне *Android* архитектуре која је коришћена при развоју апликације *BarberBooker*.



Слика 4.1.2.2 Модерна архитектура *Android* апликације

4.1.3. Архитектура Ktor серверске апликације

Суштина *Ktor* сервера у овом систему је да пружи REST API клијентској *Android* апликацији преко ког она може да захтева ресурсе за које је потребна комуникација са базом података или спољним сервисима. REST API сервера је прецизиран користећи уграђену фукнцију из пакета *io.ktor.server.routing*, помоћу које се дефинишу руте преко којих клијент може да тражи ресурсе. Комуникација са базом података се не обавља директно из рута, већ се зову методе статичких класа за приступ подацима (DAO (*Data Access Object*)) у којима се извршавају одговарајуће операције над базом података користећи неку од отворених JDBC конекција са базом. Може се закључити да је Ktor сервер подељен на два слоја, а то су слој рута и слој приступа подацима. Оваква подела чини код модуларним, читљивијим и лакшим за тестирање.

У овом раду, свака статичка DAO класа се користи за интеракцију са тачно једном табелом из базе података. С друге стране, руте су груписане логички у функције које представљају прикључке (на енглеском *plugin*), при чему су сви ти прикључци део једне веће функције, тј. прикључка, чијим позивом се клијенту објављују на коришћење све руте сервера.

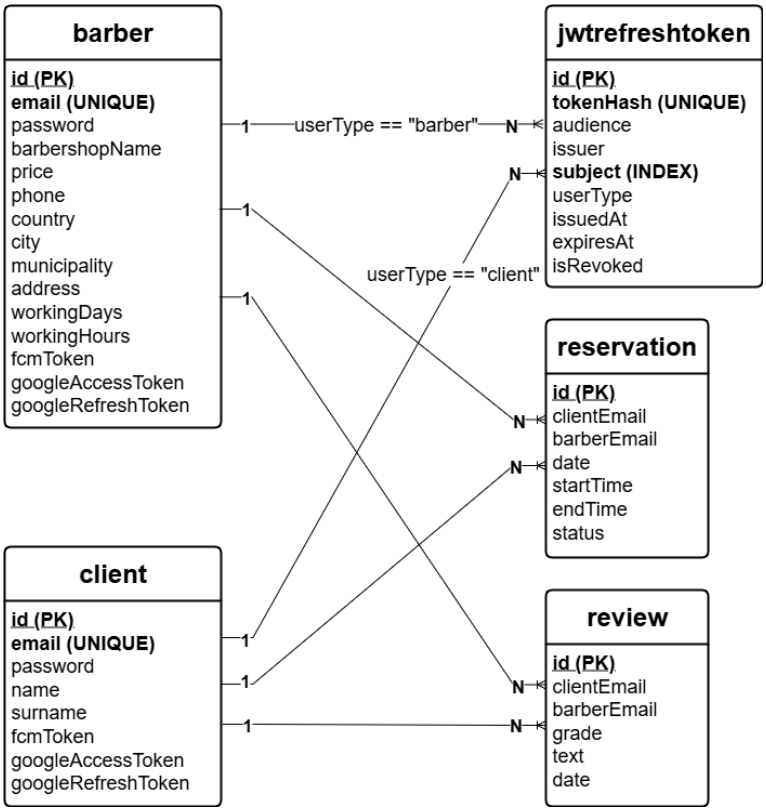
4.2. База података

Дизајн базе података представља једну од кључних пројектних одлука зато што од шеме базе података директно зависи начин на који ће серверска апликација комуницирати с њом. У овом раду искоришћена је *MySQL* релациона база података која се састоји из пет табела:

- *barber* – за чување информација о фризерским салонима.
- *client* – за чување информација о клијентима.
- *jwtrefreshtoken* – за чување свих издатих JWT токена освежења који нису истекли.
- *reservation* – за чување информација о свим резервацијама.
- *review* – за чување информација о свим рецензијама које су клијенти остављали салонима.

Један од најчешћих видова представљања шеме базе података јесте ER (*Entity-Relationship*) дијаграм. На њему се јасно виде колоне у свим табелама, као и међусобне релације између табела. Иако у бази података коришћеној у овој апликацији нема експлицитно дефинисаних страних кључева, логичке везе између табела постоје.

На слици испод се може видети ER дијаграм базе података коришћене у овом раду.



Слика 4.2.1 ER дијаграм базе података коришћене за апликацију *BarberBooker*

4.3. Структура пројекта

У овом поглављу биће показана организација програмског кода мобилне апликације *BarberBooker*. С обзиром да се апликација састоји из клијентског и серверског дела, природно је имплементација урађена кроз два одвојена репозиторијума. Подела репозиторијума у директоријуме логички прати архитектуре изложене у поглављу 4.1. [31] [32]

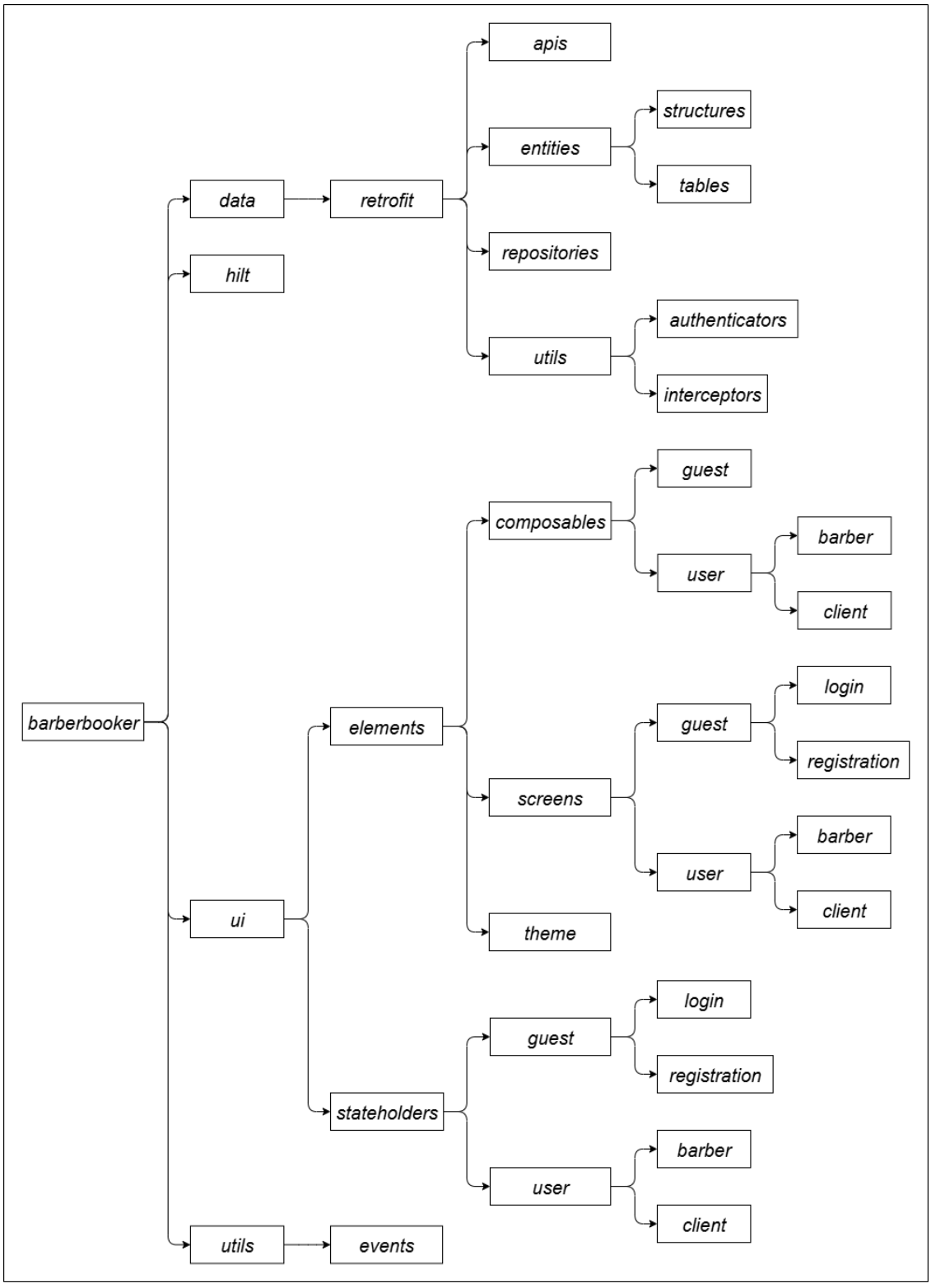
Прво ће бити објашњена структура клијентске апликације. Корени директоријум програмског кода апликације је *BarberBooker/app/src/main/java/rs/ac/bg/etf/barberbooker*. Овај директоријум је даље подељен на директоријуме *data*, *hilt*, *ui* и *utils*. Фолдер *data* енкапсулира код који представља слој података клијентске апликације, док фолдер *ui* садржи код који се односи на слој корисничког интерфејса. У *hilt* директоријуму су дефинисане статичке класе које обезбеђују аутоматско инстанцирање објеката одговарајућих класа, односно *dependency injection*. У фолдеру *utils* се налази подфолдер *events* у којем је дефинисана статичка класа *SessionExpiredEventBus* која је искоришћена за обавештавање слоја корисничког интерфејса да је пријављеном кориснику истекао JWT токен освежења, због чега ће он тада бити аутоматски одјављен из апликације.

Директоријум *data* садржи подфолдер *retrofit* који је подељен у директоријуме *apis*, *repositories*, *entities* и *utils*. У *apis* директоријуму написани су интерфејси помоћу којих се дефинише REST API клијентске апликације. На основу ових интерфејса се кроз *dependency injection* одговарајући објекти аутоматски прослеђују класама репозиторијума. Управо у фолдеру *repositories* су дефнисане све класе репозиторијума преко којих класе чувара стања, односно *ViewModel* класе, траже одговарајуће ресурсе од серверске апликације преко мреже. Фолдер *entities* је даље подељен на фолдере *structures* и *tables*. У фолдеру *structures* се налазе класе које се користе као тела HTTP захтева које клијент шаље серверу, али и као тела HTTP одговора које сервер шаље клијенту. Објекти ових класа се аутоматски серијализују у JSON формат при слању, а такође се и аутоматски врши десеријализација из JSON формата у оригинални објекат при пријему. У фолдеру *tables* su дефинисане класе које се односе на табеле у бази података. Помоћу њих се размењују објекти који садрже све информације као и један ред табеле у бази. Фолдер *utils* је подељен на директоријуме *authenticators* и *interceptors*. У фолдеру *authenticators* се налази класа која обезбеђује да се у случају неуспешне аутентикације корисника на основу његовог приступног токена пошаље захтев серверу за новим приступним токеном користећи његов токен освежења. У фолдеру *interceptors* се налазе класе које обезбеђују пресретање HTTP захтева које клијент шаље серверу. Класа *LoggingInterceptor* у овом фолдеру обезбеђује да се сви одлазећи HTTP захтеви испишу ради бољег праћења. Класа *SessionExpiredInterceptor* обезбеђује да се слој корисничког интерфејса обавести у ситуацији када је кориснику истекао токен освежења јер га тада треба аутоматски одјавити из система. Класа *JwtAuthenticationInterceptor* додаје корисников токен приступа у сваки одлазећи HTTP захтев како би сервер могао да спроведе аутентикацију и врати тражене ресурсе уколико је токен валидан.

Директоријум *ui* је подељен на подфолдере *elements* и *stateholders*. Фолдер *elements* садржи све елементе корисничког интерфејса, док фолдер *stateholders* садржи све класе чувара стања, односно *ViewModel* класе. Даље је фолдер *elements* подељен у директоријуме *composables*, *screens* и *theme*. У фолдеру *composables* се налазе елементи корисничког интерфејса који се користе за приказ делова страница у апликацији. У фолдеру *screens* се налазе елементи корисничког интерфејса који представљају целокупне странице. У

директоријуму *theme* се налазе помоћне класе које се користе за дефинисање боја и стилова текста.

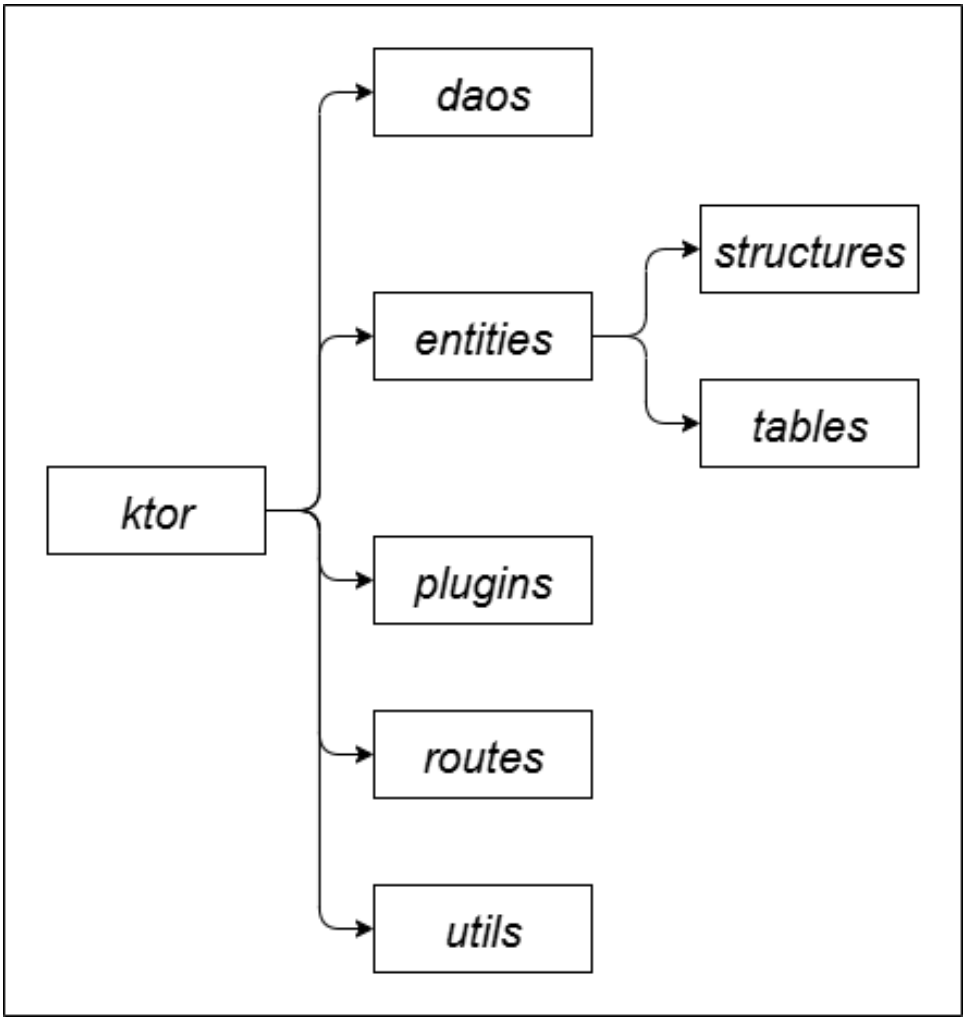
На слици испод се може видети директоријумска организација програмског кода клијентског дела мобилне апликације *BarberBooker*:



Слика 4.3.1 Организација програмског кода клијентског дела мобилне апликације *BarberBooker*

У наставку ће бити објашњена структура серверске апликације. Корени директоријум програмског кода апликације је *BarberBookerServer/src/main/kotlin/rs/etf/snippet/rest/ktor*. Овај директоријум је даље подељен на директоријуме *daos*, *entities*, *plugins*, *routes* и *utils*. Фолдер *daos* представља слој приступа подацима и у њему се налазе статичке класе које се користе за операције над базом података. Фолдер *entities* има идентичну структуру и садржај као и фолдер *entities* у клијентској апликацији. У фолдеру *plugins* се налазе прикључци помоћу којих се при покретању сервера региструју све руте, омогућава JWT аутентикација, као и аутоматска серијализација *Kotlin* објеката у JSON формат, уз аутоматску десеријализацију из JSON облика у *Kotlin* објекте. Фолдер *routes* представља слој рута и у њему су дефинисане све руте преко којих клијент тражи ресурсе од сервера. Коначно, у фолдеру *utils* се налазе различите услужне класе коришћене при дефнисању рута.

На слици испод се може видети директоријумска организација програмског кода серверског дела мобилне апликације *BarberBooker*:



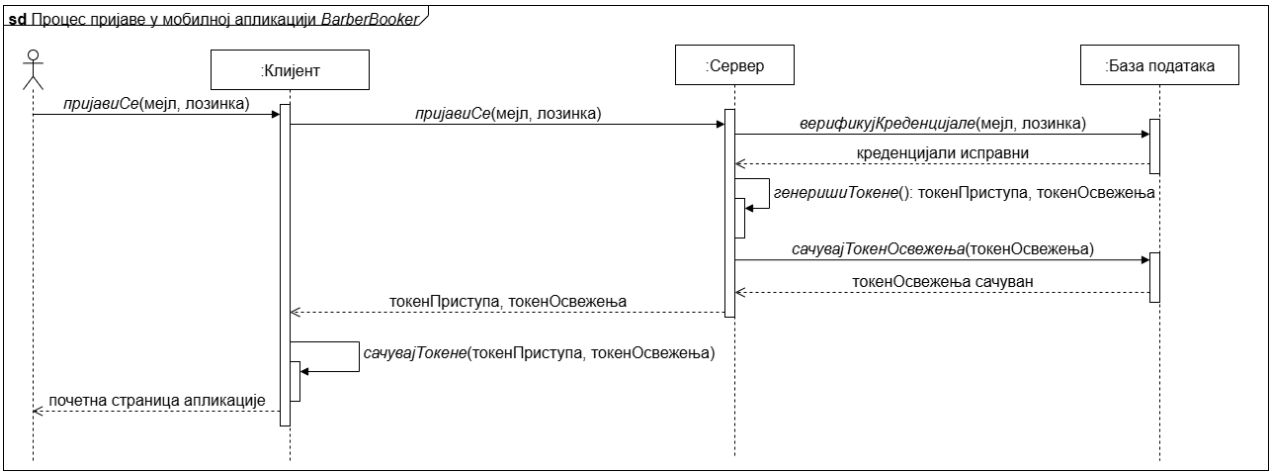
Слика 4.3.2 Организација програмског кода серверског дела мобилне апликације *BarberBooker*

4.4. Најважнији процеси

У овом поглављу биће детаљно обрађена имплементација најважнијих процеса који се одвијају у мобилној апликацији *BarberBooker*. То су аутентикација уз JWT, резервације, *push* нотификације уз FCM, као и интеграција са *Google Calendar*.

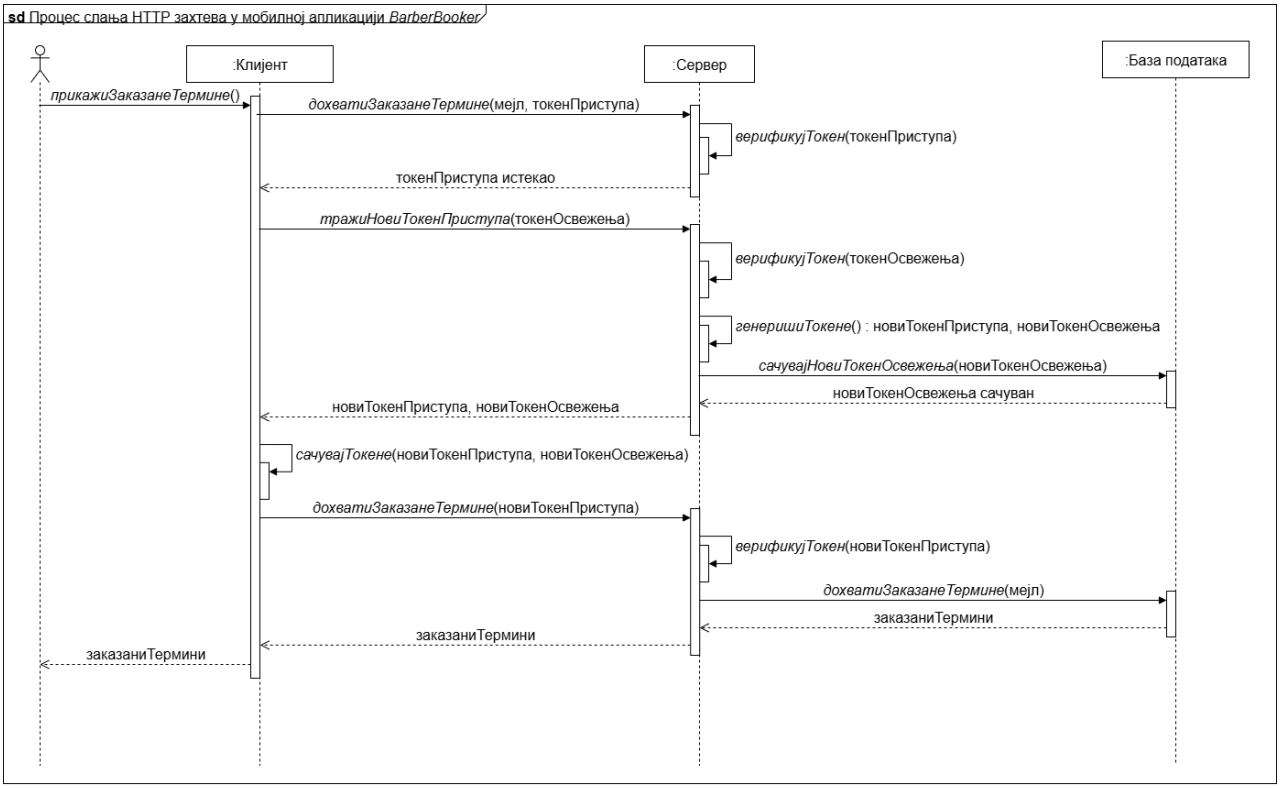
4.4.1. Аутентикација уз JWT (JSON Web Tokens)

У наставку ће бити изложени детаљи имплементације аутентикације у апликацији *BarberBooker*. При свакој пријави у апликацију, корисник с датим налогом добија токене приступа и освежења помоћу којих он онда може да захтева одговарајуће ресурсе од сервера. Испод се на UML дијаграму секвенце може видети процес пријаве корисника на систем. На дијаграму је уведена претпоставка да је корисник унео исправне креденцијале при пријави.



Слика 4.4.1.1 UML дијаграм секвенце за процес пријаве у мобилној апликацији *BarberBooker*

Токени приступа и освежења се чувају у фајлу на мобилном уређају корисника у оригиналном облику. Било би боље да се ови подаци чувају енкриптовани, па би ово било једно од могућих безбедносних побољшања у даљем развоју апликације. Уколико токен приступа истекне, корисник може добити нови токен приступа од сервера помоћу свог токена освежења. У овом случају корисник ће, зарад боље сигурности, добити и нови токен освежења иако му стари није истекао. Ово се догађа аутоматски без икаквог знања корисника, као и без икаквих последица по његово коришћење апликације. Међутим, уколико је и токен освежења истекао, корисник ће бити аутоматски одјављен из система, па ће бити приморан да се поново пријави како би користио апликацију. Након успешне поновне пријаве, он ће добити нови пар валидних токена. Испод ће на UML дијаграму секвенце бити приказан сценарио када корисник жели да види своје заказане термине, при чему му је истекао токен приступа, а има валидан токен освежења.



Слика 4.4.1.2 UML дијаграм секвенце за процес слања HTTP захтева од клијента ка серверу у мобилној апликацији *BarberBooker*

Треба нагласити да се, ради додатне безбедности, у бази података чувају сви токени освежења који нису истекли. Иницијално се сваки нови токен освежења чува, али се једном дневно догађа чишћење базе при ком се бришу сви истекли токени јер они не могу бити искоришћени за напад. Потенцијални злонамерни нападач који шпијунира комуникацију корисника преко мреже може доћи до његовог токена освежења и искористити га за добијање токена приступа. На тај начин он може да користи апликацију као да је пријављен као баш тај корисник. У бази се чувају сви токени освежења који нису истекли управо да би се детектовали овакви напади. То се не ради за токене приступа зато што они трају само 15 минута, па је мањи временски период у којем је напад могућ. Са друге стране, токени освежења трају седам дана, па су зато они најчешћа мета нападача. Валидни токени освежења се разликују од невалидних захваљујући вредности колоне *isRevoked* у табели *jwtrefreshtoken*. Токен освежења који није истекао постаје невалидан када корисник добије нови пар токена приступа и освежења, као и када се корисник одјави из апликације.

4.4.2. Резервације

У овом делу биће укратко објашњена имплементација категоризације резервација. Резервације се разврставају на основу вредности колоне *status* у табели *reservation*. Могуће вредности статуса резервације су:

- PENDING – клијент је послао захтев за резервацијом и чека салон да га потврди или одбије.
- ACCEPTED – салон је потврдио резервацију.
- REJECTED – салон је одбио захтев за резервацијом.

- WAITING_CONFIRMATION – време договореног термина је у прошлости, па се чека да салон потврди да ли је клијент заиста дошао.
- DONE_SUCCESS – салон је потврдио да је клијент заиста дошао на договорени термин.
- DONE_FAILURE – салон је потврдио да клијент није дошао на договорени термин.

4.4.3. Push нотификације уз FCM (Firebase Cloud Messaging)

За имплементацију push нотификација са FCM било је неопходно конфигурисати како клијентску, тако и серверску апликацију. Са *Firebase Cloud Console* платформе скинути су неопходни конфигурациони фајлови јединствени за *BarberBooker* апликацију у JSON формату. На основу података из тих фајлова, FCM платформа зна да је сервер *BarberBooker* апликације ауторизован да иницира слање *push* нотификација, а с друге стране зна и да је клијентска апликација овлашћена да прима те нотификације.

Након пријаве корисника у *BarberBooker* апликацију, клијентска апликација аутоматски тражи FCM токен од *Firebase* сервера. Овај токен јединствено идентификује физички уређај на којем је покренута клијентска апликација. FCM токени се чувају у бази за сваког корисника да би касније *BarberBooker* сервер могао да пошаље токен *Firebase* серверу, на основу чега он тачно зна ком уређају треба да пошаље нотификацију.

Када клијент пошаље захтев за резервацијом салону или остави рецензију, шаље се *push* нотификација салону. Такође, клијентима се шаљу нотификације када им салон прихвати или одбије резервацију.

4.4.4. Интеграција са Google Calendar

Предуслов за приступ *Google Calendar* API јесте успешно обављен поступак *Google Sign-In. Google OAuth* 2.0 протокол се користи за ауторизацију корисника при *Google Sign-In* процесу. Прво је неопходно на платформи *Google Cloud Console* да се за конкретну апликацију генеришу одговарајући креденцијали попут *OAuth Client ID* (Identification) и *OAuth Client Secret*. Ове податке знају само *Google* и апликација. На основу *OAuth Client ID, Google* препознаје апликацију када она затражи од *Google* ауторизационог сервера код који је неопходан за касније захтевање приступних токена и токена освежења који су потребни за приступ *Google Calendar* API. Овиме је процес *Google Sign-In* завршен. [27]

На *Google Cloud Console* платформи је потребно генерисати и одговарајуће креденцијале за сервер апликације (*Client ID*, *Client Secret*) како би он могао да приступа одговарајућим *Google* API. Клијентски део апликације треба да проследи серверу ауторизациони код који је добијен од *Google* како би сервер могао да добије приступни токен и токен освежења од *Google*. Уз овај код, као и уз своје креденцијале, сервер успешно може да добије токене од *Google* са којима коначно може да приступа *Google* API попут *Google Calendar* API и да преко њега креира догађаје у *Google Calendar* од пријављеног корисника. [27]

4.5. Контејнеризација уз *Docker*

У овом раду серверски део апликације се извршава у оквиру два *Docker* контејнера: један представља базу података, а други сам сервер чији су сав програмски код и библиотеке претходно спаковане у један дебели JAR (Java ARchive) фајл који се покреће при покретању контејнера. Ови контејнери се једноставно покрећу помоћу *docker-compose.yaml* фајла који

управо омогућава покретање више *Docker* контејнера одједном, уз могућност додатне конфигурације попут постављања системских променљивих, одређивања порта ослушкивања контејнера и остале.

5. ФУНКЦИОНАЛНОСТИ

У овом поглављу биће детаљно објашњене све функционалности које пружа мобилна апликација *BarberBooker*. Функционалности ће бити обрађене редом за сваки тип корисника у апликацији, а то су гост, клијент и фризерски салон. Осим текстуалних описа, биће приказани и UML (*Unified Modeling Language*) дијаграми случајева коришћења, који ће значајно допринети разумевању свих функционалности.

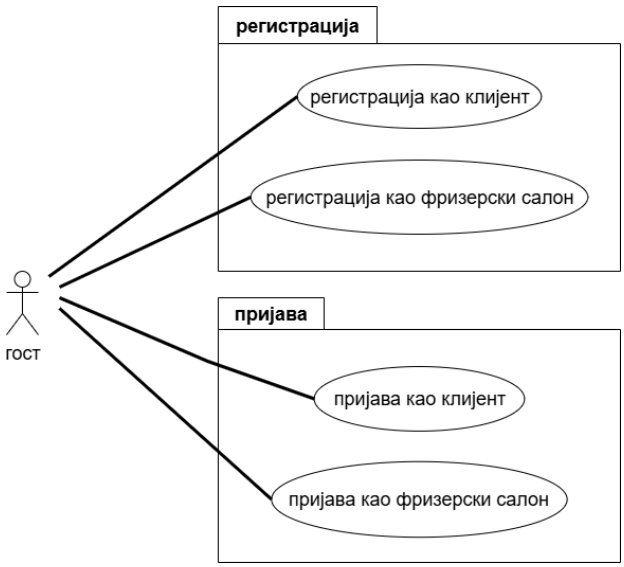
5.1. Функционалности госта

У наставку ће бити објашњене све функционалности које име гост, тј. корисник који није пријављен у апликацији. Након тога, биће приказан UML дијаграм случајева коришћења који ће и визуелно представити све наведене функционалности.

Функционалности које има гост су следеће:

- регистрација као клијент – уносом мејл адресе, лозинке, имена и презимена могуће је направити нови клијентски налог.
- регистрација као фризерски салон – уносом мејл адресе, лозинке, назива салона, радног времена, цене услуге, државе, града, општине, назива улице, броја адресе, као и броја мобилног телефона могуће је направити нови фризерски налог.
- пријава као клијент – уносом мејл адресе и лозинке могуће ја пријавити се као клијент.
- пријава као фризерски салон – уносом мејл адресе и лозинке могуће је пријавити се као фризерски салон.

На слици испод се може видети UML дијаграм случајева коришћења који обухвата све функционалности госта.



Слика 5.1.1 UML дијаграм случајева коришћења за госта

5.2. Функционалности клијента

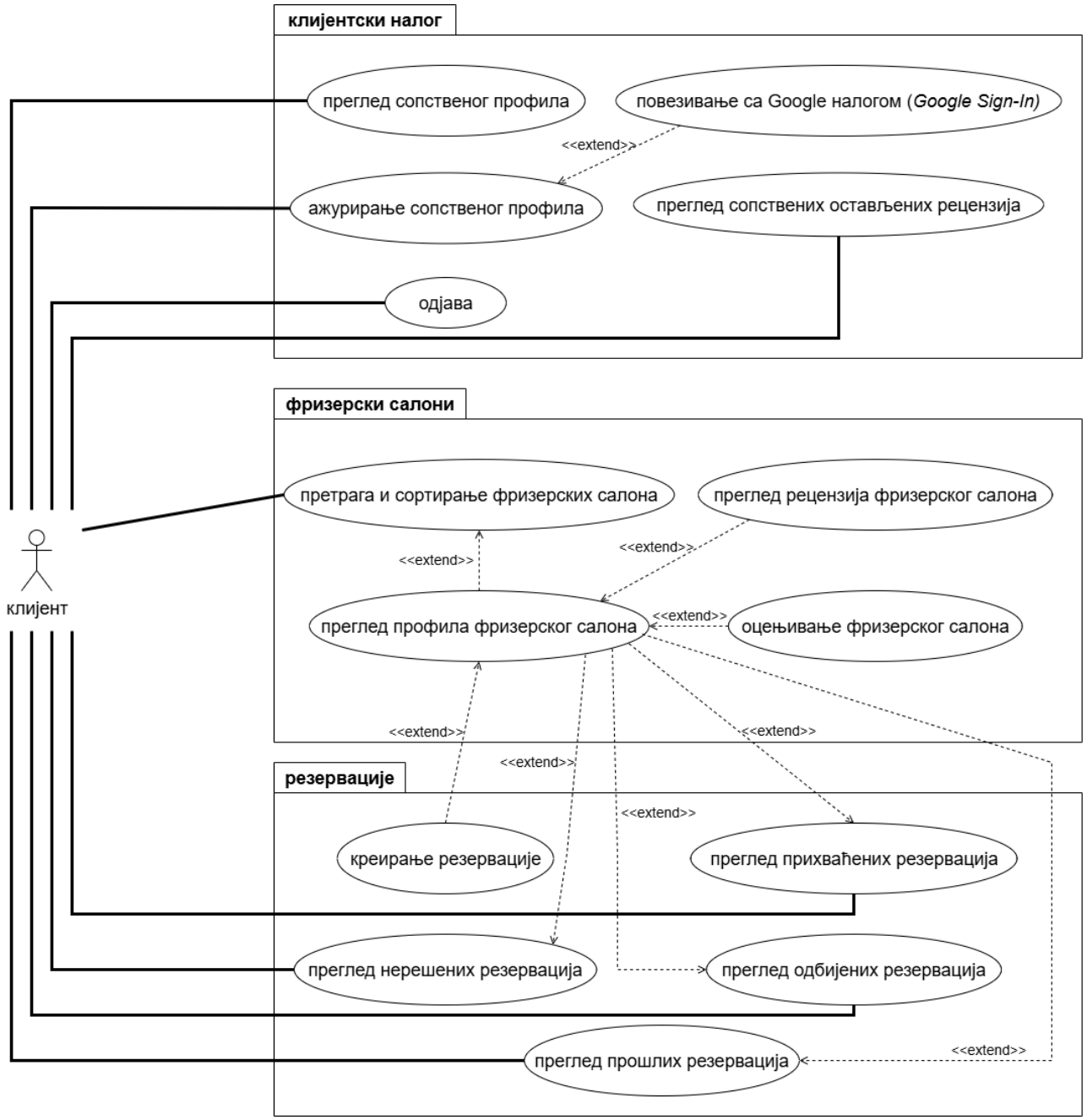
У наставку ће бити објашњене све функционалности које има клијент, односно особа која жели да у апликацији пронађе одговарајући фризерски салон који ће јој пружити услугу. Након тога, биће приказан UML дијаграм случајева коришћења који ће и визуелно представити све наведене функционалности.

Функционалности које има клијент су следеће:

- преглед сопственог профила – могуће је прегледати податке унете при регистрацији.
- ажурирање сопственог профила – могуће је изменити податке унете при регистрацији.
- повезивање налога из апликације са *Google* налогом (*Google Sign-In*) – могуће је повезати налог из апликације са својим *Google* налогом. Уколико клијент ово уради, за сваку потврђену резервацију ће се аутоматски креирати догађај у његовом *Google Calendar*.
- преглед сопствених остављених рецензија – могуће је прегледати све резенције које је клијент остављао салонима.
- одјава – могуће је одјавити се из апликације.
- претрага и сортирање фризерских салона – могуће је претражити салоне по називу, граду и општини. Након прегледа излистаних резултата претраге, могуће је сортирати их по просечној оцени и по цени услуге.
- преглед профила фризерског салона – могуће је прегледати профил фризерског салона. До њега се може доћи или кликом на неки од резултата претраге, или кликом на неку од резервација за тај салон.
- преглед рецензија фризерског салона – могуће је прегледати све рецензије које постоје за дати салон. До њих се може доћи из екрана на ком се прегледа профил салона.
- оцењивање фризерског салона – могуће је оценити салон обавезном оценом од један до пет, уз опциони текстуални коментар. Оцењивање је омогућено само у случају када је фризерски салон потврдио да је клијент заиста био код њих макар једном.
- креирање резервације – могуће је креирати резервацију одабиром датума и времена. Клијенту ће бити омогућено да одабере искључиво валидне датуме и времена на основу радног времена салона, клијентске и фризерске доступности.
- преглед прихваћених резервација – могуће је прегледати све заказане термине у будућности.
- преглед нерешених резервација – могуће је прегледати све резервације које још увек чекају на одговор од фризерског салона.
- преглед одбијених резервација – могуће је прегледати све резервације које су одбијене од стране фризерског салона.

- преглед прошлих резервација – могуће је видети све потврђене резервације из прошлости, уз информацију о томе да ли је клијент заиста отишао у салон у договорено време или не.

На слици испод се може видети UML дијаграм случајева коришћења који обухвата све функционалности клијента.



Слика 5.2.1 UML дијаграм случајева коришћења за клијента

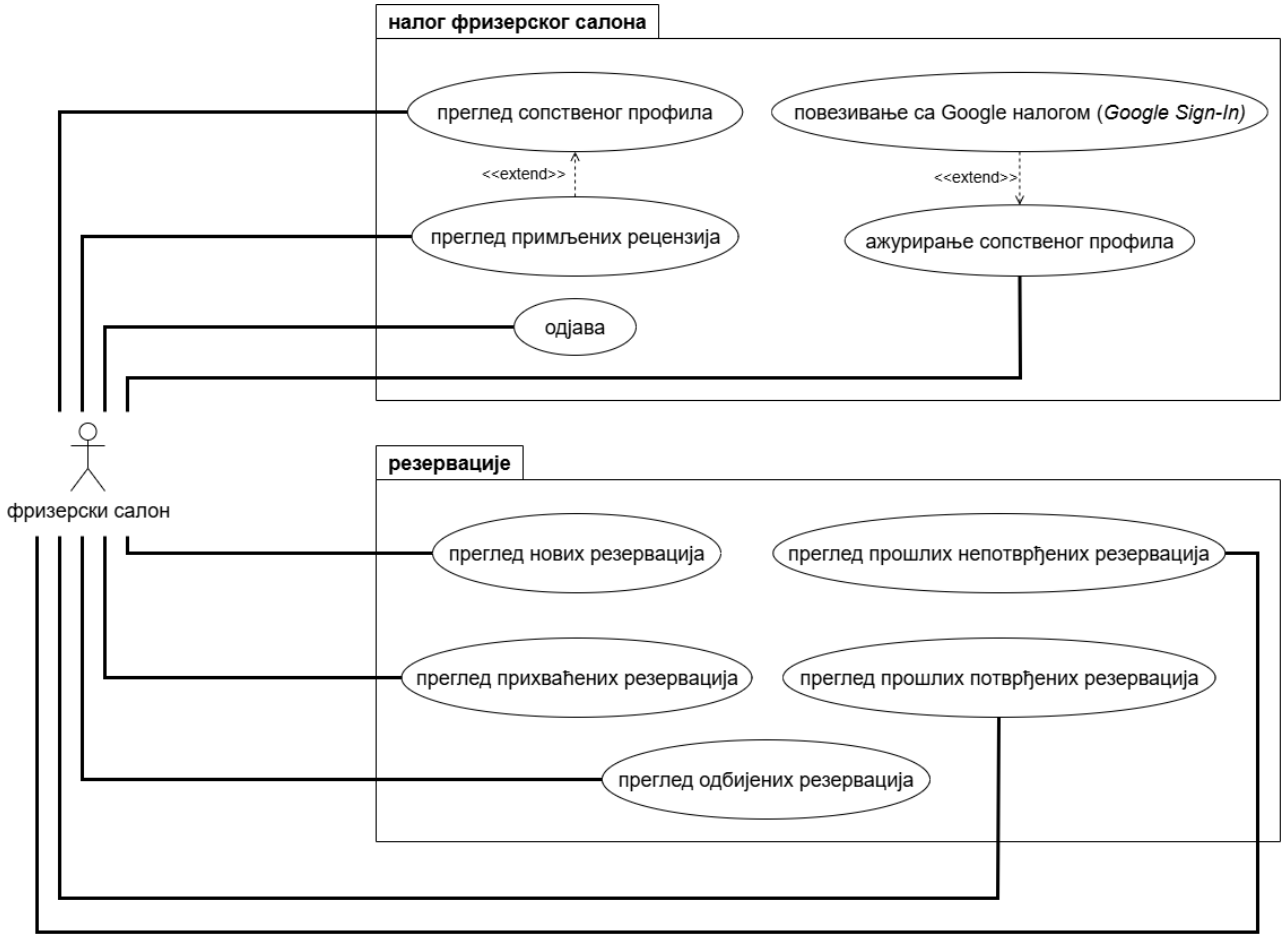
5.3. Функционалности фризерског салона

У наставку ће бити објашњене све функционалности које име фризерски салон. Након тога, биће приказан UML дијаграм случајева коришћења који ће и визуелно представити све наведене функционалности.

Функционалности које има фризерски салон су следеће:

- преглед сопственог профила – могуће је прегледати податке унете при регистрацији.
- преглед примљених рецензија – могуће је прегледати све рецензије које су клијенти оставили.
- ажурирање сопственог профила – могуће је изменити податке унете при регистрацији.
- повезивање налога из апликације са *Google* налогом (*Google Sign-In*) – могуће је повезати налог из апликације са својим *Google* налогом. Уколико фризерски салон ово уради, за сваку потврђену резервацију ће се аутоматски креирати догађај у његовом *Google Calendar*.
- одјава – могуће је одјавити се из апликације.
- преглед нових резервација уз могућност њиховог прихватања или одбијања – могуће је прегледати све нове резервације које салон треба да прихвати или одбије.
- преглед прихваћених резервација – могуће је прегледати све заказане термине у будућности.
- преглед прошлих непотврђених резервација – могуће је прегледати све термине из прошлости за које салон још увек није потврдио да ли је клијент заиста дошао. На овој страници салон управо то треба да потврди за сваку резервацију која се ту излиста.
- преглед прошлих потврђених резервација – могуће је прегледати све термине из прошлости за које је салон потвдио да ли је клијент заиста дошао.
- преглед одбијених резервација – могуће је прегледати све резервације које је салон одбио.

На слици испод се може видети UML дијаграм случајева коришћења који обухвата све функционалности фризерског салона.



Слика 5.3.1 UML дијаграм случајева коришћења за фризерски салон

5.4. Ограничења мобилне апликације *BarberBooker* из угла корисника

У овом поглављу ће бити обрађена ограничења апликације *BarberBooker* која корисници лако могу приметити при коришћењу платформе. С обзиром да овај рад представља развој минималног одрживог производа, очекивано је да апликација има одређене недостатке. Ове мане ће бити систематично објашњене за сваки тип корисника. Додатно, биће дискутовано и о могућим решењима за наведена ограничења.

Главна ограничења госта су:

- немогућност било ког вида коришћења платформе – гост је приморан да се региструје, а затим и пријави у апликацији да би могао да је користи. Ово није нужно лоше зато што многе реалне апликације такође обавезују кориснике да се региструју, а затим и пријаве да би могли да их користе. Ово ограничење би могло да се отклони тако што би се дозволило и госту да претражује салоне. Наравно, за све остале функционалности апликације попут заказивања термина и остављања рецензија, корисници би и даље морали да се прво пријаве.
- немогућност иницијалне пријаве коришћењем нпр. *Google* налога (*Google Sign-In*) – корисници немају опцију да се иницијално пријаве помоћу свог *Google* налога како би на тај начин избегли заморан процес ручног уношења података при регистрацији, а затим и при пријави. Тренутно имају само могућност да повежу свој већ креирани налог у апликацији са својим *Google* налогом. Увођење ове

функционалности би значајно побољшало корисничко искуство зато што би убрзало приступ главним функционалностима апликације како новим, тако и постојећим корисницима.

Главна ограничења клијента су:

- немогућност отказивања или ажурирања резервација – клијенти тренутно не могу никако да измене резервацију у апликацији након што је направе. Отказивања или промене планова су регуларне ситуације које се често догађају, стога би било обавезно да се то и омогући у апликацији.
- немогућност онлајн плаћања – много људи у данашње време највише воли да плаћање обавља картицом због једноставности. Из тог разлога, увођење онлајн плаћања унутар апликације би драстично унапредило корисничко искуство. Ово би такође омогућило додатну заштиту салона од нерегуларних отказивања резервација у виду аутоматског скидања новца са клијентске картице. Ово ограничење би требало отклонити интеграцијом добро познатих видова онлајн плаћања попут *Apple Pay*, *Google Pay*, кредитне картице и слично.

Главна ограничења фризерског салона су:

- недовољна флексибилност при дефинисању радног времена – апликација приморава салоне да сваки свој радни дан имају исто радно време. Ово ограничење би требало отклонити како би салони могли за сваки свој радни дан да дефинишу произвољно радно време, што и јесте реална потреба.
- немогућност дефинисања различитих услуга – тренутно салони уопште не могу да понуде више различитих услуга, већ се у апликацији имплицитно подразумева да је у питању мушко шишање. Ово је озбиљно ограничење које је уведено ради једноставности, али би дефинитивно морало да буде отклоњено како би апликација заиста била употребљива. Као што се може видети у поглављу број два, обе анализиране апликације омогућавају салонима да понуде многобројне услуге.
- немогућност дефинисања трајања услуге – салони не могу да одреде колико им траје један термин, већ је то предодређено да буде 30 минута за све салоне. Ово ограничење би такође требало отклонити како би се задовољиле реалне потребе салона.

Главна ограничења заједничка за клијенте и фризерске салоне су:

- недостатак визуелног приказа распореда унутар апликације – тренутно корисници могу видети све врсте резервација на исти начин, у виду листе попуњене текстом. Њихово корисничко искуство би било значајно боље уколико би се имплементирао јаснији визуелни приказ њихових будућих термина у виду некаквог календара или слично. Са друге стране, пошто је имплементирана аутоматска синхронизација са *Google Calendar*, корисници могу тамо видети леп визуелни приказ њиховог распореда, али за то онда морају да напусте апликацију *BarberBooker*, што негативно утиче на корисничко искуство.
- непостојање странице са излистаним нотификацијама – иако су *push* нотификације обезбеђене, није обрађена ситуација у којој се покуша слање нотификације кориснику који није пријављен у апликацији у датом тренутку. То значи да тај корисник неће видети ту нотификацију следећи пут када се пријави. Ово ограничење би требало отклонити увођењем нове странице на којој би корисник могао да види излистане како нове, тако и старе нотификације.

6. Закључак

Може се рећи да је исход овог рада успешан зато што је из њега проистекла мобилна апликација *BarberBooker* коју клијенти могу да користе за претраживање салона и заказивање термина, уз различите додатне функционалности. Треба нагласити да ова апликација представља минимално одржив производ, што значи да су све најважније основне функционалности имплементиране. Ипак, постоји простор за даља унапређења која су већ наведена у поглављу 5.4. Захваљујући доброј архитектури и модерним технологијама које су коришћене за развој апликације, будуће надоградње ће бити једноставне за имплементацију.

С обзиром да постоји мали број квалитетних мобилних апликација сличне намене, а број различитих врста салона је велики, потенцијал овог решења је озбиљан. Оваква апликација са једне стране оптимизује пословање салона, а са друге стране олакшава клијентима проналазак услуге. Уз адекватне надоградње система, као и одговарајуће пласирање на тржиште, *BarberBooker* платформа може да оствари значајан комерцијални успех.

Литература

[1] Google Play, *СредиМе – Заказивање салона*, https://play.google.com/store/search?q=sredime&c=apps, 17.10.2025.

[2] Google Play, *Fresha for customers*, https://play.google.com/store/search?q=fresha&c=apps, 17.10.2025.

[3] Google Play, *Booksy for Customers*, https://play.google.com/store/search?q=booksy&c=apps, 18.10.2025.

[4] E. Ries, *The Lean Startup*, Crown Business, 2011.

[5] Google Play, *СредиМе ПРО за власнике салона*, https://play.google.com/store/search?q=sredime%20pro&c=apps, 18.10.2025.

[6] *Fresha*, https://www.fresha.com/, 18.10.2025.

[7] Google Play, *Fresha for business*, https://play.google.com/store/search?q=fresha%20for%20business&c=apps, 18.10.2025.

[8] *Meet Android Studio*, https://developer.android.com/studio/intro, 20.10.2025.

[9] *IntelliJ IDEA Overview*, https://www.jetbrains.com/help/idea/discover-intellij-idea.html, 20.10.2025.

[10] *Kotlin Overview*, https://developer.android.com/kotlin/overview, 20.10.2025.

[11] *Android's Kotlin-first approach*, https://developer.android.com/kotlin/first, 20.10.2025.

[12] *Asynchronous Programming Techniques*, https://kotlinlang.org/docs/async-programming.html, 20.10.2025.

[13] *Coroutines*, https://kotlinlang.org/docs/coroutines-overview.html, 20.10.2025.

[14] *Use a Bill of Materials*, https://developer.android.com/develop/ui/compose/bom, 20.10.2025.

[15] *Handling lifecycles with lifecycle-aware components*, https://developer.android.com/topic/libraries/architecture/lifecycle, 20.10.2025.

[16] *ViewModel Overview*, https://developer.android.com/topic/libraries/architecture/viewmodel, 20.10.2025.

[17] *Navigation*, https://developer.android.com/guide/navigation, 20.10.2025.

[18] *Dependency Injection in Android*, https://developer.android.com/training/dependency-injection, 20.10.2025.

[19] *Dependency Injection with Hilt*, https://developer.android.com/training/dependency-injection/hilt-android , 20.10.2025.

[20] *Ktor vs. Spring Boot: 5 Key Differences for Kotlin Devs*, https://digma.ai/ktor-vs-spring-boot-5-key-differences-for-kotlin-devs/, 20.10.2025.

[21] *MySQL: Understanding What It Is and How It's Used*, https://www.oracle.com/mysql/what-is-mysql/, 20.10.2025.

[22] *MySQL Enterprise Edition*, *Workbench*, https://www.mysql.com/products/workbench/, 20.10.2025.

[23] *Java SE Technologies – Database*, https://www.oracle.com/java/technologies/javase/javase-tech-database.html, 20.10.2025.

[24] *Down the Rabbit Hole*, https://github.com/brettwooldridge/HikariCP/wiki/Down-the-Rabbit-Hole, 20.10.2025.

[25] *Introduction to JSON Web Tokens*, https://www.jwt.io/introduction, 20.10.2025.

[26] *Firebase Cloud Messaging*, https://firebase.google.com/docs/cloud-messaging, 20.10.2025.

[27] *What is Docker?*, https://docs.docker.com/get-started/docker-overview/, 20.10.2025.

[28] *REST API Tutorial*, https://restfulapi.net/, 22.10.2025.

[29] *MVVM (Model View ViewModel) Architecture Pattern in Android*, https://www.geeksforgeeks.org/android/mvvm-model-view-viewmodel-architecture-pattern-in-android/, 22.10.2025.

[30] *Guide to app architecture*, https://developer.android.com/topic/architecture, 22.10.2025.

[31] *BarberBooker*, https://github.com/PavleSarenac/BarberBooker, 23.10.2025.

[32] *BarberBookerServer*, https://github.com/PavleSarenac/BarberBookerServer, 23.10.2025.

[33] Using OAuth 2.0 to Access Google APIs, https://developers.google.com/identity/protocols/oauth2, 20.10.2025.

Списак скраћеница

MVP	*Minimum Viable Product*
3G	*Third Generation*
UX	*User Experience*
UI	*User Interface*
IDE	*Integrated Development Environment*
NDK	*Native Development Kit*
JVM	*Java Virtual Machine*
JS	*JavaScript*
BOM	*Bill of Materials*
HTTP	*HyperText Transfer Protocol*
API	*Application Programming Interface*
RDBMS	*Relational Database Management System*
GUI	*Graphical User Interface*
SQL	*Structured Query Language*
JDBC	*Java Database Connectivity*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
RFC	*Request for Comments*
HMAC	*Hash-based Message Authentication Code*
SHA256	*Secure Hash Algorithm 256-bit*
RSA	*Rivest-Shamir-Adleman*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
FCM	*Firebase Cloud Messaging*
OAuth	*Open Authorization*
ID	*Identification*
JAR	*Java ARchive*
REST	*REpresentational State Transfer*
URI	*Uniform Resource Identifier*
HTML	*HyperText Markup Language*
XML	*eXtensible Markup Language*
PDF	*Portable Document Format*
MVVM	*Model-View-ViewModel*
MVP	*Model-View-Presenter*
MVC	*Model-View-Controller*
DAO	*Data Access Object*
ER	*Entity-Relationship*
UML	*Unified Modeling Language*

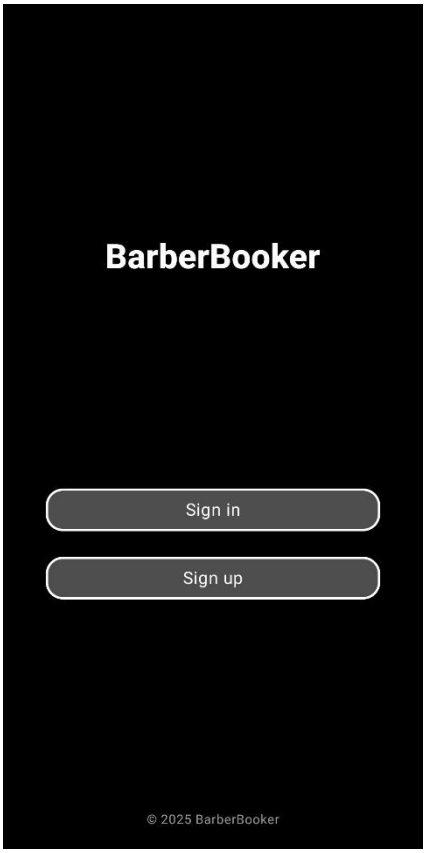
СПИСАК СЛИКА

СПИСАК ТАБЕЛА

A. Кориснички интерфејс мобилне апликације *BarberBooker* из угла госта

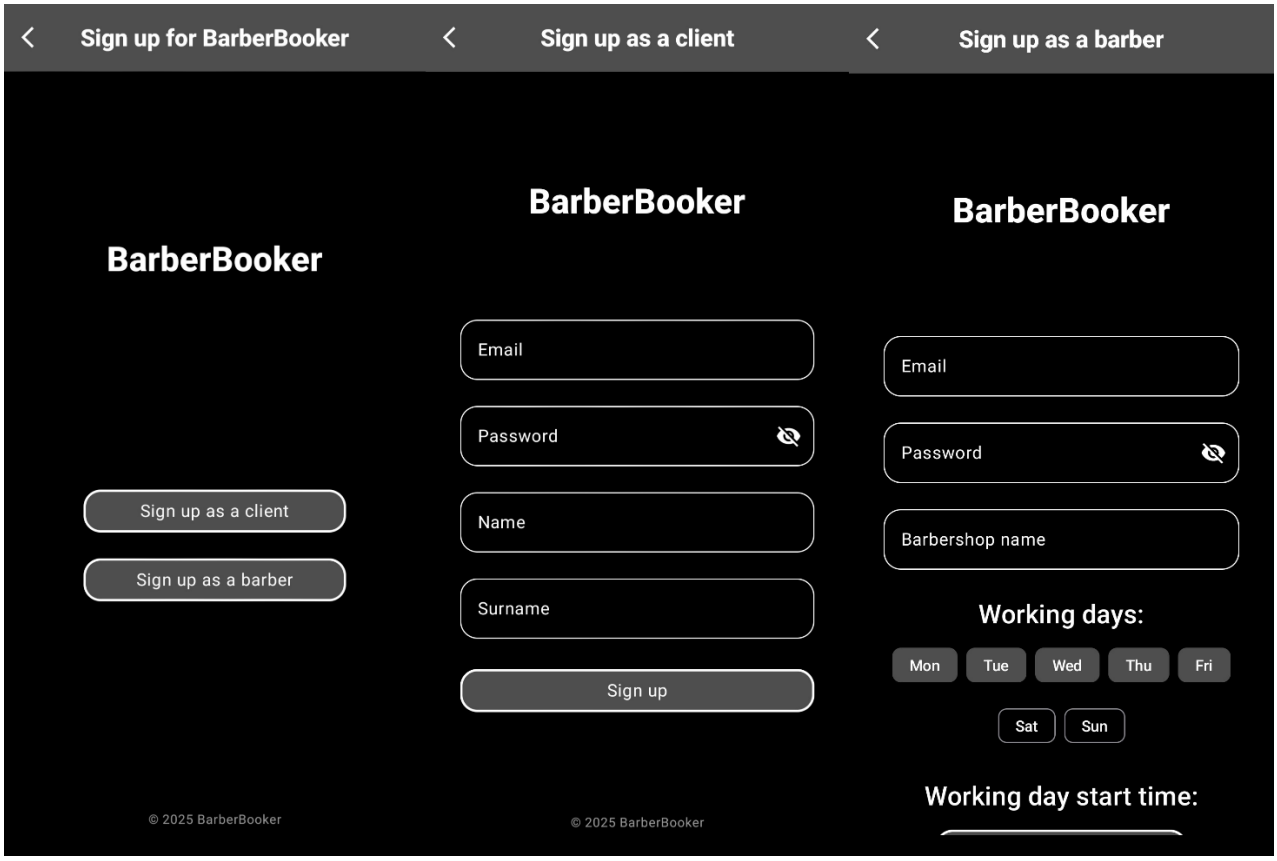
A.1. Почетна страница

На слици испод се може видети изглед почетне странице апликације.



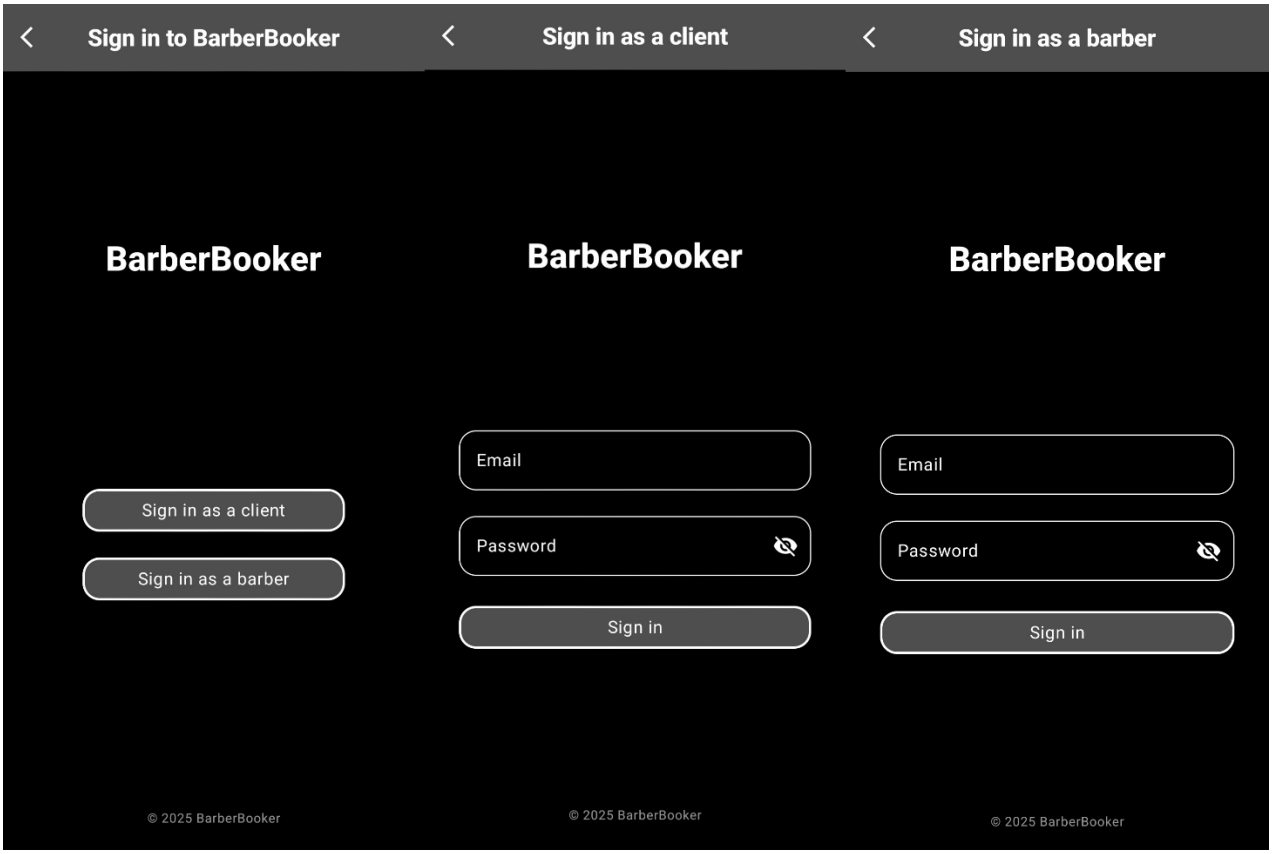
Слика A.1.1 Почетна страница апликације *BarberBooker*

A.2. Регистрација



Слика А.2.1 Странице везане за регистрацију у апликацији *BarberBooker*

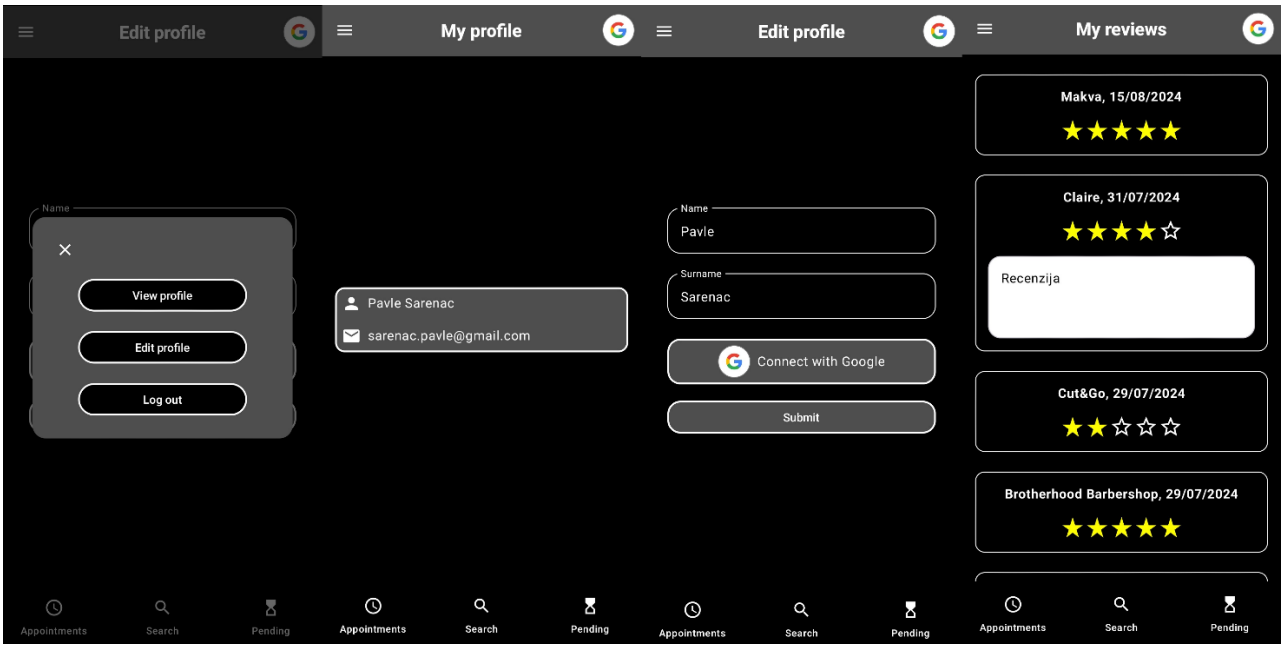
А.3. Пријава



Слика А.3.1 Странице везане за пријаву у апликацији *BarberBooker*

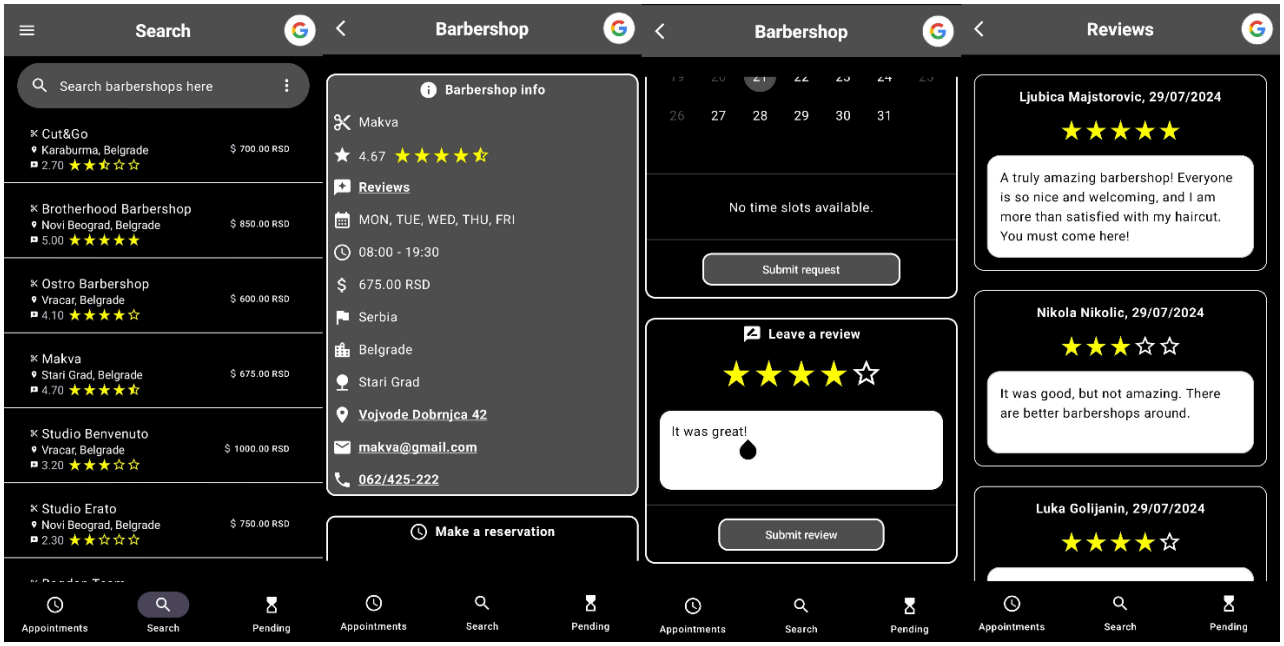
B. Кориснички интерфејс мобилне апликације *BarberBooker* из угла клијента

B.1. Клијентски налог



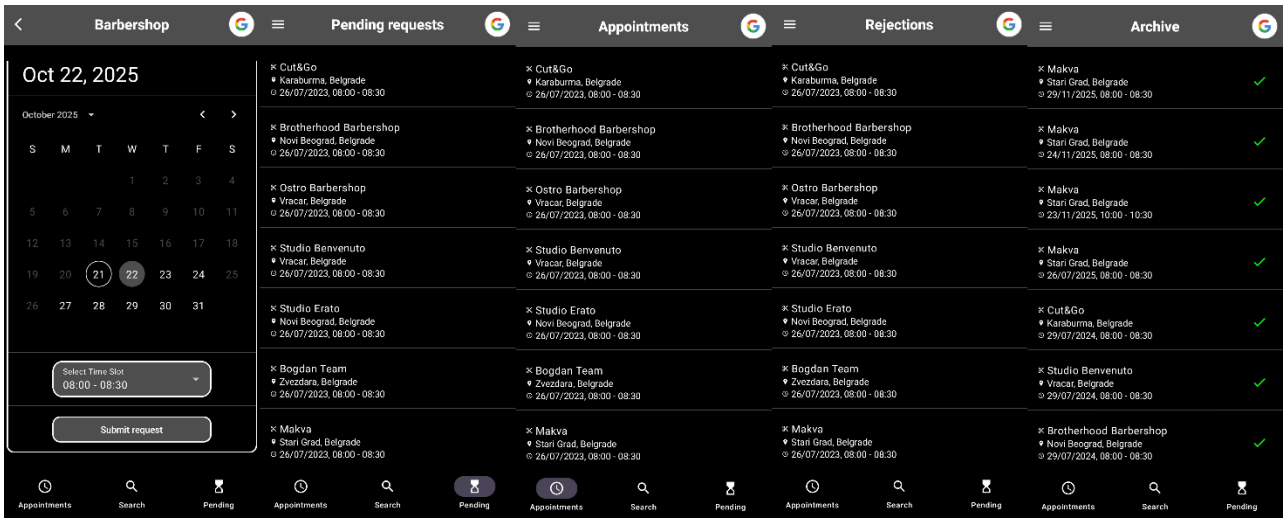
Слика B.1.1 Странице везане за ситуације када клијент прегледа свој налог у апликацији *BarberBooker*

B.2. Фризерски салони



Слика B.2.1 Странице везане за ситуације када клијент претражује салоне у апликацији *BarberBooker*

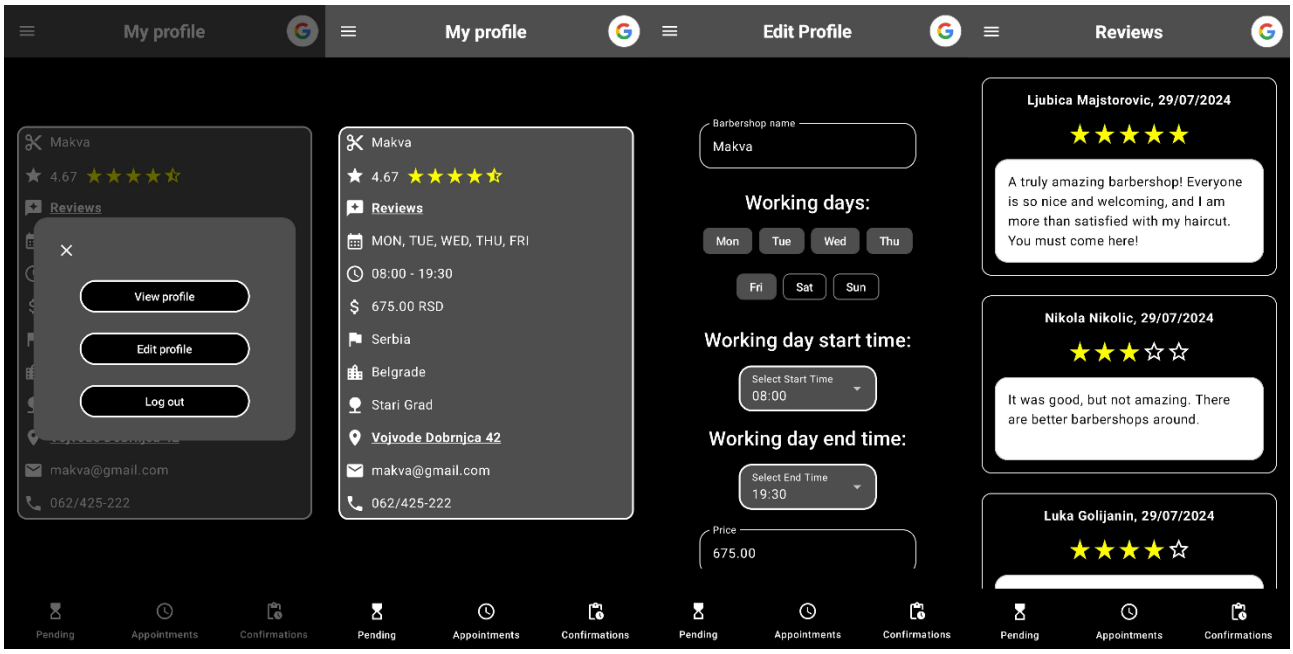
B.3. Резервације



Слика B.3.1 Странице везане за клијентске резервације у апликацији *BarberBooker*

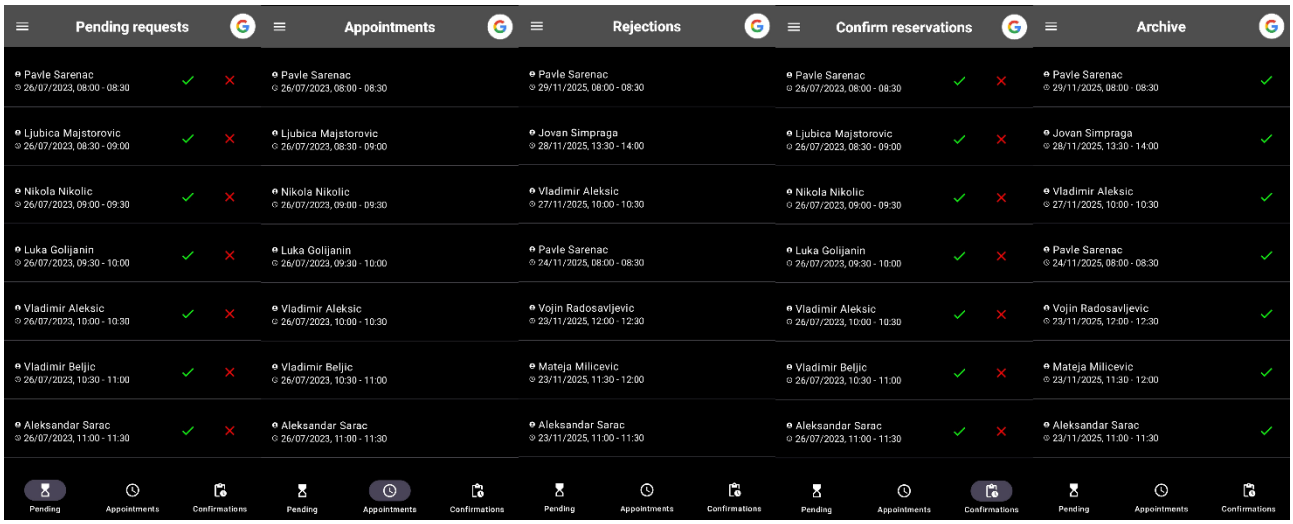
C. Кориснички интерфејс мобилне апликације *BarberBooker* из угла фризерског салона

C.1. Налог фризерског салона



Слика C.1.1 Странице везане за ситуације када фризерски салон прегледа свој налог у апликацији *BarberBooker*

C.2. Резервације



Слика C.2.1 Странице везане за резервације фризерског салона у апликацији *BarberBooker*