|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТУНОВОМСАДУ  **ФАКУЛТЕТТЕХНИЧКИХНАУКАУНОВОМСАДУ** |  |

Имеипрезимекандидата

**Насловдипломскограда**

Дипломскирад

- Основнеакадемскестудије -

НовиСад, 2022.

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТУНОВОМСАДУ  **ФАКУЛТЕТТЕХНИЧКИХНАУКА**  21000 НОВИСАД, ТргДоситејаОбрадовића 6 | Датум: |
|  |
| **ЗАДАТАКЗАИЗРАДУДИПЛОМСКОГ(BACHELOR)РАДА** | Лист: |
| 1/1 |

*(Податкеуносипредметнинаставник - ментор)*

| Врстастудија: | **Основнеакадемскестудије** |
| --- | --- |
| Студијскипрограм: | **Софтверскоинжењерствоиинформационетехнологије**  **илиРачунарствоиаутоматика** |
| Руководилацстудијскогпрограма: | **проф. дрМирославЗарић (SW) илипроф. дрМиланРапаић(РА)** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Имеипрезиме** | Бројиндекса: | **SW xx/20xx** |
| Област: | **Електротехничко и рачунарско инжењерство** | | |
| Ментор: | **ДрИмеипрезиме, звање** | | |
| НАОСНОВУПОДНЕТЕПРИЈАВЕ, ПРИЛОЖЕНЕДОКУМЕНТАЦИЈЕИОДРЕДБИСТАТУТАФАКУЛТЕТАИЗДАЈЕСЕЗАДАТАКЗАДИПЛОМСКИРАД, САСЛЕДЕЋИМЕЛЕМЕНТИМА:   * проблем – темарада; * начинрешавањапроблемаиначинпрактичнепроверерезултатарада, акојетаквапроверанеопходна; * литература | | | |

**НАСЛОВДИПЛОМСКОГ(BACHELOR) РАДА:**

|  |
| --- |
| **Насловдипломскограда** |

**ТЕКСТЗАДАТКА:**

|  |
| --- |
| 1. Анализирати стање у области.  2. Израдити спецификацију захтева софтверског решења.  3. Израдити спецификацију дизајна софтверског решења.  4. Имплементирати софтверско решење према израђеној спецификацији.  5. Тестирати имплементирано софтверско решење.  6. Документовати (1), (2), (3), (4) и (5). |

|  |  |
| --- | --- |
| Руководилацстудијскогпрограма: | Менторрада: |
|  |  |

|  |
| --- |
| Примеракза:- Студента;  - Ментора |

# КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

|  |  |
| --- | --- |
| Редни број, **РБР**: |  |
| Идентификациони број, **ИБР**: |  |
| Тип документације, **ТД**: | монографска публикација |
| Тип записа, **ТЗ**: | текстуални штампани документ |
| Врста рада, **ВР**: | [дипломски или мастер] рад |
| Аутор, **АУ**: | Име и презиме кандидата |
| Ментор, **МН**: | др [име ментора], [звање ментора – доцент, ванредни професор или редовни професор] |
| Наслов рада, **НР**: | Наслов рада |
| Језик публикације, **ЈП**: | српски |
| Језик извода, **ЈИ**: | српски / енглески |
| Земља публиковања, **ЗП**: | Србија |
| Уже географско подручје, **УГП**: | Војводина |
| Година, **ГО**: | 2022 |
| Издавач, **ИЗ**: | ауторски репринт |
| Место и адреса, **МА**: | Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6 |
| Физички опис рада, **ФО**: | бр. поглавља / страница / цитата / табела / слика / графикона / прилога |
| Научна област, **НО**: | Софтверско инжењерство и информационе технологије |
| Научна дисциплина, **НД**: | Софтверско инжењерство |
| Предметна одредница /  кључне речи, **ПО**: | 3-5 кључних речи које бисте користили у претраживачу да нађете рад са овом темом |
| **УДК** |  |
| Чува се, **ЧУ**: | Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад |
| Важна напомена, **ВН**: |  |
| Извод, **ИЗ**: | апстракт – један пасус који добро описује суштину рада – проблем, м |
| Датум прихватања теме, **ДП**: |  |
| Датум одбране, **ДО**: |  |
| Чланови комисије, **КО**: |  |
| председник | др Име Презиме, звање |
| члан | др Име Презиме, звање |
| ментор | др Име Презиме, звање |
| Потпис ментора | |

# KEY WORDS DOCUMENTATION

|  |  |
| --- | --- |
| Accession number, **ANO**: |  |
| Identification number, **INO**: |  |
| Document type, **DT**: | monographic publication |
| Type of record, **TR**: | textual material |
| Contents code, **CC**: | [bachelor or master ] thesis |
| Author, **AU**: | Ime i prezime kandidata |
| Mentor, **MN**: | Ime i prezime mentora, [zvanje - assistant professor, associate professor ili full professor], PhD |
| Title, **TI**: | Naslov rada na engleskom |
| Language of text, **LT**: | Serbian |
| Language of abstract, **LA**: | Serbian / English |
| Country of publication, **CP**: | Serbia |
| Locality of publication, **LP**: | Vojvodina |
| Publication year, **PY**: | 2021 |
| Publisher, **PB**: | author’s reprint |
| Publication place, **PP**: | Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6 |
| Physical description, **PD**: | br. poglavlja / stranica / citata / tabela / slika / grafikona / priloga |
| Scientific field, **SF**: | Software Engineering and Information Technologies |
| Scientific discipline, **SD**: | Software Engineering |
| Subject / Keywords, **S/KW**: | Ključne reči na engleskom |
| **UDC** |  |
| Holding data, **HD**: | Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad |
| Note, **N**: |  |
| Abstract, **AB**: | Prevod apstrakta na engleski |
| Accepted by sci. Board on, **ASB**: |  |
| Defended on, **DE**: |  |
| Defense board, **DB**: |  |
| president | Ime i prezime, zvanje na eng., PhD |
| member | Ime i prezime, zvanje na eng., PhD |
| mentor | Ime i prezime, zvanje na eng., PhD |
| Mentor's signature | |

**Садржај**

[КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА 5](#_Toc98832020)

[KEY WORDS DOCUMENTATION 6](#_Toc98832021)

[1. UVOD 9](#_Toc98832022)

[2. PREGLED SLIČNIH SISTEMA 11](#_Toc98832023)

[3. KORIŠĆENE SOFTVERSKE TEHNOLOGIJE 13](#_Toc98832024)

[3.1 Spring 13](#_Toc98832025)

[3.2 Opis tehnologije na koji se vaš rad oslanja 13](#_Toc98832026)

[4. SPECIFIKACIJA 15](#_Toc98832027)

[4.1 Specifikacija zahteva 15](#_Toc98832028)

[4.1.1 Funkcionalni zahtevi 15](#_Toc98832029)

[4.1.2 Nefunkcionalni zahtevi 15](#_Toc98832030)

[4.2 Specifikacija sistema 15](#_Toc98832031)

[4.2.1 Model podataka 16](#_Toc98832032)

[4.2.2 Arhitektura sistema 16](#_Toc98832033)

[5. IMPLEMENTACIJA 17](#_Toc98832034)

[6. DEMONSTRACIJA 19](#_Toc98832035)

[7. ZAKLJUČAK 21](#_Toc98832036)

[8. LITERATURA 23](#_Toc98832037)

[9. BIOGRAFIJA 25](#_Toc98832038)

# Увод

Да би софтверски инжењери били ефикаснији у креирању производа, потребно је да се што више фокусирају на сам развој софтвера и избегну превелику потрошњу времена и ресурса на изградњу и одржавање инфраструктуре. Желимо да бринемо о томе како функционише код у оквиру наше апликације, а не о томе да ли треба да зашрафимо нову плочицу радне меморије унутар сервера.

Ту нам помаже рачунарство у облаку, које омогућава доступност рачунарских ресурса на основу корисничких захтева (енгл. on-demand availability). Компаније које пружају услуге рачунарства у облаку називају се клауд провајдери (енгл. Cloud providers). Постоје три основна типа рачунарства у облаку:

* Софтвер као услуга (енгл. Software as a Service - SaaS)
* Платформа као услуга (енгл. Platform as a Service - PaaS)
* Инфраструктура као услуга (енгл. Infrastructure as a Service - IaaS)

Иницијално, инфраструктура као услуга је била реализована тако што бисмо путем мреже подешавали наш удаљени сервер. Ово јесте много боље од тога да зашрафљујемо сервер у нашем подруму, али ипак морамо донекле да водимо рачуна о том удаљеном серверу (нпр. да подешавамо оперативни систем на њему). Временом се појавила идеја о серверлес архитектури, која подразумева да бринемо искључиво о коду који ће се извршавати у одабраном окружењу, док ће све остало бити конфигурисано од стране провајдера.

Две ствари које су готово неизбежне у оквиру сваког информационог система, па и оног изграђеног кроз серверлес архитектуру, јесу аутентификација и ауторизација. Аутентификацијом утврђујемо који ентитет приступа систему, док ауторизацијом утврђујемо којим деловима система ентитет може да приступи.

У оквиру овог рада посматраћемо имплементацију аутентификације и ауторизације корисника унутар апликације Адверто. Адверто представља апликацију за постављање и читање огласа, коју ћемо изградити у серверлес архитектури, коришћењем сервиса које нам нуди један од водећих клауд провајдера данашњице. У питању је компанија Амазон, односно њен огранак који се зове AWS (Amazon Web Services). AWS је клауд провајдер који нуди преко 200 различитих сервиса, као што су EC2, S3, RDS, API Gateway, Lambda, Cognito, итд.

# 

# Коришћене софтверске технологије

У овом поглављу позабавићемо се технологијама које су коришћене за израду Адверто апликације. Прво ћемо се позабавити алатима за подизање инфраструктуре кроз код, а потом ћемо описати сервисе које смо користили у оквиру инфраструктуре, као и библиотеке помоћу којих смо вршили интеракцију са сервисима у оквиру апликације.

## Алати за инфраструктуру кроз код

Инфраструктура коју посматрамо састоји се из више различитих сервиса на AWS-у. Могуће је креирати сервисе на 3 начина:

* Коришћењем AWS-ове веб апликације познате као конзола (AWS Console)
* Коришћењем интерфејса командне линије (AWS CLI – AWS Command Line Interface)
* Коришћењем неког од алата за подизање инфраструктуре кроз код (Infrastructure as Code - IaC)

Први начин је погодан за упознавање са сервисима, али у продукцијском окружењу није препоручљив јер морамо некако памтити шта смо све и којим редом искликтали у конзоли, док је други начин превише напоран за инжењере. Најбољи од претходно наведених начина јесте трећи начин, који нам омогућава да уредно кроз код бележимо све кораке које смо направили да би креирали и подесили одређени сервис. Основни алат, а уједно и сервис за подизање инфраструктуре кроз код, у виду JSON или YAML синтаксе, јесте Cloud Formation.



Слика 2.1 – Cloud Formation

Временом се испоставило да та синтакса није претерано погодна за програмере, па је AWS креирао AWS CDK (Cloud Development Kit) који представља радни оквир за подизање инфраструктуре кроз код. Он подржава више програмских језика међу којима су Јаваскрипт, Тајпскрипт, Пајтон, Јава, Си шарп и Гоу, а најчешће се користе Тајпскрипт и Пајтон. Код написан у оквиру AWS CDK се преводи у Cloud Formation и тако се испоручује на AWS.

Постоје различити радни оквири који се даље надограђују на Клауд Формејшн или AWS CDK, а један од њих јесте и SST (Serverless Stack Toolkit). SST додатно олакшава коришћење CDK тако што додаје могућност развоја ламбда функција уживо (енгл. Live Lambda Development). Ова функционалност нам омогућава да током самог развоја уживо тестирамо код у окружењу једнаком оном које ћемо имати у продукцијској верзији.



Слика 2.1 – AWS CDK и SST

## Коришћени сервиси

Сада ћемо погледати који су то све сервиси који се појављују у оквиру апликације Адверто и како они функционишу у теорији.

### База података (DynamoDB)

База података коју ћемо користити јесте DynamoDB. Dynamo је нерелациона база креирана од стране људи из АВС-а која подржава кључ-вредност и документ базиране структуре. Специјално је оптимизована за претрагу, при чему је могуће вршити претрагу по кључу или по глобалном индексу (Query), као и претрагау скенирањем читаве базе (Scan). У оквиру претраге могуће је дефинисати и филтере. Комплексно скенирање (са филтерима) се не препоручује јер је прилично скупо. Примарни кључ се може састојати од једног или два дела. Први (обавезни) део јесте партициони кључ (Partition Key/Hash Key), а други (опциони) је сортни кључ (Sort Key/Range Key). Такође, глобални индекси (Global Indexes) се налик кључу могу састојати из једног или два дела.



Слика 2.1 – DynamoDB

### Бекенд (API Gateway, Lambda)

Бекенд наше апликације реализован је помоћу AWS Lambda функција које су увезане у оквиру AWS API Gateway-a.

AWS Lambda представља сервис који нуди рачунарске ресурсе специфичне за извршавање функције којој је тај сервис намењен. Када користимо Lambda сервис, не инстанцирамо никакав сервер и не размишљамо о администрацији, већ се бринемо само за код који се извршава у оквиру њега, а све остало одрађује AWS. Lambda се аутоматски скалира, а наплаћује се на основу времена извршавања функције. Док се Lambda не извршава, она нас не кошта ништа. Први пут када се Lambda позове, одговор је нешто спорији због хладног подизања (cold start) који подразумева буђење наше Lambde у оквиру АВСа, јер се Lambda која се не користи одређено време аутоматски успава, па ново резервисање ресурса захтева одређено време. AWS

API Gateway сервис задужен је за креирање, објављивање, одржавање, надзор и обезбеђивање REST, HTTP и WebSocket API-ја на било којој скали. У оквиру Адверто апликације, користи се као REST API Gateway који рутира захтеве ка одговарајућим Lambda функцијама.



Слика 2.2 – API Gateway и Lambda

### Складиштење фајлова (S3)

Amazon Simple Storage Service, познатији као S3, јесте сервис за складиштење објеката који нуди велику скалабилност, доступност, безбедност и перформансе. Објекат (енгл. object) обухвата датотеку и све метаподатке који ту датотеку описују. Објекти се чувају у оквиру складишта које са назива корпa (енгл. bucket) при чему сваки објекат има кључ (енгл. key) који га јединствено идентификује у оквиру корпе. Кључ обухвата назив, путању на којој се он налази у корпи и његову верзију (уколико је верзионисање укључено унутар корпе). На пример, ако је кључ photos/image234.jpeg онда тај објекат има назив image234.jpg, а налази у директоријуму photos који се налази у корену.



Слика 2.2 – Simple Storage Service

### Аутентификација и ауторизација (Cognito, Security Token Service, Idenity and Access Managment)

Пре него што ускочимо у сам процес аутентификације и ауторизације, потребно је да разумемо основне принципе на основу којих функционише контрола права приступа ресурсима сервиса унутар AWS-а.

Пермисије за акције над ресурсима у оквиру AWS-а се реализују кроз полисе (Policies) које се представљају помоћу JSON синтаксе. Свака полиса садржи идентификатор (Id), верзију (Version) и листу исказа (Statement), при чему сваки исказ чине следећи елементи:

* Principal – листа ентитета на који се односи исказ
* NotPrincipal – листа ентитета на које не треба да делује исказ
* Action – листа акција које на које се односи исказ
* NotAction – листа акција на које не треба да делује исказ
* Effect – ефекат са вредностима Allow (дозвола) или Deny (забрана)
* Resource – листа ресурса на које се односи исказ
* NotResource – листа ресурса на које не треба да делује исказ
* Sid (optional) – опциони идентификатор исказа
* Conditions (optional) – опциона листа услова

Веома је важно напоменути то да унутар полиса, забране увек имају предност у односу на дозволе.

Постоје различите врсте полиса, но за потребе наше апликације нама су потребне само две врсте:

* Identity-based policies – полисе које се каче на кориснике, групе или роле. У оквиру Адверто апликације се каче за роле.
* Resource-based policies – полисе које се каче на ресурсе, на пример за акције над одређеним директоријумом у оквиру S3 Bucket-a.

У оквиру AWS-а, сервиси примају захтеве послате путем HTTP протокола при чему се пре слања, због аутентификације и ауторизације, захтеви потписују приступним кључем помоћу Signature Version 4 (SigV4) алгоритма. Приступни кључ чине идентификатор приступног кључа (Access Key Id) и тајна приступног кључа (Secret Access Key). При коришћењу темпорарних креденцијала о којима ћемо говорити нешто касније, потребно је приложити и токен сесије (Session Token) који служи за верификацију темпорарних креденцијала. Када захтев стигне до сервиса, сервис контактира IAM (Identity And Access Menagment) сервис који је задужен за проверу креденцијала и права приступа ресурсима на основу полиса. Уколико је приступ омогућен, сервис даље обрађује захтев, а у супротном га одбија.

Cognito је сервис који служи за аутентификацију и ауторизацију корисника. Састоји се из две компоненте:

* User Pool – компонента задужена за аутентификацију
* Identity Pool – компонента задужена за ауторизацију

У оквиру User Pool-а могуће је подесити разне функционалности везане за ток регистрације и пријављивања као што су типови креденцијала (корисничко име, мејл, број телефона), критеријум лозинке, обавезни атрибути корисника, двофакторска аутентификација (путем СМС-а или Time-based One-time Password-a), SSO помоћу провајдера као што су Google, Facebook, Amazon и Apple или протокола као што су OpenID и SAML. Такође, унутар User Pool-a, потребно је подесити и User Pool Client апликацију (или више њих) којима се може доделити приступ одређеним информацијама везаним за кориснике, а са којима ће комуницирати клијентске апликације ван Cognito сервиса.

При успешној пријави, корисник као одговор добија три токена:

* Identity Token
* Access Token
* Refresh Token - токен који се шаље заједно са истеклим Access токеном како би се добио нови Access токен

Identity Pool решава проблем ауторизације тако што валидира Identity токен и потом, у случају валидног токена, комуницира са STS сервисом. STS (Security Token Service) генерише темпорарне AWS креденцијале који се састоје од Access Key Id-а, Secret Access Key-а и Session Token-а.

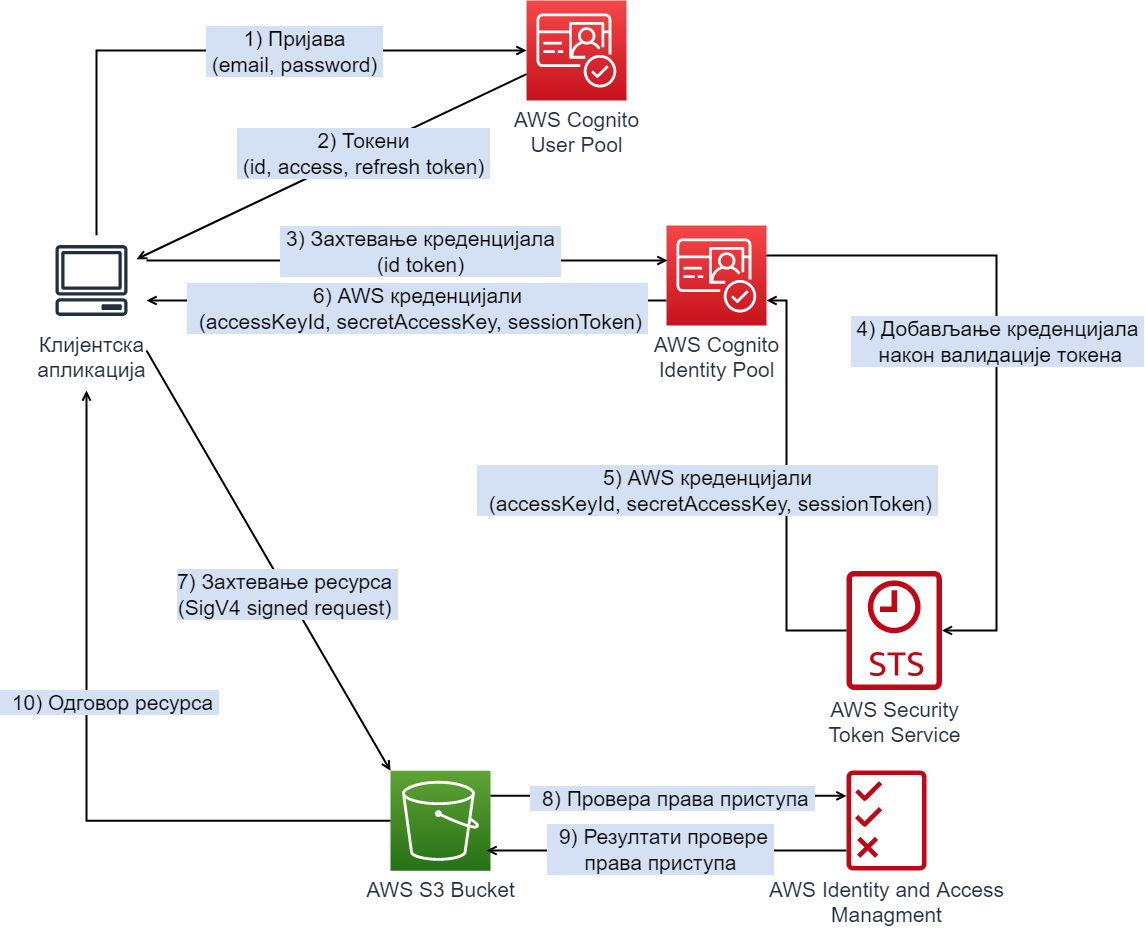
Додељивање рола се подешава у оквиру Identity Pool-а и могуће га је реализовати на 3 начина:

* Додељивањем подразумеване аутентификоване роле
* На основу групе којој корисник припада (Choose role from token)
* На основу правила која посматрају вредности атрибута које корисник поседује (Choose role from rule - слика)

Ако се одаберу други или трећи начин, могуће је да корисник не задовољава критерујме ниједне улоге (не припада групи или нема одговарајуће вредности атрибута), па је потребно одлучити да ли ће се у том случају кориснику доделити подразумевана улога (уколико је дефинисана).

Унутар Identity Pool-а се такође може омогућити да неаутентификовани корисници имају приступ систему, што захтева креирање улоге за неаутентификоване кориснике.

Дијаграм тока аутентификације и ауторизације приказан је на слици 3.1.



Слика 3.1 – Дијаграм тока аутентификације и ауторизације

# SPECIFIKACIJA

## Specifikacija zahteva

U jednom paragrafu najaviti da ovo poglavlje sadrži opis funkcionalnih i nefunkcionalnih zahteva koje je potrebno da sistem omogući. U ovom poglavlju ne spominjete ništa vezano za implementaciju, već samo precizirate šta je sve trebalo implementirati.

### Funkcionalni zahtevi

Opisati funkcionalne zahteve korišćenjem dijagrama slučajeva korišćenja ili skice korisničkog interfejsa.

Nakon dijagrama slučajeva korišćenja bi trebalo svaki slučaj zasebno opisati:

* koji su mu preduslovi
* od kojih koraka se sastoji
* čime rezultuje
* da li postoje neki izuzeci.

### Nefunkcionalni zahtevi

Opisati nefunkcionalne zahteve softverskog sistema. Na primer:

* kakve su performanse sistema neophodne
* na koji način je korisniku potrebno olakšati korišćenje (*user experience*)
* kakav je korisnički interfejs neophodan
* da li je potrebno da *front-end*poseduje *responsive*dizajn
* da li je neophodna višejezičnost (lokalizacija)
* sa kojim formatima ili standardima bi sistem trebao biti kompatibilan
* da li potrebna prenosivost između različitih platformi, odnosno, portabilnost sistema
* koliko bi sistem trebao biti otporan na greške i na koji način bi se trebao oporavljati od grešaka.

## Specifikacija sistema

* U prvom paragrafu najaviti koji modeli su predstavljeni. Na primer, „U ovom poglavlju su prikazani model podataka i arhitektura sistema.“. Specifikacija dizajna se može predstaviti pomoću određenih tipova UML dijagrama, na primer:
  + Dijagrama klasa
  + Dijagrama komponenti
  + Dijagrama sekvenci
  + Dijagrama aktivnosti
  + Dijagrama stanja.

Za kreiranje ovih dijagrama se obično koristi *PowerDesigner*, ali mogu poslužiti i drugi alati, kao što su *Diagrams.net* i *VisualParadigm*.

* U ovom poglavlju ne bi trebalo da se spominje ništa što ima veze sa implementacijom, već se opisuje samo model sistema.

### Model podataka

### Arhitektura sistema

Predstaviti arhitekturu celog sistema (npr. kroz dijagram komponenti). Složenije celine sistema (glavna domenska logika, algoritmi) opisati kroz dijagram klasa i dinamičke dijagrame (sekvenci ili aktivnosti).

# IMPLEMENTACIJA

* Paragraf koji navodi šta je prikazano u ovom poglavlju. Na primer, „U ovom poglavlju je predstavljena implementacija sistema za...“
* Ovo poglavlje služi za opis svih bitnih elemenata implementacije softverskog sistema. Treba da objasnite kako ste u svoj projekat uključili tehnologije nabrojane u poglavlju .
* Možete navoditi konkretne detalje u vezi programskog koda i objasniti kako ste rešavali programerske izazove na koje ste nailazili u realizaciji svog projekta. Za ilustraciju svog rešenja možete ubacivati listinge (fragmente programskog koda).
* Za nazive klasa, metoda, atributa i sličnog, treba koristiti stil Kod.

|  |
| --- |
| **Primer**  Nalistingu 1 jeprikazanametodasaberikojaomogućavaizračunavanjezbirabrojeva. Ovametodazadobijeneparametreaibizračunavanjihovzbirivraćadobijenirezultat.”  publicint saberi(int a, int b) {  return a + b;  }  Listing 1 - Izračunavanjezbirabrojeva |

Primer 5.1Primer listinga

# DEMONSTRACIJA

* Pokažite bitne elemente korišćenja aplikacije.
* Ovaj odeljak može biti poput uputstvo za korišćenje sistema.
* Opišite jedan scenario (ili više) pri korišćenjuvaše aplikacije. Korak po korak prikažite kako korisnici stupaju u interakciju s vašom aplikacijom. Ubacite slike sa izgledom ekrana, koji će ilustrovati važne faze u njegovom korišćenju. Ovaj scenario (ili više njih) koji je ovde predstavljen bi bilo poželjno da bude pokriven i dinamičkim dijagramima u Poglavlju 4, i delimično ili potpuno pokriven listinzima u Poglavlju 5.

# ZAKLJUČAK

* Rekapitulacija glavnih poenti u radu:
  + Rešavani problem i motivacija za njegovo rešavanje
  + Grub opis rešenja
  + Osvrnuti se na poglavlje 2 (Pregled sličnih sistema) i zaključiti šta je to što ste vi uradili bolje ili drugačije od drugih.
* Opis mogućih pravaca daljeg proširivanja/unapređenja/otklanjanje identifikovanih nedostataka rešenja

# LITERATURA

1. *Cider* razvojno okruženje <https://cider.readthedocs.io/en/latest/> [Datum pristupa 13.08. 2016].
2. Abraham, A., 2005. Rule‐Based expert systems. *Handbook of measuring system design*.
3. Gabriel, R.P. and Pitman, K.M., 1988. Endpaper: Technical issues of separation in function cells and value cells. *Lisp and Symbolic Computation*, *1*(1), pp.81-101.

# BIOGRAFIJA

Лука Матић је рођен 18.04.1999. године у Сремској Митровици. Завршио је Гимназију "Стеван Пузић" у Руми. Факултет техничких наука у Новом Саду, смер Рачунарство и аутоматика, усмерење Примењене рачунарске науке и информатика, уписао је школске 2018/2019. године. Положио је све испите прописане планом и програмом и стекао услов за одбрану завршног рада.