Универзитет у Београду

Факултет организационих наука

Завршни рад

Софтверски систем филмског форума у Jakarta EE окружењу

Ментор:

Др Милош Милић

Студент:

Вељко Благојевић 353/2019

Београд, 2023. године

Садржај

1. Увод ... 1
2. Преглед коришћених технологија ... 3
   2.1. Јава ... 3
      2.1.1. Концепти објектно-оријентисаног програмирања у Јава програмском језику ... 3
      2.1.2. Spring радни оквир ... 4
      2.1.3. Софтверски патерни ... 6
      2.1.4. Перзистенција података ... 8
      2.1.5. RESTful веб сервиси ... 10
      2.1.6. Заштита веб апликација ... 11
      2.1.7. Spring Boot ... 12
   2.2. JavaScript ... 13
      2.2.1. TypeScript ... 13
      2.2.2. React ... 13
      2.2.3. Једностраничне апликације ... 14
      2.2.4. HTTP клијент ... 14
3. Студијски пример ... 15
   3.1. Прикупљање корисничких захтева ... 16
      3.1.1. Вербални опис ... 16
      3.1.2. Случајеви коришћења ... 17
   3.2. Анализа ... 26
      3.2.1. Понашање софтверског система - Системски дијаграми секвенци ... 26
      3.2.2. Понашање софтверског система – уговори ... 41
      3.2.3. Структура софтверског система ... 43
   3.3. Пројектовање ... 52
      3.3.1. Пројектовање корисничког интерфејса ... 54
      3.3.2. Пројектовање апликационе логике ... 81
      3.3.3. Пројектовање брокера базе података ... 98
      3.3.4. Пројектовање складишта података ... 100
      3.3.5. Коначна архитектура софтверског система ... 105
   3.4. Имплементација ... 106
   3.5. Тестирање ... 109
4. Закључак ... 110
5. Литература ... 111

Попис слика

Слика 1 Модули Spring радног оквира [5] 5
Слика 2 Модел случајева коришћења 17
Слика 3 Концептуални (доменски) модел студијског примера 43
Слика 4 Логичка структура и понашање софтверског система 44
Слика 5 Тронивојска архитектура [2] 52
Слика 6 Архитектура софтверског система 53
Слика 7 Структура корисничког интерфејса [2] 54
Слика 8 Форма за пријаву корисника 54
Слика 9 Форма за регистрацију корисника 55
Слика 10 Почетна страна 55
Слика 11 Страна са листом филмова 56
Слика 12 Форма са детаљима филма 57
Слика 13 Проналажење филма, алтернативни сценарио 1 57
Слика 14 Форма са листом особља 58
Слика 15 Форма са детаљима особе 59
Слика 16 Проналазак особе, алтернативни сценарио 1 59
Слика 17 Форма са листом филмова 60
Слика 18 Форма са детаљима филма 61
Слика 19 Додавање у листу жеља 61
Слика 20 Форма корисника са листом жеља 62
Слика 21 Додавање у листу жеља, алтернативни сценарио 1 62
Слика 22 Додавање у листу жеља, алтернативни сценарио 2 62
Слика 23 Форма са листом филмова 63
Слика 24 Форма са детаљима филма 64
Слика 25 Уклањање из листе жеља 64
Слика 26 Форма корисника са листом жеља 65
Слика 27 Уклањање из листе жеља, алтернативни сценарио 1 65
Слика 28 Уклањање из листе жеља, алтернативни сценарио 2 65
Слика 29 Форма са листом филмова 66
Слика 30 Форма са детаљима филма 67
Слика 31 Додавање рецензије 67
Слика 32 Унос података рецензије 68
Слика 33 Додата рецензија 68
Слика 34 Додавање рецензије, алтернативни сценарио 1 69
Слика 35 Додавање рецензије, алтернативни сценарио 2 69
Слика 36 Форма са листом филмова 70
Слика 37 Форма са детаљима филма 71
Слика 38 Измена рецензије 72
Слика 39 Унос нових података рецензије 72
Слика 40 Измењена рецензија 72
Слика 41 Измена рецензије, алтернативни сценарио 1 73
Слика 42 Измена рецензије, алтернативни сценарио 2 73
Слика 43 Форма са листом филмова 74

Слика 44 Форма са детаљима филма 75
Слика 45 Измена филма 76
Слика 46 Провера унесених података филма 76
Слика 47 Измењени подаци филма 77
Слика 48 Измена података филма, алтернативни сценарио 1 78
Слика 49 Измена података филма, алтернативни сценарио 1 78
Слика 50 Форма за пријаву корисника 79
Слика 51 Пријава корисника 79
Слика 52 Форма са детаљима корисника 80
Слика 53 Пријава корисника, алтернативни сценарио 1 80
Слика 54 Концептуални модел 93
Слика 55 Пакет са доменским класама 93
Слика 56 Доменска класа Actor 94
Слика 57 Доменска класа CrewMember 94
Слика 58 Енумерација Gender 94
Слика 59 Доменска класа Genre 94
Слика 60 Доменска класа Movie 95
Слика 61 Доменска класа Person 95
Слика 62 Доменска класа Review 96
Слика 63 Енумерација Role 96
Слика 64 Доменска класа User 97
Слика 65 Доменска класа WatchList 97
Слика 66 Интерфејс CrudRepository 98
Слика 67 Хијерархија репозиторијума 99
Слика 68 Табеле базе података 100
Слика 69 Коначна архитектура софтверског система 105
Слика 70 Пакети и класе серверске апликације (први део) 106
Слика 71 Пакети и класе серверске апликације (други део) 107
Слика 72 Пакети и класе клијентске апликације (први део) 108
Слика 73 Пакети и класе клијентске апликације (други део) 108

Попис дијаграма

Дијаграм 1 Преглед детаља филма 27
Дијаграм 2 Преглед детаља филма, алтернативни сценарио 1 27
Дијаграм 3 Преглед детаља особља 28
Дијаграм 4 Преглед детаља особља, алтернативни сценарио 1 28
Дијаграм 5 Додавање филма у листу жеља 29
Дијаграм 6 Додавање у листу жеља, алтернативни сценарио 1 30
Дијаграм 7 Додавање у листу жеља, алтернативни сценарио 2 30
Дијаграм 8 Уклањање филма из листе жеља 31
Дијаграм 9 Уклањање из листе жеља, алтернативни сценарио 1 32
Дијаграм 10 Уклањање из листе жеља, алтернативни сценарио 2 32
Дијаграм 11 Додавање рецензије 33
Дијаграм 12 Додавање рецензије, алтернативни сценарио 1 34
Дијаграм 13 Додавање рецензије, алтернативни сценарио 2 34
Дијаграм 14 Измена рецензије 35
Дијаграм 15 Измена рецензије, алтернативни сценарио 1 36
Дијаграм 16 Измена рецензије, алтернативни сценарио 2 36
Дијаграм 17 Измена филма 37
Дијаграм 18 Измена филма, алтернативни сценарио 1 38
Дијаграм 19 Измена филма, алтернативни сценарио 2 38
Дијаграм 20 Пријава корисника на систем 39
Дијаграм 21 Пријава корисника на систем, алтернативни сценарио 1 39
Дијаграм 22 Уговор 1 82
Дијаграм 23 Уговор 2 83
Дијаграм 24 Уговор 3 84
Дијаграм 25 Уговор 4 85
Дијаграм 26 Уговор 5 86
Дијаграм 27 Уговор 6 87
Дијаграм 28 Уговор 7 88
Дијаграм 29 Уговор 8 89
Дијаграм 30 Уговор 9 90
Дијаграм 31 Уговор 10 91
Дијаграм 32 Уговор 11 92

Попис табела

Табела 1 Movie 46
Табела 2 User 47
Табела 3 Review 47
Табела 4 WatchList 48
Табела 5 Movie_WatchList 48
Табела 6 Genre 49
Табела 7 Movie_Genre 49
Табела 8 Person 50
Табела 9 Actor 51
Табела 10 CrewMember 51
Табела 11 Складиште података, табела Actor 101
Табела 12 Складиште података, табела CrewMember 101
Табела 13 Складиште података, табела Genre 101
Табела 14 Складиште података, табела Movie 102
Табела 15 Складиште података, табела Movie_Genre 102
Табела 16 Складиште података, табела Movie_WatchList 102
Табела 17 Складиште података, табела Person 103
Табела 18 Складиште података, табела Review 103
Табела 19 Складиште података, табела User 103
Табела 20 Складиште података, табела WatchList 104

1. Увод

Филм представља серију слика, које брзим смењивањем стварају илузију кретања уз помоћ *phi* феномена. Због релативно скорашњег зачетка и велике зависности од науке и технологије филм се сматра најмодернијом уметношћу али његова суштина датира још од праисторије. Покретне слике дају филму могућност за пренос наративне идеје, приче и осећања.

Ричото Канудо у свом тексту „Теорија седам уметности“ (франц. *Théorie de sept arts*), објављеном 1911. године, тврди да је филм нова уметност која је уједно и најважнија. Канудо у том делу сматра да је филм синтеза свих античких уметничких форми: архитектуре, скулптуре, сликарства, музике и поезије. [1]

Филм преноси поруке, тематику и причу при чему су људи наклоњени да причају о истима. Филмска критика представља природни след након настанка филма и дефинише се као анализа и процена филма. Критика може да има велики утицај на популарност и одзив публике, чиме могу да промовисане филмове доведу у пропаст, као и да независно издате филмове доведу до неочекиваног успеха, што говори о њеном значају у кинематографској индустрији.

Развојем Интернета, појефтињењем технологије за продукцију филма, као и поједностављењем метода масовне дистрибуције, смањио се праг који је неопходно превазићи како би се један филм пласирао у јавност. Велики број филмова је проистекао на основу претходно поменутог, због тога се још већи значај даје критикама, као и „на каквом гласу“ се налази филм у друштву.

Данас, како је појавом Интернета упрошћен приступ огромној количини филмског садржаја, потребно га је и испратити адекватном дигиталном заједницом за критику филмова. Управо овде софтверски системи долазе до значаја. Ширење мишљења критичара, њихових рецензија и основних информација о филмовима је произвело директну потребу за израду оваквог софтверског система. Како је број издатих филмова, глумаца и издавачких кућа све већи, појединац једноставно не може све сам да испрати и посвети своје време адекватним филмовима, тако помоћ заједнице и лако доступних информација везане за ову уметност постају важније и потребније. Идеја овог софтверског система је управо дистрибуција критика, информација о филмовима, њиховим жанровима, глумцима и особљем.

Анализа, пројектовање и имплементација једног софтверског решења може обухватати многе приступе, технологије и технике. Како је комбинација свих ових елемената изузетно велика, овај завршни рад ће се фокусирати на оне које су довољно уопштене, генеричке и широм распрострањене да је могуће извести на светло велики број програмерских идеја.

Један од приступа је упрошћена Ларманова метода развоја софтвера. Животни циклус софтверског система се састоји из следећих фаза:

1. Прикупљање захтева од корисника
2. Анализе
3. Пројектовања
4. Имплементације
5. Тестирања

Развој софтверског система код упрошћене Ларманове методе има јасан, логички след где се свака фаза надовезује на претходну. [2]

У другом поглављу овог рада је дат преглед технологија које су коришћење током израде датог софтверског система, а оне укључују Jakarta EE технологије и Spring Boot алата и оквира за израду серверске стране апликације, React за израду корисничког интерфејса, као и MySQL базе података која служи за чување података.

У трећем поглављу је приказан развој софтверског система филмског форума кроз упрошћену Ларманову методу развоја софтвера. [2]

У четвртом поглављу је приказан закључак рада и предлози даљег унапређења филмског форума као софтверског система.

Пето поглавље представља преглед литературе која је коришћена при изради овог рада.

2. Преглед коришћених технологија

У овом поглављу су приказане технологије које су коришћене при развоју софтверског система филмског форума.

Популаран приступ имплементацији једног софтверског решења у модерном свету су једностраничне апликације (енгл. *Single Page Application – SPA*) које се извршавају у веб окружењу. Због модерности и великог броја бенефита овај тип имплементације ће се примењивати на посматраном пројекту. Технологије које су примењене у о овом раду су Spring Boot и React оквири за развој веб апликација.

2.1. Јава

Јава је технологија која уједно представља и платформу као и објектно-оријентисан језик. Јава је састављена од скупа технологија и алата који се користе за развој програма. Апликација која је написана у Јава програмском језику се може покретати на различитим платформама, што је омогућено уз помоћ Јава Виртуелне Машине (енгл. *Java Virtual Machine - JVM*). Јава компајлер преводи програм из .java датотеке у бајт-код који се даље чува под екстензијом .class, након чега се такав код интерпретира у Јава Виртуелној Машини, чиме је омогућена потпуна платформска независност.

2.1.1. Концепти објектно-оријентисаног програмирања у Јава програмском језику

Програми који су написани са објектно-оријентисаном парадигмом користе објекте као основу при пројектовању и развоју самог софтверског решења. Ова парадигма се заснива на различитим техникама, као што су наслеђивање, модуларност, полиморфизам и учаурење (енгл. *Encapsulation*). [3]

Јава је један од програмских језика који је заснован на објектно-оријентисаној парадигми и користи класе и објекте као своје основне градивне блокове. Класа је шаблон за креирање објеката, обезбеђује почетне вредности за променљиве као и имплементације понашања кроз методе. Објекат је инстанца класе и уобичајени начин његовог креирања је уз помоћ конструктора класе.

Темељи који сваки објектно-оријентисани језик поседује су: [4]

1. Апстракција и скривање детаља. Апстракција је поступак раздвајања битног од небитног при чему је неопходно уочити податке и везе који су битни за дати домен проблема. Детаљи се занемарују на неком нивоу апстракције како би се број концепата са којима се пројектант суочава свео на разумну меру.
2. Енкапсулација је поступак обједињавања стања и понашања у једну целину. Крајњи резултат енкапсулације је класа.
3. Наслеђивање је могућност хијерархијске организације класа. Овим поступком једна класа може да има комплетан садржај друге класе, притом тај садржај може проширити или редефинисати.
4. Полиморфизам је концепт помоћу којег се може извести једна акција на различите начине. У Јава програмском језику постоји два типа полиморфизма, у време извршавања и у време компајлирања.

2.1.2. Spring радни оквир

Spring радни оквир (енгл. *framework*) представља често коришћен Јава оквир који је заснован на концептима инверзије контроле и управљања контејнерима. Spring радни оквир има следеће карактеристике:

1. Инверзија контроле (енгл. *Inversion of Control - IoC*): Spring користи принцип инверзије контроле како би олакшао управљање зависностима између објеката. Уместо да објекти сами стварају и одржавају везе између њих, Spring преузима ту одговорност. Принцип повезивања међусобно зависних објеката у коду се назива уметање зависности (енгл. *Dependency Injection - DI*). На тај начин је олакшано тестирање, али је и повећана флексибилност и поновно коришћење кода.
2. Аспектно оријентисано програмирање (енгл. *Aspect Oriented Programming - AOP*) је парадигма којом Spring омогућава програмерима да примене аспекте на свој код како би добили додатну функционалност. Аспектно оријентисано програмирање омогућава модуларност и решавање одређених функционалности које се могу применити на више делова апликације.
3. Модуларност и слојевита архитектура пружају могућност модуларног развоја апликација кроз концепте попут компоненти, сервиса и репозиторијума. Ово олакшава организацију кода, побољшава читљивост и одрживост.
4. Интеграција са разним технологијама је битан аспект било каквог алата, при чему Spring омогућава подршку за разне библиотеке, као што су Java Database Connection (JDBC) за приступ базама података, RESTful веб сервисе, Java Message Service за обраду порука, Hibernate за објектно-релационо мапирање (енгл. *Object Relational Mapping - ORM*), и многе друге.
5. Једноставно тестирање које је омогућено уз помоћ концепта инверзије контроле и уметања зависности. Програмерима је због тога омогућено изоловање компоненти као и замена имплементација ради ефикасног тестирања.

Spring радни оквир се састоји од различитих модула који пружају специфичне функционалности, као што су Spring Core, Spring MVC (*Model-View-Controller*), Spring Data, Spring Security и многи други. Ови модули се могу комбиновати како би се изградио комплетан софтверски систем. Подскуп модула Spring радног оквира је приказан на наредној слици.

Слика 1 Модули Spring радног оквира [5]

Предности коришћења Spring радног оквира укључују флексибилност, скалабилност, поновно коришћење кода, добру подршку за тестирање и употребу од стране велике заједнице софтверских инжењера. Spring радни оквир је постао стандардни избор за развој Јава апликација, како у индустрији тако и у академском свету.

2.1.3. Софтверски патерни

Софтверски патерни (енгл. *Design Patterns*) представљају опште, поново употребљиво решење за честе проблеме који се срећу приликом развоја софтвера. Рихл (енгл. *Dirk Riehle*) и Цулиговен (енгл. *Heinz Zullighoven*) тврде да патерни представљају апстракцију конкретног облика која се може поново употребити у специфичним контекстима. [6] Патерни дефинишу шаблоне који су довољно општи да се могу употребити на великом броју доменских проблема, али је сваки довољно специфичан да се зна тачно коју врсту проблема решава.

Spring радни оквир употребљава више софтверских патерна и принципа у својој архитектури и његова правилна употреба захтева добро разумевање софтверских патерна и принципа као што су синглтон (енгл. *Singleton*) и уметање зависности (енгл. *Dependency injection*).

2.1.3.1. Синглтон

Синглтон патерн је креациони патерн који обезбеђује класи само једно појављивање и глобални приступ до ње. [6] Самим тим за одређену класу се може сматрати да ће се објекат дате класе инстанцирати само једном у читавом раду апликације и бити искоришћен у свим деловима апликације.

У општем смислу, Синглтон патерн се може имплементирати праћењем наредних корака: [7]

1. Дефинисање статичке променљиве која чува једну инстанцу класе коју је потребно трансформисати у синглтон.
2. Конструктор је потребно дефинисати да има приватан модификатор приступа, чиме се омогућава да га једино дата класа може позвати.
3. Дефинисање јавне, статичке методе која ће заправо инстанцирати објекат први пут и вратити ту исту инстанцу када год је позвана.

У контексту Spring радног оквира омогућено је једноставно управљање синглтон објектима кроз контејнер за инверзију контроле (енгл. *Inversion of Control Container*). У Spring радном оквиру, класе које су означене одговарајућим анотацијама (нпр. @Component, @Service, @Repository) постају синглтон инстанце.

Spring контејнер се брине о креирању и управљању синглтон инстанцама. Приликом првог захтева за одређеним објектом, Spring радни оквир ће га инстанцирати и сачувати у контејнеру. Након тога, сваки следећи захтев за објектом исте класе ће добити претходно креирану инстанцу.

Важно је напоменути да је у Spring радном оквиру могуће конфигурисати и друге опсеге животног циклуса објеката (нпр. опсег прототипа, опсег сесије, опсег захтева) путем одговарајућих анотација и конфигурација.

Коришћење синглтон софтверског патерна у Spring радном оквиру омогућава ефикасно дељење ресурса, ефикасно управљање меморијом и поједностављено управљање стањем апликације.

2.1.3.2. Уметање зависности

Уметање зависности (енгл. *Dependency Injection - DI*) представља принцип доделе зависности објекту путем методе или конструктора, уместо директног инстанцирања унутар саме класе. [8] Уметање зависности омогућава да се зависност према другом објекту додели од стране независне компоненте. Основна идеја иза датог принципа је да објекат не би требало да буде одговоран за креирање својих зависности, већ да би требало да их добија споља, од стране друге компоненте система.

DI се користи како би се смањила зависност између компоненти и побољшала модуларност, флексибилност и тестабилност софтвера.

Уметање зависности се може постићи на више начина: [9]

- Уметање зависности путем конструктора (енгл. *Constructor injection*): Односи се на уметање зависности кроз конструктор класе. Конструктор класе има параметре који представљају зависности, а те зависности се прослеђују приликом креирања објекта. Ова техника осигурава да објекат буде у потпуности иницијализован са свим својим зависностима пре него што буде коришћен.
- Уметање зависности путем методе (енгл. *Method injection*): Односи се на уметање зависности кроз методе објекта. Објекат има методе које прихватају зависности као аргументе, а те зависности се прослеђују након креирања објекта. Ова техника омогућава флексибилност при постављању зависности, јер се зависности могу мењати или ажурирати у било ком тренутку.
- Уметање зависности путем *сетер* методе (енгл. *Setter injection*): Овај начин је сличан уметању зависности путем методе, али користи специфичне методе за постављање зависности.

Бенефити принципа уметања зависности укључују смањење високо повезаног (енгл. *tightly coupled*) кода, једноставнију замену зависности, олакшано тестирање и бољу модуларност апликације. Уметање зависности такође промовише принцип инверзије контроле (енгл. *Inversion of Control - IoC*), где контрола над креирањем и управљањем објектима прелази на контејнер и управо се на овим принципа и заснива Spring радни оквир.

2.1.4. Перзистенција података

У модерном развоју софтвера, управљање подацима је кључни аспект сваке апликације. Систем базе података је рачунарски програм који кориснику дозвољава да управља подацима (да чита, ажурира, заштити податке). Први и основни аспект базе података је начин представљања података. Постоје више парадигми и свака има своје предности и мане, а неке од њих су мрежни модел, хијерархијски модел и нерелациони модел. Међутим најширу примену у комерцијалним решењима је добио релациони модел, где се подаци чувају у табелама, названим релацијама, при чему је модел заснован на релационом рачуну. Овај модел углавном користи SQL као декларативни, упитни језик за манипулацију подацима. [10]

Општи консензус је да програмери преферирају да раде са објектима у програмском коду за рад са подацима, наспрам SQL упитног језика при приступу подацима. Потреба за таквим радом је испраћена новим алатима који дозвољавају управо такав приступ. Системи за објектно-релационо мапирање (енгл. *Object relational mapping - ORM*) су технологије и алати који имају могућност аутоматске трансформације података из релационих табела у објекте програма и обрнуто. [11]

2.1.4.1. Hibernate радни оквир

Hibernate радни оквир је имплементација система за објектно-релационо мапирање у Јава окружењу. Овај пројекат је софтвер отвореног кода (енгл. *Open-source code*) и представља комплетно, функционално решење за управљање релационом базом података. [12]

Hibernate радни оквир користи мета-податке како би генерисао SQL упите у време извршавања кода. Тиме је скроз избегнуто писање SQL упита и сво управљање и манипулацију над базом података програмер врши уз помоћ метода које су дате унутар поменутог радног оквира.

Hibernate радни оквир поседује неколико кључних карактеристика: [12]

- Мапирање између објеката и табела: Hibernate омогућава дефинисање мапирања између објеката и табела у бази података. Ово мапирање се може обавити помоћу анотација директно на класама или путем XML конфигурација.
- Управљање везама између табела.
- Управљање трансакцијама чиме је обезбеђен интегритет података на декларативан начин.
- Касно учитавање (енгл. *Lazy Loading*) је концепт где се подаци прикупљају само по потреби чиме се повећавају перформансе апликације и смањује оптерећење базе података.
- Кеширање (енгл. *caching*) зарад оптимизованог приступа често приступаним подацима.

Детаљно познавање програмског интерфејса овог радног оквира је кључно за његову адекватну употребу. Hibernate радни оквир пружа класе и методе које су одговорне при извршавању *CRUD* (енгл. *Create Read Update Delete - CRUD*) операција чиме се успоставља директна веза између пословне логике апликације и самог оквира. За те операције су коришћење класе Session, Transaction и Query. Постављање конфигурације алата се врши помоћу класе Configuration. [12]

2.1.4.2. Spring Data JPA

Spring Data JPA представља подскуп Spring Data алата и задужен је за имплементацију слоја приступа базама података заснованим на релационом моделу. Spring Data JPA пружа стандардизован приступ објектно-релационом мапирању, при чему се могу користити разни ORM алати за имплементацију приступа подацима. Уобичајена имплементација је уз помоћ Hibernate радног оквира. JPA је увео нови начин писања упита над релационим табелама коришћењем JPQL (Java Persistence Query Language) језика, специјализованог упитног језика унутар јава програмског кода. [13]

Spring Data JPA такође пружа подршку за једноставну обраду података коришћењем концепата као што су сортирање и пагинација.

Још једна функционалност JPA алата која повећава ниво апстрактности и олакшава рад програмерима су Јава „упитне методе“ (енгл. *Query methods*). Ово су методе које у свом називу садрже специфичне кључне речи, чиме се путем механизма рефлексије омогућава њихова анализа и креирање SQL упита. [14]

2.1.5. RESTful веб сервиси

REST (*Representational State Transfer*) је стил архитектуре који дефинише скуп принципа на основу којих се пројектују веб сервиси. Стандард је да се користи HTTP протокол за комуникацију и пренос ресурса. При пројектовању REST веб сервиса потребно је испоштовати одређена правила. Веб сервиси морају бити без стања што значи да сваки HTTP захтев садржи све потребне информације како би сервер могао да обради захтев, због чега се одговорност у чувању сесије пребацује на клијента. Веб сервиси би требало да адекватно користе HTTP методе (GET, POST, PATCH итд.). Изложене путање би требало да буду интуитивне, а ресурси се преносе путем XML или JSON нотације. [15]

2.1.5.1. Spring Web

Spring Web модул је део Spring радног оквира који омогућава развој веб апликација у Јава окружењу. Модул пружа подршку за различите технологије и архитектуре, укључујући RESTful веб сервисе. Користећи Spring MVC (енгл. *Model View Controller*) архитектуру, програмери могу дефинисати контролере који ће обрађивати HTTP захтеве и генерисати одговоре у JSON или XML формату, што је често случај код RESTful сервиса. Spring Web такође пружа анотације и конфигурационе опције које олакшавају дефинисање рута, конверзију података и валидацију уноса.

2.1.6. Заштита веб апликација

Приступ Интернету, размена података и трансфер ресурса су процеси кроз које сваки корисник рачунара редовно пролази. Заштита имају кључну улогу у одржавању интернета сигурним и може јој се приступити на више начина. Један од приступа је вршење аутентификације и ауторизације захтева између клијентског и серверског дела апликације. Аутентификација је механизам заштите којим се одређује идентитет корисника пре него што се дозволи приступ садржају. Ауторизација представља технику одређивања нивоа овлашћености датог корисника.

2.1.6.1. JSON Web Token

JWT (*JSON Web Token*) је један од стандарда који се користи при аутентификацији и ауторизацији. JWT је стандард за сигуран пренос података између две стране у облику JSON објекта који се енкриптује.

Састоји се од три дела: заглавља (енгл. *Header*), корисничких података (енгл. *Payload*) и потписа (енгл. Signature). Заглавље садржи информације о алгоритму за потписивање, Payload садржи податке о кориснику или друге релевантне податке, а потпис се користи за верификацију интегритета токена. [16]

JWT има неколико предности у односу на традиционални приступ коришћењем сесија и колачића (енгл. *cookies*). JWT је само-одржив, што значи да садржи све потребне податке за проверу аутентичности и ауторизације, тако да сервер не мора чувати додатне информације уз помоћ сесија.

2.1.6.2. Spring Security

Spring Security је модул у Spring радном оквиру који пружа свеобухватну подршку за имплементацију сигурности у Јава апликацијама. Главни циљ Spring Security модула је заштита апликација од различитих безбедносних претњи, као што су неовлашћени приступ, могућност аутентификације и ауторизације корисника, заштита ресурса и управљање сесијама. [16]

Spring Security пружа различите механизме за аутентификацију корисника. То укључује подршку за традиционалну аутентификацију, где корисници уносе своје корисничко име и лозинку, као и подршку за алтернативне методе аутентификације попут *OAuth, OpenID* и других. Такође подржава интегрисање са екстерним системима за управљање налозима.

Комбинација Spring Security модула и JWT токена омогућава безбедну аутентификацију и ауторизацију корисника у веб апликацијама. Spring Security пружа механизме за верификацију и управљање JWT токенима, док JWT обезбеђује сигуран пренос података о кориснику између клијента и сервера. Овај приступ пружа флексибилност, скалабилност и интероперабилност у развоју сигурних веб апликација. [17]

2.1.7. Spring Boot

Spring Boot је оквир за развој Јава апликација који олакшава изградњу самосталних, само-одрживих и високо продуктивних софтверских решења. Основна идеја иза Spring Boot радног оквира је елиминисање комплексности конфигурације и повећање продуктивности програмера.

Кључне карактеристике Spring Boot радног оквира су: [18]

- Конвенција над конфигурацијом: Spring Boot примењује конвенције уместо да захтева обимну конфигурацију. Већина поставки се аутоматски конфигурише према стандардним конвенцијама, што програмерима штеди време и смањује могућност настанка грешке.
- Уграђени сервлет контејнер: Spring Boot има уграђени сервлет контејнер, тако да апликација може бити покренута као самосталан *JAR* фајл без потребе за спољним веб сервером. То поједностављује дистрибуцију и имплементацију апликација.
- Аутоматска конфигурација: Spring Boot аутоматски конфигурише велики број компоненти на основу зависности које су додате у пројекат. На пример, ако се користи Spring Data JPA, Spring Boot аутоматски конфигурише везу са базом података на основу података из конфигурације.
- Управљање зависностима: Spring Boot омогућава једноставно управљање зависностима кроз Maven или Gradle алате. Помоћу стартер зависности, програмери могу лако додавати и уклањати потребне модуле и библиотеке.
- Интеграција са Spring екосистемом: Spring Boot је интегрисан са другим пројектима из Spring екосистема, као што су Spring MVC, Spring Data, Spring Security и многи други. То пружа програмерима широк спектар функционалности и алата за развој комплетних апликација.

Spring Boot такође подржава разне технологије и протоколе, као што су RESTful веб сервиси, WebSocket, микросервиси, планирање задатака (scheduling), обрада порука (messaging) и још много тога.

2.2. JavaScript

JavaScript је програмски језик намењен за веб странице и покретање у интернет прегледачу. JavaScript је динамичан, слабо типизиран и интерпретиран језик. Језик је вишеструке парадигме јер подржава и објектно-оријентисани, императивни и функционални приступ писања програма. Широка распрострањеност и велика заједница су довели и овај језик на сервер уз помоћ NodeJS алата, додали строгу типизираност уз помоћ TypeScript пројекта и још много тога, али ипак је главна улога JavaScript језика извршавање клијентског дела апликације у веб прегледачу. Самим тиме су и настали бројни радни оквири који олакшавају израду баш таквих клијентских веб апликација.

2.2.1. TypeScript

TypeScript је језик који представља надградњу стандардног JavaScript језика, увођењем синтаксе за статичко типизирање. На овај начин је уведена могућност коришћења типова података. Њиховом употребом омогућава се строга типизација што непосредно олакшава рад програмера. Овим поступком је омогућен приказ одређених типова грешака и пре самог покретања апликације, што иницијално није било могуће са стандардним JavaScript језиком. TypeScript код се свакако после, путем процеса превођења (енгл. *transpile*) и компајлирања преводи у чист JavaScript код чиме се одржава компатибилност на високом нивоу.

2.2.2. React

React је JavaScript радни оквир за израду корисничких интерфејса (енгл. *User Interface - UI*) веб апликација. Отвореног је кода и развијен од стране Facebook компаније. React се фокусира на брзину, ефикасност и скалабилност у изградњи модерних интерактивних веб апликација.

Једна од кључних карактеристика React радног оквира је архитектура заснована на компонентама. Развој у React оквиру се заснива на декомпозицији корисничког интерфејса на мање, поновно употребљиве компоненте. Свака компонента може имати своје стање (енгл. *state*) и своје особине (енгл. *props*), што омогућава флексибилно управљање подацима и интеракцијом са корисником. Компоненте се могу једноставно комбиновати и уређивати, што олакшава организацију и одржавање кода. Функционалне компоненте су једноставне JavaScript функције које служе као основне градивне јединице корисничког интерфејса у React оквиру. Једна од предности функционалних компонената је то што могу користити *React Hooks* функције. Ове функције обезбеђују додатне функционалности и могућности управљања стањем у функционалним компонентама. На пример, популарна функција "useState" се користи за декларисање и управљање стањем у функционалној компоненти.

Једна од главних предности React радног оквира је виртуелни DOM (енгл. *Virtual Document Object Model*). DOM је хијерархијска структура објеката који чине структуру саме странице. Виртуелни DOM је ефикасна репрезентација стварне DOM структуре и користи се за оптимизацију перформанси. Када се стање компоненте промени, React прво ажурира виртуелни DOM, а затим упоређује тај виртуелни са стварним како би пронашао само промене које треба применити. Овим се минимизује потреба за директном манипулацијом DOM структуре и омогућава се бржи приказ корисничког интерфејса са штедљивијом употребом ресурса.

React такође подржава једносмерну везу података. То значи да се подаци крећу од врха хијерархије компонената ка дну хијерархије. Када се стање промени у горњој компоненти, ова промена се пропагира кроз хијерархију до доњих компоненти. Ова једносмерна комуникација поједностављује праћење стања апликације и смањује вероватноћу настанка грешке.

Поред тога, React има богат екосистем библиотека и алата. Постоје различите библиотеке за рутирање (нпр. React Router), управљање стањем (нпр. Redux) и стилизацију (нпр. Styled Components), које се могу лако интегрисати у React апликације.

Узимајући у обзир све ове карактеристике, React је постао популаран избор за развој модерних веб апликација. Његова једноставност, ефикасност и добра подршка од стране заједнице чине га снажним оквиром за изградњу скалабилних и интерактивних корисничких интерфејса.

2.2.3. Једностраничне апликације

Једностранична апликација (енгл. *Single Page Application - SPA*) је врста веб апликације која се извршава у једној HTML страници, а корисничко искуство се постиже динамичким ажурирањем садржаја на истој страници без потребе за поновним учитавањем целе странице. Уместо тога, једностраничне апликације користи JavaScript за управљање стањем апликације и приказивање различитих погледа кориснику.

У традиционалним вишестраничним апликацијама, сваки пут када корисник изврши интеракцију са апликацијом, захтева се нова HTML страница са сервера, што резултира учитавањем целокупног садржаја апликације. Ово може бити спор процес, посебно код апликација са комплексним интерфејсима. Са једностраничном апликацијом, иницијална страница се учитава само једном, а касније се само ажурирају делови који се мењају, чиме се постиже брже и флуидније корисничко искуство.

2.2.4. HTTP клијент

За позивање са серверском апликацијом путем REST API интерфејса потребан је HTTP клијент. HTTP клијент је компонента која шаље захтеве према серверској страни, прихвата одговоре и адекватно их обрађује.

У овом пројекту је изабрана библиотека axios као једна од JavaScript имплементација HTTP клијента. Axios је популарна JavaScript библиотека за слање HTTP захтева из веб прегледача. Она омогућава једноставно и ефикасно комуницирање са сервером, извршавање AJAX позива и манипулацију подацима. Једна од главних предности axios библиотеке је његова једноставност употребе. За употребу axios библиотеке, прво је потребно инсталирати библиотеку и укључити је у пројекат. Након тога, могуће је извршити HTTP захтев једноставном синтаксом.

3. Студијски пример

Ово поглавље се односи на детаљно објашњење саме израде студијског примера, при чему је посебно потребно обратити пажњу на упрошћену Ларманову методу развоја софтвера јер је помоћу ње извршен цео процес развоја овог софтверског решења.

Упрошћена Ларманова метода се састоји из пет основних фаза: [2]

1. Прикупљање захтева: У овој фази, тим за развој се сусреће са клијентом и прикупља све захтеве и потребе за софтвером. Ова фаза укључује и идентификацију стејкхолдера и анализирање њихових захтева и очекивања.
2. Фаза анализе: У овој фази, тим анализира прикупљене захтеве и на основу њих дефинише функционалне и нефункционалне захтеве. Тим истражује додатне информације и изворе, као и пословне процесе које софтвер треба да подржи. Ова фаза обухвата и детаљну спецификацију захтева.
3. Фаза пројектовања: У овој фази, пројектује се архитектура софтверског система. Одабирају се адекватни софтверски патерни и технологије, и креирају се дијаграми и спецификације. Овде се такође планирају активности и ресурси потребни за имплементацију.
4. Фаза имплементације: У овој фази, тим приступа имплементацији софтвера на основу пројектованих спецификација. Имплементација се обично врши по модулима или компонентама, примењујући одговарајуће програмске језике и технологије.
5. Фаза тестирања: У овој фази, софтвер се тестира и проверава да би се осигурала исправност и функционалност. Тестирање укључује различите нивое, као што су јединично тестирање, интеграционо тестирање и системско тестирање. Циљ је открити и исправити грешке и уверити се да софтвер испуњава све захтеве и очекивања стејкхолдера.

Ове фазе представљају основни оквир за развој со система. Упрошћена Ларманова метода је агилна и итеративно-инкрементална, што значи да се фазе понављају и усавршавају уз сваку итерацију, што доводи до континуираних побољшања и еволуције софтвера.

3.1. Прикупљање корисничких захтева

Почетна фаза у развоју било каквог система применом ове методе је прикупљање захтева од стране корисника. Ти захтеви представљају услове које систем треба да задовољи по завршетку израде. Ово је фаза на коју се ослањају све наредне фазе што је чини круцијалном и нетолерантном на грешке, јер сваки вид неправилности касније може експоненцијално да повећа трошак или трајање израде пројекта. Због тога је потребно издвојити што је могуће више времена и труда за комуникацију са клијентом како би се правилно разумеле његове потребе и захтеви.

3.1.1. Вербални опис

Потребно је направити софтверски систем који омогућава увид у најутицајније филмове модерног доба. Апликација треба да прикаже филмове, основе податке о датом ентитету, као и глумце и поставу иза камере. Основне податке о филму представљају његов опис, трајање, датум премијере, жанрови итд.

Систем треба и да има могућност приказа активности других корисника уз помоћ њихових рецензија и филмских листа.
Потребно је направити кориснике са различитим ролама у систему.
Прва је администратор који може да мења основне податке о филму.
Друга рола представља критичара који такође може и да пише рецензије и оцењује филмове.
Обичан корисник је трећи тип роле и он има могућност прегледа свог садржаја унутар апликације као што су описи филмова, рецензије стручњака, додавање филмова у своју личну листу жеља. Овим је омогућен одређен вид подсетника корисницима система, које филмове желе да гледају у будућности.

3.1.2. Случајеви коришћења

1. Преглед детаља филма
2. Преглед детаља филмског особља
3. Додавање филма у листу жеља
4. Уклањање филма из листе жеља
5. Додавање рецензије филма
6. Измена рецензије филма
7. Измена података о филму
8. Пријава на систем

Слика 2 Модел случајева коришћења

СК1: Случај коришћења – Преглед детаља филма

Назив СК
Преглед детаља филма

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен под својом шифром. Кориснику је приказана листа филмова.

Основни сценарио СК

1. Корисник **бира** филм који жели да му се прикаже. (АПУСО)
2. Корисник **позива** систем да му се прикаже изабрани филм. (АПСО)
3. Систем **тражи** филм по задатој вредности. (СО)
4. Систем **приказује** кориснику податке о филму и поруку „Систем је учитао филм“. (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе филм он приказује кориснику поруку: “Систем не може да нађе филм по задатој вредности”. (ИА)

СК2: Случај коришћења – Преглед детаља особља

Назив СК
Преглед детаља особља

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен под својом шифром. Кориснику је приказана листа особа које су радиле на филмовима.

Основни сценарио СК

1. Корисник **бира** особу који жели да му се прикаже. (АПУСО)
2. Корисник **позива** систем да му се прикаже изабрана особа. (АПСО)
3. Систем **тражи** особу по задатој вредности. (СО)
4. Систем **приказује** кориснику податке о особи и поруку „Систем је учитао особу“. (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе особу он приказује кориснику поруку: “Систем не може да нађе особу по задатој вредности”. (ИА)

СК3: Случај коришћења - Додавање у листу жеља

Назив СК
Додавање филма у листу жеља

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен под својом шифром. Систем приказује листу филмова.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује филмове. (АПУСО)
2. Корисник **позива** систем да нађе филмове по задатој вредности. (АПСО)
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** кориснику филмове и поруку: "Систем је нашао филмове по задатој вредности". (ИА)
5. Корисник **бира** филм који жели да дода у листу жеља. (АПУСО)
6. Корисник **уноси (додаје)** филм у листу жеља. (АПУСО)
7. Корисник **позива** систем да дода филм у листу жеља. (АПСО)
8. Систем **памти** податке о листи жеља. (СО)
9. Систем **приказује** кориснику запамћену листу жеља и поруку „Систем је запамтио листу жеља". (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује кориснику поруку: "Систем не може да нађе филмове по задатој вредности". Прекида се извршење сценарија. (ИА)

9.1 Уколико систем не може да запамти податке о листи жеља он приказује кориснику поруку "Систем не може да запамти листу жеља". (ИА)

СК4: Случај коришћења – Уклањање из листе жеља

Назив СК
Уклањање филма из листе жеља

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и кориснику је пријављен под својом шифром. Систем приказује листу филмова.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује филмове. (АПУСО)
2. Корисник **позива** систем да нађе филмове по задатој вредности. (АПСО)
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем приказује кориснику филмове и поруку: "Систем је нашао филмове по задатој вредности". (ИА)
5. Корисник **бира** филм који жели да уклони из листе жеља. (АПУСО)
6. Корисник **позива** систем да уклони филм из листе жеља. (АПСО)
7. Систем **уклања** филм из листе жеља. (СО)
8. Систем **приказује** кориснику поруку: "Систем је обрисао филм из листе жеља." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује кориснику поруку: "Систем не може да нађе филмове по задатој вредности". Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да обрише филм он приказује кориснику поруку "Систем не може да обрише филм из листе жеља". (ИА)

СК5: Случај коришћења – Додавање рецензије

Назив СК
Додавање рецензије филма

Актори СК
Критичар

Учесници СК
Критичар и систем (програм)

Предуслов: Систем је укључен и критичар је пријављен под својом шифром. Систем приказује листу филмова.

Основни сценарио СК

1. Критичар **уноси** вредност по којој претражује филмове. (АПУСО)
2. Критичар **позива** систем да нађе филмове по задатој вредности. (АПСО)
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем приказује критичару филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Критичар **бира** филм за који жели да напише рецензију. (АПУСО)
6. Критичар **уноси** потребне податке о рецензији филма. (АПУСО)
7. Критичар **позива** систем да запамти податке о рецензији филма. (АПСО)
8. Систем **памти** податке о рецензији филма. (СО)
9. Систем **приказује** критичару запамћену рецензију филма и поруку „Систем је запамтио рецензију филма“. (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује критичару поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

9.1 Уколико систем не може да запамти податке о рецензији филма он приказује критичару поруку “Систем не може да запамти рецензију филма”. (ИА)

СК6: Случај коришћења – Измена рецензије

Назив СК
Промена података рецензије филма

Актори СК
Критичар

Учесници СК
Критичар и систем (програм)

Предуслов: Систем је укључен и критичар је пријављен под својом шифром. Систем приказује листу филмова.

Основни сценарио СК

1. Критичар **уноси** вредност по којој претражује филмове. (АПУСО)
2. Критичар **позива** систем да нађе филмове по задатој вредности. (АПСО)
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** критичару филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Критичар **бира** филм за који жели да промени податке о рецензији филма. (АПУСО)
6. Критичар **позива** систем да учита рецензију изабраног филма. (АПСО)
7. Систем **учитава** рецензију изабраног филма. (СО)
8. Систем **приказује** критичару рецензију филма и поруку: "Систем је учитао рецензију филма". (ИА)
9. Критичар **уноси (мења)** податке о рецензији филма. (АПУСО)
10. Критичар **контролише** да ли је коректно унео податке о рецензији филма. (АНСО)
11. Критичар **позива** систем да запамти податке о рецензији филма. (АПСО)
12. Систем **памти** податке о рецензији филма. (СО)
13. Систем **приказује** критичару запамћену рецензију филма и поруку „Систем је запамтио рецензију филма“. (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује критичару поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

13.1 Уколико систем не може да запамти податке о рецензији филма он приказује критичару поруку “Систем не може да запамти рецензију филма”. (ИА)

СК7: Случај коришћења – Измена филма

Назив СК
Промена података филма

Актори СК
Администратор

Учесници СК
Администратор и систем (програм)

Предуслов: Систем је укључен и администратор је пријављен под својом шифром. Систем приказује листу филмова.

Основни сценарио СК

1. Администратор **уноси** вредност по којој претражује филмове. (АПУСО)
2. Администратор **позива** систем да нађе филмове по задатој вредности. (АПСО)
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем приказује администратору филмове и „Систем је нашао филмове по задатој вредности“. (ИА)
5. Администратор **бира** филм за који жели да промени податке. (АПУСО)
6. Администратор **позива** систем да учита изабрани филм. (АПСО)
7. Систем **учитава** изабрани филм. (СО)
8. Систем **приказује** администратору изабрани филм и поруку: "Систем је учитао филм". (ИА)
9. Администратор **уноси (мења)** податке о филму. (АПУСО)
10. Администратор **контролише** да ли је коректно унео податке филма. (АНСО)
11. Администратор **позива** систем да запамти податке о филму. (АПСО)
12. Систем **памти** податке о филму. (СО)
13. Систем **приказује** администратору запамћени филм и „Систем је запамтио филм“. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да нађе филмове он приказује администратору поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

13.1 Уколико систем не може да запамти податке о филму он приказује администратору поруку “Систем не може да запамти филм”. (ИА)

СК8: Случај коришћења – Пријава корисника на систем

Назив СК
Пријава корисника на систем

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник није пријављен под својом шифром.

Основни сценарио СК

1. Корисник **уноси** своје податке (корисничко име и лозинку). (АПУСО)
2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)
3. Корисник **позива** систем да га пријави на систем. (АПСО)
4. Систем **пријављује** корисника. (СО)
5. Систем приказује кориснику корисника и поруку „Систем је пријавио корисника на систем“. (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да пријави корисника он приказује кориснику поруку: “Систем не може да пријави корисника”. (ИА)

3.2. Анализа

Фаза анализе се односи на описивање логичке структуре и понашања софтверског система, односно пословне логике која стоји иза софтверског система. [2] Функционалност се дефинише користећи системске дијаграме секвенци и уговоре о системским операцијама, док се структура дефинише користећи концептуални и релациони модел. [2]

3.2.1. Понашање софтверског система - Системски дијаграми секвенци

Коришћењем дијаграма секвенци представља се интеракција између ентитета при одређеном секвенцијалном редоследу. Дијаграм секвенци је сачињен од ентитета које учествују у интеракцији, временских линија и порука које се размењују међу ентитетима учесницима. [2]

ДС1: Дијаграм секвенце случаја коришћења – Преглед детаља филма

1. Форма позива систем да учита листу филмова. (АПСО)
2. Систем приказује форми листу филмова. (ИА)
3. Корисник **позива** систем да му се прикаже изабрани филм. (АПСО)
4. Систем приказује кориснику податке о филму и „Систем је учитао филм“. (ИА)

Дијаграм 1 Преглед детаља филма

Алтернативна сценарија

4.1 Уколико систем не може да нађе филм он приказује кориснику поруку: “Систем не може да нађе филм по задатој вредности”. (ИА)

Дијаграм 2 Преглед детаља филма, алтернативни сценарио 1

Са наведених дијаграма секвенци уочавају се 2 системске операције које треба пројектовати:

1. signal **getAll**(List<Movie>)
2. signal **getById**(Movie)

ДС2: Дијаграм секвенце случаја коришћења – Преглед детаља особља

1. Форма позива систем да учита листу особља. (АПСО)
2. Систем приказује форми листу особља. (ИА)
3. Корисник **позива** систем да му се прикаже изабрана особа. (АПСО)
4. Систем приказује кориснику податке о особи и „Систем је учитао особу“. (ИА)

Дијаграм 3 Преглед детаља особља

Алтернативна сценарија

4.1 Уколико систем не може да нађе особу он приказује кориснику поруку: “Систем не може да нађе особу по задатој вредности”. (ИА)

Дијаграм 4 Преглед детаља особља, алтернативни сценарио 1

Са наведених дијаграма секвенци уочавају се 2 системске операције које треба пројектовати:

1. signal **getAll**(List<Person>)
2. signal **getById**(Person)

ДС3: Дијаграм секвенце случаја коришћења – Додавање у листу жеља

1. Форма позива систем да учита листу филмова. (АПСО)
2. Систем приказује форми листу филмова. (ИА)
3. Корисник **позива** систем да нађе филмове по задатој вредности. (АПСО)
4. Систем приказује кориснику филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Корисник **позива** систем да запамти податке о листи жеља. (АПСО)
6. Систем **приказује** кориснику запамћену листу жеља и „Систем је запамтио листу жеља“. (ИА)

Дијаграм 5 Додавање филма у листу жеља

Алтернативни сценарији

4.1 Уколико систем не може да нађе филмове он приказује кориснику поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Дијаграм 6 Додавање у листу жеља, алтернативни сценарио 1

6.1 Уколико систем не може да запамти податке о листи жеља он приказује кориснику поруку “Систем не може да запамти листу жеља”. (ИА)

Дијаграм 7 Додавање у листу жеља, алтернативни сценарио 2

Са наведених дијаграма секвенци уочавају се 4 системске операције које треба пројектовати:

1. signal **getAll**(List<Movie>)
2. signal **getMoviesByTitle**(title, List<Movie>)
3. signal **getById**(Movie)
4. signal **addToWatchlist**(Movie)

ДС4: Дијаграм секвенце случаја коришћења – Уклањање из листе жеља

1. Форма позива систем да учита листу филмова. (АПСО)
2. Систем приказује форми листу филмова. (ИА)
3. Корисник **позива** систем да нађе филмове по задатој вредности. (АПСО)
4. Систем приказује кориснику филмове и поруку: “Систем је нашао филмове по задатој вредности”. (ИА)
5. Корисник **позива** систем да уклони филм из листе жеља. (АПСО)
6. Систем **приказује** кориснику поруку: “Систем је обрисао филм из листе жеља.” (ИА)

Дијаграм 8 Уклањање филма из листе жеља

Алтернативни сценарији

4.1. Уколико систем не може да нађе филмове он приказује кориснику поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Дијаграм 9 Уклањање из листе жеља, алтернативни сценарио 1

6.1 Уколико систем не може да обрише филм из листе жеља он приказује кориснику поруку “Систем не може да обрише филм из листе жеља”. (ИА)

Дијаграм 10 Уклањање из листе жеља, алтернативни сценарио 2

Са наведених дијаграма секвенци уочавају се 4 системске операције које треба пројектовати:

1. signal **getAll**(List<Movie>)
2. signal **getMoviesByTitle**(title, List<Movie>)
3. signal **getById**(Movie)
4. signal **removeFromWatchlist**(Movie)

ДС5: Дијаграм секвенце случаја коришћења – Додавање рецензије

1. Форма **позива** систем да учита листу филмова. (АПСО)
2. Систем **приказује** форми листу филмова. (ИА)
3. Критичар **позива** систем да нађе филмове по задатој вредности. (АПСО)
4. Систем **приказује** критичару филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Критичар **позива** систем да запамти податке о рецензији филма. (АПСО)
6. Систем **приказује** критичару запамћену рецензију филма и „Систем је запамтио рецензију филма“. (ИА)

Дијаграм 11 Додавање рецензије

Алтернативни сценарији

4.1. Уколико систем не може да нађе филмове он приказује критичару поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Дијаграм 12 Додавање рецензије, алтернативни сценарио 1

6.1 Уколико систем не може да запамти податке о рецензији филма он приказује критичару поруку “Систем не може да запамти рецензију филма”. (ИА)

Дијаграм 13 Додавање рецензије, алтернативни сценарио 2

Са наведених дијаграма секвенци уочавају се 4 системске операције које треба пројектовати:

1. signal **getAll**(List<Movie>)
2. signal **getMoviesByTitle**(title, List<Movie>)
3. signal **getById**(Movie)
4. signal **createReview**(Review)

ДС6: Дијаграм секвенце случаја коришћења – Измена рецензије

1. Форма позива систем да учита листу филмова. (АПСО)
2. Систем приказује форми листу филмова. (ИА)
3. Критичар **позива** систем да нађе филмове по задатој вредности. (АПСО)
4. Систем **приказује** критичару филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Критичар **позива** систем да учита рецензију изабраног филма. (АПСО)
6. Систем **приказује** критичару рецензију филма и поруку: "Систем је учитао рецензију филма". (ИА)
7. Критичар **позива** систем да запамти податке о рецензији филма. (АПСО)
8. Систем **приказује** критичару запамћену рецензију филма и поруку „Систем је запамтио рецензију филма“. (ИА)

Дијаграм 14 Измена рецензије

Алтернативни сценарији

4.1 Уколико систем не може да нађе филм он приказује критичару поруку: “Систем не може да нађе филм по задатој вредности”. Прекида се извршење сценарија. (ИА)

Дијаграм 15 Измена рецензије, алтернативни сценарио 1

8.1 Уколико систем не може да запамти податке о рецензији филма он приказује критичару поруку “Систем не може да запамти рецензију филма”. (ИА)

Дијаграм 16 Измена рецензије, алтернативни сценарио 2

Са наведених дијаграма секвенци уочавају се 4 системске операције које треба пројектовати:

1. signal **getAll**(List<Movie>)
2. signal **getMoviesByTitle**(title, List<Movie>)
3. signal **getById**(Movie)
4. signal **updateReview**(Review)

ДС7: Дијаграм секвенце случаја коришћења – Измена филма

1. Форма позива систем да учита листу филмова. (АПСО)
2. Систем приказује форми листу филмова. (ИА)
3. Администратор **позива** систем да нађе филмове по задатој вредности. (АПСО)
4. Систем **приказује** администратору филмове и „Систем је нашао филмове по задатој вредности“. (ИА)
5. Администратор **позива** систем да учита изабрани филм. (АПСО)
6. Систем **приказује** администратору изабрани филм и поруку: "Систем је учитао филм". (ИА)
7. Администратор **позива** систем да запамти податке о филму. (АПСО)
8. Систем **приказује** администратору запамћени филм и „Систем је запамтио филм“. (ИА)

Дијаграм 17 Измена филма

Алтернативни сценарији

4.1 Уколико систем не може да нађе филмове он приказује администратору поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Дијаграм 18 Измена филма, алтернативни сценарио 1

8.1 Уколико систем не може да запамти податке о филму он приказује администратору поруку “Систем не може да запамти филм”. (ИА)

Дијаграм 19 Измена филма, алтернативни сценарио 2

Са наведених дијаграма секвенци уочавају се 4 системске операције које треба пројектовати:

1. signal **getAll**(List<Movie>)
2. signal **getMoviesByTitle**(title, List<Movie>)
3. signal **getById**(Movie)
4. signal **updateMovie**(Movie)

ДС8: Дијаграм секвенце случаја коришћења – Пријава корисника на систем

1. Корисник **позива** систем да га пријави на систем. (АПСО)
2. Систем приказује кориснику корисника и поруку „Систем је пријавио корисника на систем“. (ИА)

Дијаграм 20 Пријава корисника на систем

Алтернативна сценарија

2.1 Уколико систем не може да пријави корисника он приказује кориснику поруку: “Систем не може да пријави корисника”. (ИА)

Дијаграм 21 Пријава корисника на систем, алтернативни сценарио 1

Са наведених дијаграма секвенци уочава се 1 системска операција коју треба пројектовати:

1. signal **login**(User)

На основу свих наведених системских дијаграма секвенци може се уочити следећих 10 системских операција:

1. signal **getAll**(List<Movie>)
2. signal **getMoviesByTitle**(title, List<Movie>)
3. signal **getById**(Movie)
4. signal **getAll**(List<Person>)
5. signal **getById**(Person)
6. signal **addToWatchlist**(Movie)
7. signal **removeFromWatchlist**(Movie)
8. signal **createReview**(Review)
9. signal **updateReview**(Review)
10. signal **updateMovie**(Movie)
11. signal **login**(User)

3.2.2. Понашање софтверског система – уговори

За сваки елемент листе системских операција се креира уговор којим се представља опис њеног функционисања. Системска операција би требало да има свој потпис тј. назив методе, као и улазне и излазне параметре. Уговор садржи операцију са њеним потписом и параметрима, повезаност са случајевима коришћења, предуслове и постуслове.

Уговор УГ1: getAll(List<Movie>): signal;
Веза са СК: СК1, СК3, СК4, СК5, СК6, СК7
Предуслови: /
Постуслови: /

Уговор УГ2: getMoviesByTitle(title, List<Movie>): signal;
Веза са СК: СК3, СК4, СК5, СК6, СК7
Предуслови: /
Постуслови: /

Уговор УГ3: getById(Movie): signal;
Веза са СК: СК1, СК3, СК4, СК5, СК6, СК7
Предуслови: /
Постуслови: /

Уговор УГ4: getAll(List<Person>): signal;
Веза са СК: СК2
Предуслови: /
Постуслови: /

Уговор УГ5: getById(Person): signal;
Веза са СК: СК2
Предуслови: /
Постуслови: /

Уговор УГ6: addToWatchlist(Movie): signal;
Веза са СК: СК3
Предуслови: Структурна и вредносна ограничења над објектом листе жеља морају бити задовољена и изабрани филм се већ не налази у датој листи жеља.
Постуслови: Листа жеља садржи изабрани филм.

Уговор УГ7: removeFromWatchlist(Movie): signal;
Веза са СК: СК4
Предуслови: Структурна ограничења над објектом листе жеља морају бити задовољена као и то да се дати филм већ налази унутар листе жеља корисника.
Постуслови: Листа жеља више не садржи изабрани филм.

Уговор УГ8: createReview(Review): signal;
Веза са СК: СК5
Предуслови: Вредносна и структурна ограничења над објектом рецензија морају бити

задовољена, као и да дати критичар није већ написао рецензију за дати филм.
Постуслови: Креирана је рецензија.

Уговор УГ9: updateReview(Review): signal;
Веза са СК: СК6
Предуслови: Вредносна и структурна ограничења над рецензијом морају бити задовољена, као и да критичар није написао рецензију за дати филм.
Постуслови: Измењена је рецензија.

Уговор УГ10: updateMovie(Movie): signal;
Веза са СК: СК7
Предуслови: Вредносна и структурна ограничења над филмом морају бити задовољена.
Постуслови: Измењени су подаци изабраног филма.

Уговор УГ11: login(User): signal;
Веза са СК: СК8
Предуслови: /
Постуслови: /

3.2.3. Структура софтверског система

Структура софтверског система је основа за даље пројектовање базе података. Може се представити уз помоћу концептуалног (доменског) модела и као и уз помоћ релационог модела. [2]

3.2.3.1. Концептуални (доменски) модел

Концептуални модел је сачињен од класа и међусобних веза при чему он представља приказ структуре ентитета унутар софтверског система. При одређивању класа и веза, приказују се и атрибути који имају адекватан тип података. [2]

Слика 3 Концептуални (доменски) модел студијског примера

Као резултат анализе сценарија СК и прављења концептуалног модела добија се логичка структура и понашање софтверског система:

Слика 4 Логичка структура и понашање софтверског система

3.2.3.2. Релациони модел

Сама шема релационе базе података се прави директно наспрам постављеног релационог модела, који је настао трансформацијом из концептуалног модела.

Релациони модел се представља уз помоћ табела (релација), примарних и спољних кључева, атрибута и њихових типова, као и свих ограничења.

Movie (**movie_id**, title, release_date, overview, runtime, budget, revenue, poster_path, backdrop_path, homepage, imdb, original_language, tagline)

User (**user_id**, firstname, lastname, username, password, email, role)

Review (**review_id**, content, rating, timestamp, *user_id, movie_id*)

WatchList (**watch_list_id**, *user_id*)

Movie_WatchList (***movie_id, watch_list_id***)

Genre (**genre_id**, name)

Movie_Genre (***movie_id, genre_id***)

Person (**person_id**, name, birthday, biography, imdb, place_of_birth, profile_path, gender)

Actor (**actor_id, *movie_id***, character, *person_id*)

CrewMember (**crew_member_id, *movie_id***, job, department, *person_id*)

Табела Movie		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	**INSERT** / **UPDATE** CASCADES Actor, Crew_member, Review, Movie_WatchList, Movie_Genre **DELETE** RESTRICTED Actor, Crew_member, Review, Movie_WatchList, Movie_Genre
	movie_id	Long	not null AND > 0			
	title	String	not null			
	release_date	Date	not null			
	overview	String	not null			
	runtime	Integer	not null AND > 0			
	budget	Integer	not null AND > 0			
	revenue	Integer	not null AND > 0			
	poster_path	String	not null			
	backdrop_path	String	not null			
	homepage	String				
	imdb	String				
	original_language	String				
	tagline	String	not null			

Табела 1 Movie

Табела User		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	**INSERT** / **UPDATE** CASCADES Review, WatchList **DELETE** RESTRICTED Review, WatchList
	user_id	Long	not null AND > 0			
	firstname	String	not null			
	lastname	String	not null			
	username	String	not null			
	password	String	not null AND > 0			
	email	String	not null AND > 0			
	role	String	not null AND IN (‘ADMIN’, ‘CRITIC’, ‘USER’)			

Табела 2 User

Табела Review		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	**INSERT** RESTRICTED User, Movie **UPDATE** RESTRICTED User, Movie **DELETE** /
	review_id	Long	not null AND > 0			
	content	String	not null			
	rating	Integer	not null AND BETWEEN 1 AND 10			
	timestamp	Date	not null			
	user_id	Long	not null AND > 0			
	movie_id	Long	not null AND > 0			

Табела 3 Review

<table>
<tr><td colspan="2">Табела WatchList</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="3">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="3">INSERT
RESTRICTED
User

UPDATE
CASCADES
Movie_WatchList
RESTRICTED
User

DELETE
RESTRICTED
Movie_WatchList</td></tr>
<tr><td>watch_list_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>user_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
</table>

Табела 4 WatchList

<table>
<tr><td colspan="2">Табела Movie_WatchList</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="3">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="3">INSERT
RESTRICTED
Movie, WatchList

UPDATE
RESTRICTED
Movie, WatchList

DELETE
/</td></tr>
<tr><td>movie_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>watch_list_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
</table>

Табела 5 Movie_WatchList

<table>
<tr><td colspan="2">Табела Genre</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="3">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="3">INSERT
/

UPDATE
CASCADES
Movie_Genre

DELETE
RESTRICTED
Movie_Genre</td></tr>
<tr><td>genre_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>name</td><td>String</td><td>not null</td><td></td><td></td></tr>
</table>

Табела 6 Genre

<table>
<tr><td colspan="2">Табела Movie_Genre</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="3">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="3">INSERT
RESTRICTED
Movie, Genre

UPDATE
RESTRICTED
Movie, Genre

DELETE
/</td></tr>
<tr><td>genre_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>movie_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
</table>

Табела 7 Movie_Genre

<table>
<tr><td colspan="2">Табела Person</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="9">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="9">INSERT
/
UPDATE
CASCADES
Actor,
Crew_member
DELETE
RESTRICTED
Actor,
Crew_member</td></tr>
<tr><td>person_id</td><td>Long</td><td>not null
AND > 0</td><td></td><td></td></tr>
<tr><td>name</td><td>String</td><td>not null</td><td></td><td></td></tr>
<tr><td>birthday</td><td>Date</td><td></td><td></td><td></td></tr>
<tr><td>biography</td><td>String</td><td>not null</td><td></td><td></td></tr>
<tr><td>imdb</td><td>String</td><td></td><td></td><td></td></tr>
<tr><td>place_of_birth</td><td>String</td><td></td><td></td><td></td></tr>
<tr><td>profile_path</td><td>String</td><td></td><td></td><td></td></tr>
<tr><td>gender</td><td>String</td><td>not null
AND
IN (‘MALE,
‘FEMALE’,
‘OTHER’)</td><td></td><td></td></tr>
</table>

Табела 8 Person

<table>
<tr><td colspan="2">Табела Actor</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="5">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="5">INSERT
RESTRICTED
Person, Movie

UPDATE
RESTRICTED
Person, Movie

DELETE
/</td></tr>
<tr><td>actor_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>movie_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>character</td><td>String</td><td>not null</td><td></td><td></td></tr>
<tr><td>person_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
</table>

Табела 9 Actor

<table>
<tr><td colspan="2">Табела CrewMember</td><td colspan="2">Просто вредносно ограничење</td><td colspan="2">Сложено вредносно ограничење</td><td>Структурно ограничење</td></tr>
<tr><td rowspan="6">Атрибути</td><td>Име</td><td>Тип атрибута</td><td>Вредност атрибута</td><td>Међузав. атрибута једне табеле</td><td>Међузав. атрибута више табела</td><td rowspan="6">INSERT
RESTRICTED
Person, Movie

UPDATE
RESTRICTED
Person, Movie

DELETE
/</td></tr>
<tr><td>crew_member_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>movie_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
<tr><td>job</td><td>String</td><td>not null</td><td></td><td></td></tr>
<tr><td>department</td><td>String</td><td>not null</td><td></td><td></td></tr>
<tr><td>user_id</td><td>Long</td><td>not null AND > 0</td><td></td><td></td></tr>
</table>

Табела 10 CrewMember

3.3. Пројектовање

Фаза пројектовања дефинише структуру и понашање софтверског система па самим тим и његову архитектуру. [2] У овом завршном раду користи се тронивојска архитектура софтверског система у којој се систем састоји из 3 нивоа, што је и приказано на наредној слици: [2]

Слика 5 Тронивојска архитектура [2]

Архитектура је подељена на три целине што представља стандардну тронивојску архитектуру која је приказана на следећој слици:

Слика 6 Архитектура софтверског система

Први слој је презентациони и он представља кориснички интерфејс са којим актор директно интерагује. Састоји се од екранских форми, као и HTTP клијента који се повезује на други слој. Други слој је слој свих сервиса и пословне логике апликације. Трећи слој је слој базе података који служи да перзистира податке у меморију рачунара.

Други слој се може још додатно рашчланити на дате целине:

1. Ауторизациони *middleware* (Компонента који проверава приступ на основу креденцијала и рола)
2. Валидатори (Компонента који проверава примљене податке)
3. Контролери (Диспечер примљених захтева од стране корисника)
4. Мапери (Конвертер објеката за трансфер података у ентитете система)
5. Сервиси (Скуп пословних правила, апликационе логике и филтрирања)
6. Репозиторијуми (Перзистентни оквир за директно мапирање објеката програмског језика и табела базе података и управљање трансакцијама)

3.3.1. Пројектовање корисничког интерфејса

Кориснички интерфејс представља компоненту софтверског система који је директно задужен за интеракцију са корисником путем приказа садржаја и ослушкивања корисникових захтева. [2] Сачињен је од екранске форме и контролера корисничког интерфејса који је у веб апликацији имплементиран као HTTP клијент.

Слика 7 Структура корисничког интерфејса [2]

Пројектовање екранских форми

Кориснички интерфејс је дефинисан преко скупа екранских форми. Пројектује се на такав начин да сваки случај коришћења има екранску форму са којом се може повезати. Серверска страна нема графички кориснички интерфејс. [2]

На клијентској страни, по покретању програма, потребно је да корисник изврши пријаву на систем коришћењем својих креденцијала. Наравно, уколико не поседује налог може га креирати избором опције за регистрацију.

Слика 8 Форма за пријаву корисника

Слика 9 Форма за регистрацију корисника

Након успешне пријаве на систем кориснику се приказује почетна страна која је уједно и приказ листе популарних и култних филмова модерног доба:

Слика 10 Почетна страна

СК1: Случај коришћења – Преглед детаља филма

Назив СК
Преглед детаља филма

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен под својом шифром. Кориснику је приказана листа филмова.

Слика 11 Страна са листом филмова

Основни сценарио СК

1. Корисник **бира** филм који жели да му се прикаже. (АПУСО)
2. Корисник **позива** систем да му се прикаже изабрани филм. (АПСО)
   Опис акције: Кликом на постер филма која представља изабрани филм позива се системска операција getById(Movie) која учитава податке о филму
3. Систем **тражи** филм по задатој вредности. (СО)
4. Систем приказује кориснику податке о филму и поруку „Систем је учитао филм“. (ИА)

Слика 12 Форма са детаљима филма

Алтернативна сценарија

4.1 Уколико систем не може да нађе филм он приказује кориснику поруку: “Систем не може да нађе филм по задатој вредности”. (ИА)

Слика 13 Проналажење филма, алтернативни сценарио 1

СК2: Случај коришћења – Преглед детаља особља

Назив СК
Преглед детаља особља

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен под својом шифром. Кориснику је приказана листа особа које су радиле на филмовима.

Слика 14 Форма са листом особља

Основни сценарио СК

1. Корисник **бира** особу који жели да му се прикаже. (АПУСО)
2. Корисник **позива** систем да му се прикаже изабрана особа. (АПСО)
   Опис акције: Кликом на слику особе која представља одабрану особу позива се системска операција getById(Person) која учитава податке о особи.
3. Систем **тражи** особу по задатој вредности. (СО)
4. Систем приказује кориснику податке о особи и поруку „Систем је учитао особу“. (ИА)

Слика 15 Форма са детаљима особе

Алтернативна сценарија

4.1 Уколико систем не може да нађе особу он приказује кориснику поруку: “Систем не може да нађе особу по задатој вредности”. (ИА)

Слика 16 Проналазак особе, алтернативни сценарио 1

СК3: Случај коришћења - Додавање у листу жеља

Назив СК
Додавање филма у листу жеља

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен под својом шифром. Систем приказује листу филмова.

Слика 17 Форма са листом филмова

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује филмове. (АПУСО)
2. Корисник **позива** систем да нађе филмове по задатој вредности. (АПСО)
   Опис акције: Кликом на дугме за мењање странице и/или попуњавањем поља за претрагу на основу наслова филма позива се системска операција getMoviesByTitle(title, List<Movie>) која учитава филмове са задатим вредностима претраге.
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** кориснику филмове и поруку: "Систем је нашао филмове по задатој вредности". (ИА)
5. Корисник **бира** филм који жели да дода у листу жеља. (АПУСО)

Слика 18 Форма са детаљима филма

6. Корисник **уноси (додаје)** филм у листу жеља. (АПУСО)

Слика 19 Додавање у листу жеља

7. Корисник **позива** систем да дода филм у листу жеља. (АПСО)
   Опис акције: Кликом на дугме за додавање филма у листу жеља позива се системска операција addToWatchList(Movie) која додаје изабрани филм у личну листу жеља.
8. Систем **памти** податке о листи жеља. (СО)
9. Систем **приказује** кориснику запамћену листу жеља и „Систем је запамтио листу жеља“. (ИА)

Слика 20 Форма корисника са листом жеља

Алтернативна сценарија

4.1. Уколико систем не може да нађе филмове он приказује кориснику поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Слика 21 Додавање у листу жеља, алтернативни сценарио 1

9.1 Уколико систем не може да запамти податке о листи жеља он приказује кориснику поруку “Систем не може да запамти листу жеља”. (ИА)

Слика 22 Додавање у листу жеља, алтернативни сценарио 2

СК4: Случај коришћења – Уклањање из листе жеља

Назив СК
Уклањање филма из листе жеља

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и кориснику је пријављен под својом шифром. Систем приказује листу филмова.

Слика 23 Форма са листом филмова

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује филмове. (АПУСО)
2. Корисник **позива** систем да нађе филмове по задатој вредности. (АПСО)
   Опис акције: Кликом на дугме за мењање странице и/или попуњавањем поља за претрагу на основу наслова филма позива се системска операција getMoviesByTitle(title, List<Movie>) која учитава филмове са задатим вредностима претраге.
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** кориснику филмове и поруку: "Систем је нашао филмове по задатој вредности". (ИА)
5. Корисник **бира** филм који жели да уклони из листе жеља. (АПУСО)

Слика 24 Форма са детаљима филма

6. Корисник **позива** систем да уклони филм из листе жеља. (АПСО)
   Опис акције: Кликом на дугме за брисање филма из листе жеља позива се системска операција removeFromWatchList(Movie) која брише изабрани филм из личне листе жеља.

Слика 25 Уклањање из листе жеља

7. Систем **уклања** филм из листе жеља. (СО)
8. Систем **приказује** кориснику поруку: “Систем је обрисао филм из листе жеља.” (ИА)

Слика 26 Форма корисника са листом жеља

Алтернативна сценарија

4.1. Уколико систем не може да нађе филмове он приказује кориснику поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Слика 27 Уклањање из листе жеља, алтернативни сценарио 1

8.1 Уколико систем не може да обрише филм он приказује кориснику поруку “Систем не може да обрише филм из листе жеља”. (ИА)

Слика 28 Уклањање из листе жеља, алтернативни сценарио 2

СК5: Случај коришћења – Додавање рецензије

Назив СК
Додавање рецензије филма

Актори СК
Критичар

Учесници СК
Критичар и систем (програм)

Предуслов: Систем је укључен и критичар је пријављен под својом шифром. Систем приказује листу филмова.

Слика 29 Форма са листом филмова

Основни сценарио СК

1. Критичар **уноси** вредност по којој претражује филмове. (АПУСО)
2. Критичар **позива** систем да нађе филмове по задатој вредности. (АПСО)
   Опис акције: Кликом на дугме за мењање странице и/или попуњавањем поља за претрагу на основу наслова филма позива се системска операција getMoviesByTitle(title, List<Movie>) која учитава филмове са задатим вредностима претраге.
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** критичару филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Критичар **бира** филм за који жели да напише рецензију. (АПУСО)

Слика 30 Форма са детаљима филма

6. Критичар **уноси** потребне податке о рецензији филма. (АПУСО)

Слика 31 Додавање рецензије

Слика 32 Унос података рецензије

7. Критичар **позива** систем да запамти податке о рецензији филма. (АПСО)
   Опис акције: Кликом на дугме за додавање рецензије, следи попуњавање детаља рецензије након чега се кликом на дугме за потврду додате рецензије позива се системска операција createReview(Review) која памти нову рецензију за дати филм.
8. Систем **памти** податке о рецензији филма. (СО)
9. Систем **приказује** критичару запамћену рецензију филма и „Систем је запамтио рецензију филма“. (ИА)

Слика 33 Додата рецензија

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује кориснику поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Слика 34 Додавање рецензије, алтернативни сценарио 1

9.1 Уколико систем не може да запамти податке о рецензији филма он приказује кориснику поруку “Систем не може да запамти рецензији филма”. (ИА)

Слика 35 Додавање рецензије, алтернативни сценарио 2

СК6: Случај коришћења – Измена рецензије

Назив СК
Промена података рецензије филма

Актори СК
Критичар

Учесници СК
Критичар и систем (програм)

Предуслов: Систем је укључен и критичар је пријављен под својом шифром. Систем приказује листу филмова.

Слика 36 Форма са листом филмова

Основни сценарио СК

1. Критичар **уноси** вредност по којој претражује филмове. (АПУСО)
2. Критичар **позива** систем да нађе филмове по задатој вредности. (АПСО)
   Опис акције: Кликом на дугме за мењање странице и/или попуњавањем поља за претрагу на основу наслова филма позива се системска операција getMoviesByTitle(title, List<Movie>) која учитава филмове са задатим вредностима претраге.
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** критичару филмове и поруку „Систем је нашао филмове по задатој вредности“. (ИА)
5. Критичар **бира** филм за који жели да промени податке о рецензији филма. (АПУСО)

Слика 37 Форма са детаљима филма

6. Критичар **позива** систем да учита рецензију изабраног филма. (АПСО)
7. Систем **учитава** рецензију изабраног филма. (СО)
8. Систем **приказује** критичару рецензију филма и поруку: "Систем је учитао рецензију филма". (ИА)

Слика 38 Измена рецензије

9. Критичар **уноси (мења)** податке о рецензији филма. (АПУСО)
10. Критичар **контролише** да ли је коректно унео податке о рецензији филма. (АНСО)

Слика 39 Унос нових података рецензије

11. Критичар **позива** систем да запамти податке о рецензији филма. (АПСО)
    Опис акције: Кликом на дугме за ажурирање рецензије филма, попуњавају се нови подаци о рецензији након чега се кликом потврде позива системска операција updateReview(Review) која памти нове ажуриране податке о рецензији филма.
12. Систем **памти** податке о рецензији филма. (СО)
13. Систем **приказује** критичару запамћену рецензију филма и „Систем је запамтио рецензију филма“. (ИА)

Слика 40 Измењена рецензија

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује критичару поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Слика 41 Измена рецензије, алтернативни сценарио 1

13.1 Уколико систем не може да запамти податке о рецензији филма он приказује критичару поруку “Систем не може да запамти рецензију филма”. (ИА)

Слика 42 Измена рецензије, алтернативни сценарио 2

СК7: Случај коришћења – Измена филма

Назив СК
Промена података филма

Актори СК
Администратор

Учесници СК
Администратор и систем (програм)

Предуслов: Систем је укључен и критичар је пријављен под својом шифром. Систем приказује листу филмова.

Слика 43 Форма са листом филмова

Основни сценарио СК

1. Администратор **уноси** вредност по којој претражује филмове. (АПУСО)
2. Администратор **позива** систем да нађе филмове по задатој вредности. (АПСО)
   Опис акције: Кликом на дугме за мењање странице и/или попуњавањем поља за претрагу на основу наслова филма позива се системска операција getMoviesByTitle(title, List<Movie>) која учитава филмове са задатим вредностима претраге.
3. Систем **тражи** филмове по задатој вредности. (СО)
4. Систем **приказује** администратору филмове и „Систем је нашао филмове по задатој вредности“. (ИА)
5. Администратор **бира** филм за који жели да промени податке. (АПУСО)
6. Администратор **позива** систем да учита изабрани филм. (АПСО)
7. Систем **учитава** изабрани филм. (СО)
8. Систем **приказује** администратору изабрани филм и поруку: "Систем је учитао филм". (ИА)

Слика 44 Форма са детаљима филма

9. Администратор **уноси (мења)** податке о филму. (АПУСО)

Слика 45 Измена филма

10. Администратор **контролише** да ли је коректно унео податке филма. (АНСО)

Слика 46 Провера унесених података филма

11. Администратор **позива** систем да запамти податке о филму. (АПСО)
    Опис акције: Кликом на дугме за ажурирање филма, уносе се нови подаци након чега се кликом потврђују измене и позива се системска операција updateMovie(Movie) која мења податке филма.
12. Систем **памти** податке о филму. (СО)
13. Систем **приказује** администратору запамћени филм и „Систем је запамтио филм“. (ИА)

Слика 47 Измењени подаци филма

Алтернативна сценарија

4.1 Уколико систем не може да нађе филмове он приказује администратору поруку: “Систем не може да нађе филмове по задатој вредности”. Прекида се извршење сценарија. (ИА)

Слика 48 Измена података филма, алтернативни сценарио 1

13.1. Уколико систем не може да запамти податке о филму он приказује администратору поруку “Систем не може да запамти филм”. (ИА)

Слика 49 Измена података филма, алтернативни сценарио 1

СК8: Случај коришћења – Пријава корисника на систем

Назив СК
Пријава корисника на систем

Актори СК
Корисник

Учесници СК
Корисник и систем (програм)

Предуслов: Систем је укључен и корисник није пријављен под својом шифром и кориснички интерфејс је на форми за пријаву.

Слика 50 Форма за пријаву корисника

Основни сценарио СК

1. Корисник **уноси** своје податке (корисничко име и лозинку). (АПУСО)
2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

Слика 51 Пријава корисника

3. Корисник **позива** систем да га пријави на систем. (АПСО)
   Опис акције: Кликом на дугме пријаве позива се системска операција login(User) где систем упореди унете креденцијале и регистроване кориснике.

4. Систем **пријављује** корисника. (СО)
5. Систем **приказује** кориснику корисника и поруку „Систем је пријавио корисника на систем“. (ИА)

Слика 52 Форма са детаљима корисника

Алтернативна сценарија

5.1 Уколико систем не може да пријави корисника он приказује кориснику поруку: “Систем не може да пријави корисника”. (ИА)

Слика 53 Пријава корисника, алтернативни сценарио 1

Пројектовање контролера корисничког интерфејса

Контролер корисничког интерфејса је компонента софтверског система која има више одговорности. Контролер корисничког интерфејса је заслужан за сву логику која се тиче графичког интерфејса, међу чему спадају манипулација података који су приказани на форми као и адекватно мапирање објеката између екранских форми и удаљене серверске апликације. [2]

3.3.2. Пројектовање апликационе логике

Апликациона логика се извршава путем серверске апликације. Серверска страна софтверског система представља програм (или колекцију програма) који је задужен за интеракцију са клијентским апликацијама, пружање функционалности заснованим на пословној логици, као и комуникацији са системом за управљањем базом података. [2]

Комуникација са клијентима

Комуникација између сервера и клијента путем REST API-а (Representational State Transfer) је веома честа и популарна у свету веб развоја. Ова комуникација се најчешће одвија преко HTTP (Hypertext Transfer Protocol) протокола, док се за размену података користи JSON (JavaScript Object Notation) формат.

HTTP протокол дефинише начин комуникације између клијента и сервера. Клијент шаље HTTP захтеве серверу како би добио одговор. Ови захтеви могу бити GET, POST, PUT, DELETE и друге методе које дефинишу различите акције које се желе извршити над ресурсима на серверу. На пример, GET захтев се користи за добијање података са сервера, док се POST захтев користи за слање нових података серверу.

REST архитектура промовише принцип ресурса, где сваки ентитет има јединствени *URL (Uniform Resource Locator)* који га идентификује. Када клијент жели приступити одређеном ресурсу на серверу, шаље HTTP захтев на тај URL.

JSON (JavaScript Object Notation) је формат који се користи за размену структурираних података између сервера и клијента. JSON је читљив и лаган формат који се заснива на JavaScript синтакси, али се може користити са различитим програмским језицима. Подаци се представљају у облику парова кључ-вредност и могу бити организовани у објекте или низове.

Контролер апликационе логике

Контролер апликационе логике представља једну од компоненти серверског дела апликације. Улога контролера је да прихвати захтев прослеђен од стране клијента и преусмери га према сервису који може на адекватан начин да обради захтев у складу са пословном логиком. Након датог процеса, контролер апликационе логике је задужен и да резултат пословне логике врати клијенту. [2]

Пројектовање понашања система – системске операције

Пројектовање понашања софтверског система је процес препознавања потребних корака зарад правилног извршења свих системских операција које су препознате у претходним фазама. За сваку системску операцију је потребно креирати секвенцу корака која је у складу са архитектуром софтверског система. [2]

Уговор УГ1: getAll(List<Movie>): signal;
Веза са СК: СК1, СК3, СК4, СК5, СК6, СК7
Предуслови: /
Постуслови: /

Дијаграм 22 Уговор 1

Уговор УГ2: getMoviesByTitle(title, List<Movie>): signal;
Веза са СК: СК3, СК4, СК5, СК6, СК7
Предуслови: /
Постуслови: /

Дијаграм 23 Уговор 2

Уговор УГ3: getById(Movie): signal;
Веза са СК: СК1, СК3, СК4, СК5, СК6, СК7
Предуслови: /
Постуслови: /

Дијаграм 24 Уговор 3

Уговор УГ4: getAll(List<Person>): signal;
Веза са СК: СК2
Предуслови: /
Постуслови: /

Дијаграм 25 Уговор 4

Уговор УГ5: getById(Person): signal;
Веза са СК: СК2
Предуслови: /
Постуслови: /

Дијаграм 26 Уговор 5

Уговор УГ6: addToWatchlist(Movie): signal;

Веза са СК: СКЗ

Предуслови: Структурна и вредносна ограничење над листом жеља морају бити задовољена и изабрани филм се већ не налази у датој листи жеља.

Постуслови: Листа жеља садржи изабрани филм.

Дијаграм 27 Уговор 6

Уговор УГ7: removeFromWatchlist(Movie): signal;

Веза са СК: СК4

Предуслови: Структурна ограничења над листом жеља морају бити задовољена као и то да се дати филм већ налази унутар листе жеља корисника.

Постуслови: Листа жеља више не садржи изабрани филм.

Дијаграм 28 Уговор 7

Уговор УГ8: createReview(Review): signal;
Веза са СК: СК5
Предуслови: Вредносна и структурна ограничења над рецензијом морају бити задовољена, као и да дати критичар није већ написао рецензију за дати филм.
Постуслови: Креирана је рецензија.

Дијаграм 29 Уговор 8

Уговор УГ9: updateReview(Review): signal;
Веза са СК: СК6
Предуслови: Вредносна и структурна ограничења над рецензијом морају бити задовољена, критичар је већ написао рецензију за дати филм.
Постуслови: Измењена је рецензије.

Дијаграм 30 Уговор 9

Уговор УГ10: updateMovie(Movie): signal;

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над филмом морају бити задовољена.

Постуслови: Измењени су подаци изабраног филма.

Дијаграм 31 Уговор 10

Уговор УГ11: login(User): signal;
Веза са СК: СК8
Предуслови: /
Постуслови: /

Дијаграм 32 Уговор 11

Пројектовање структуре софтверског система

Концептуални модел и системске операције директно утичу на резултат пројектовања структуре софтверског система. [2]

Слика 54 Концептуални модел

Слика 55 Пакет са доменским класама

```java
@Data
public class Actor {
    private Long id;
    private String character;
    private Movie movie;
    private Person playedBy;
}
```

Слика 56 Доменска класа Actor

```java
@Data
public class CrewMember {
    private Long id;
    private String job;
    private String department;
    private Movie movie;
    private Person person;
}
```

Слика 57 Доменска класа CrewMember

```java
public enum Gender {
    2 usages
    MALE,
    2 usages
    FEMALE,
    1 usage
    OTHER
}
```

Слика 58 Енумерација Gender

```java
@Data
public class Genre {
    private Long id;
    private String name;
}
```

Слика 59 Доменска класа Genre

```java
@Data
public class Movie {
    private Long id;
    private String title;
    private Date releaseDate;
    private String overview;
    private Integer runtime;
    private Long budget;
    private Long revenue;
    private String backdropPath;
    private String posterPath;
    private String homepage;
    private String imdb;
    private String originalLanguage;
    private String tagline;
    private List<Genre> genres;
}
```

Слика 60 Доменска класа Movie

```java
@Data
public class Person {
    private Long id;
    private String name;
    private Date birthday;
    private String biography;
    private String imdb;
    private String placeOfBirth;
    private String profilePath;
    private Gender gender;
}
```

Слика 61 Доменска класа Person

```java
@Data
public class Review {
    private Long id;
    private String content;
    private Integer rating;
    private Date timestamp;
    private Movie movie;
    private User author;
}
```

Слика 62 Доменска класа Review

```java
public enum Role implements GrantedAuthority {

    3 usages
    USER,
    6 usages
    CRITIC,
    3 usages
    ADMIN;

    VeljkoBlagojevic
    @Override
    public String getAuthority() { return name(); }
}
```

Слика 63 Енумерација Role

```java
@Entity

@NoArgsConstructor
@AllArgsConstructor
@Builder
@Getter
@Setter
@EqualsAndHashCode
@ToString
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String firstname;
    private String lastname;
    private String email;

    @NaturalId
    private String username;

    @JsonIgnore
    private String password;

    @Enumerated(EnumType.STRING)
    private Role role;
```

Слика 64 Доменска класа User

```java
@Data
public class WatchList {
    private User user;
    private List<Movie> movies;
}
```

Слика 65 Доменска класа WatchList

3.3.3. Пројектовање брокера базе података

Интерфејс која представља перзистентни оквир са свим основним операцијама манипулације над објектима и табелама базе података је CrudRepository интерфејс. Додатне могућности пагинације и сортирања захтева коришћење PagingAndSortingRepository интерфејса. Како би за сваки од наших доменских класа направили репозиторијум класе, једино што је потребно је да се наследе претходно споменути интерфејси. Никаква додатна подешавања нису неопходна. Spring Boot све аутоматски конфигурише, тражи имплементације коришћењем Hibernate радног оквира. Основни CrudRepository има следеће методе:

```
public interface CrudRepository<T, ID> extends Repository<T, ID> {
    1 implementation
    <S extends T> S save(S entity);

    1 usage   2 implementations
    <S extends T> Iterable<S> saveAll(Iterable<S> entities);

    8 usages   1 implementation
    Optional<T> findById(ID id);

    1 usage   1 implementation
    boolean existsById(ID id);

    2 implementations
    Iterable<T> findAll();

    no usages   2 implementations
    Iterable<T> findAllById(Iterable<ID> ids);

    1 implementation
    long count();

    1 usage   1 implementation
    void deleteById(ID id);

    1 implementation
    void delete(T entity);

    no usages   1 implementation
    void deleteAllById(Iterable<? extends ID> ids);

    no usages   1 implementation
    void deleteAll(Iterable<? extends T> entities);

    no usages   1 implementation
    void deleteAll();
}
```

Слика 66 Интерфејс CrudRepository

Целокупни скуп компоненти задужени за комуникацију са базом података се представља путем следећег дијаграма класа:

Слика 67 Хијерархија репозиторијума

JpaRepository представља интерфејс који је специјализован за рад са релационим базама података, при чему наслеђује све методе из већ поменутих PagingAndSortingRepository и CrudRepository интерфејса.

3.3.4. Пројектовање складишта података

На основу доменских класа софтвера пројектоване су табеле (складишта података) релационог система за управљање базом података. Систем за управљање базом података који је коришћен за потребе овог пројекта је MySQL.

Слика 68 Табеле базе података

Table Name: actor | Engine: InnoDB
Database: film_forum_db | Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
actor_id	bigint	20		☑	☑	☑	☑	☐	☐
movie_id	bigint	20		☑	☑	☑	☐	☐	☐
character	varchar	255		☐	☑	☐	☐	☐	☐
person_id	bigint	20		☐	☑	☑	☐	☐	☐
				☐	☐	☐	☐	☐	☐

Табела 11 Складиште података, табела Actor

Table Name: crew_member | Engine: InnoDB
Database: film_forum_db | Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
crew_member_id	bigint	20		☑	☑	☑	☑	☐	☐
movie_id	bigint	20		☑	☑	☑	☐	☐	☐
job	varchar	255		☐	☑	☐	☐	☐	☐
department	varchar	255		☐	☑	☐	☐	☐	☐
person_id	bigint	20		☐	☑	☑	☐	☐	☐

Табела 12 Складиште података, табела CrewMember

Table Name: genre | Engine: InnoDB
Database: film_forum_db | Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
genre_id	bigint	20		☑	☑	☑	☑	☐	☐
name	varchar	255		☐	☑	☐	☐	☐	☐

Табела 13 Складиште података, табела Genre

Table Name: movie | Engine: InnoDB
Database: film_forum_db | Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
movie_id	bigint	20		☑	☑	☑	☑	☐	☐
title	varchar	255		☐	☑	☐	☐	☐	☐
release_date	date			☐	☐	☐	☐	☐	☐
overview	varchar	2047		☐	☑	☐	☐	☐	☐
runtime	bigint	20		☐	☑	☑	☐	☐	☐
budget	bigint	20		☐	☐	☑	☐	☐	☐
revenue	bigint	20		☐	☐	☑	☐	☐	☐
poster_path	varchar	255		☐	☑	☐	☐	☐	☐
backdrop_path	varchar	255		☐	☑	☐	☐	☐	☐
homepage	varchar	255		☐	☐	☐	☐	☐	☐
imdb	varchar	255		☐	☐	☐	☐	☐	☐
original_language	varchar	255		☐	☐	☐	☐	☐	☐
tagline	varchar	255		☐	☑	☐	☐	☐	☐

Табела 14 Складиште података, табела Movie

Table Name: movie_genre | Engine: InnoDB
Database: film_forum_db | Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
movie_id	bigint	20		☐	☑	☑	☐	☐	☐
genre_id	bigint	20		☐	☑	☑	☐	☐	☐

Табела 15 Складиште података, табела Movie_Genre

Table Name: movie_watch_list | Engine: InnoDB
Database: film_forum_db | Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
movie_id	bigint	20		☑	☑	☑	☐	☐	☐
watch_list_id	bigint	20		☑	☑	☑	☐	☐	☐

Табела 16 Складиште података, табела Movie_WatchList

Table Name: person
Database: film_forum_db
Engine: InnoDB
Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
person_id	bigint	20		☑	☑	☑	☑	☐	☐
name	varchar	255		☐	☑	☐	☐	☐	☐
birthday	date			☐	☐	☐	☐	☐	☐
imdb	varchar	255		☐	☐	☐	☐	☐	☐
place_of_birth	varchar	255		☐	☐	☐	☐	☐	☐
profile_path	varchar	255		☐	☐	☐	☐	☐	☐
gender	varchar	255		☐	☑	☐	☐	☐	☐

Табела 17 Складиште података, табела Person

Table Name: review
Database: film_forum_db
Engine: InnoDB
Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
review_id	bigint	20		☑	☑	☑	☑	☐	☐
content	varchar	255		☐	☑	☐	☐	☐	☐
rating	bigint	20		☐	☑	☑	☐	☐	☐
timestamp	date			☐	☑	☐	☐	☐	☐
user_id	bigint	20		☐	☑	☑	☐	☐	☐
movie_id	bigint	20		☐	☑	☑	☐	☐	☐

Табела 18 Складиште података, табела Review

Table Name: user
Database: filmforum
Engine: InnoDB
Character Set: utf8mb4
Collation: utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
id	bigint	20		☑	☑	☐	☐	☐	☐
email	varchar	255		☐	☐	☐	☐	☐	☐
firstname	varchar	255		☐	☐	☐	☐	☐	☐
lastname	varchar	255		☐	☐	☐	☐	☐	☐
password	varchar	255		☐	☐	☐	☐	☐	☐
role	enum	'ADMIN		☐	☐	☐	☐	☐	☐
username	varchar	255		☐	☐	☐	☐	☐	☐
				☐	☐	☐	☐	☐	☐

Табела 19 Складиште података, табела User

Table Name watch_list

Database film_forum_db

Engine InnoDB

Character Set utf8mb4

Collation utf8mb4_general_ci

1 Columns | 2 Indexes | 3 Foreign Keys | 4 Check Constraint | 5 Advanced | 6 SQL Preview

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update
watch_list_id	bigint	20		☑	☑	☑	☑	☐	☐
user_id	bigint	20		☐	☑	☑	☐	☐	☐

Табела 20 Складиште података, табела WatchList

3.3.5. Коначна архитектура софтверског система

Коначна архитектура софтверског система филмског форума има следећи изглед:

Слика 69 Коначна архитектура софтверског система

3.4. Имплементација

Имплементација је фаза у којој почиње да се пише програмски код и повезују модули који су претходно само идејно креирани и чија је архитектура претходно постављена. Овде се уводе конкретне технологије, као и њихове верзије.

Софтверски систем је развијан у програмском језику Јава верзија 17 заједно са Spring Boot оквиром што се тиче серверске стране. Клијентска страна је развијана уз помоћ React библиотеке и TypeScript програмског језика. Систем је пројектован као клијент-сервер веб апликација која се покреће кроз интернет прегледач. Као систем за управљање базом података коришћен је *„MySQL“*, док је развојно окружење *“IntelliJ IDEA / WebStorm”*. На основу архитектуре софтверског система добијене су следеће софтверске класе:

Слика 70 Пакети и класе серверске апликације (први део)

Слика 71 Пакети и класе серверске апликације (други део)

Слика 72 Пакети и класе клијентске апликације (први део)

Слика 73 Пакети и класе клијентске апликације (други део)

3.5. Тестирање

Фаза тестирања представља фазу где је потребно испитивати сваку од имплементираних функционалности, као и све могуће алтернативне случајеве. Тестирање је изузетно битно јер успоставља квалитет једног производа и доприноси сигурности и поузданости софтвера.

Тестирање је могуће урадити и модуларно, где се свака појединачна компонента засебно тестира. Овакав тип изолованог тестирања се назива јединично тестирање (енгл. *Unit testing*). Јава платформа поседује више радних оквира који су задужени за тестирање, где су најпознатији JUnit и Mockito оквири. Поред јединичних тестова, могућа је и имплементација интеграционих и регресионих тестова. Уколико је приликом развоја софтверских система тестирање у првом плану примењена је стратегија развоја софтвера вођена тестовима (енгл. *Test Driven Development*).

Софтверски систем филмског форума је тестиран праћењем свих добрих пракси и препорука. Свака функционалност је тестирана са великим бројем унетих података, чиме је проверен сваки могући алтернативни сценарио. Адекватан приступ при тестирању софтверског система може пружити увереност у висок ниво квалитета софтвера. Након фазе тестирања, софтверски систем се сматра готовим и испоручује се клијенту.

4. Закључак

Садржај овог завршног рада представља заокружену целину развоја софтверског система на примеру филмског форума. Унутар завршног рада дата је теоријска и практична основа за изградњу једног софтверског система. Софтверски систем је имплементиран радним оквирима и технологијама погодне за израду веб апликација које су објашњене у другом поглављу, применом упрошћене Ларманове методе која је описана у трећем поглављу. Концепти који су објашњени у дата два поглавља су довољно општи и генерички да се могу применити на било који други доменски проблем ефикасно.

Кроз дате области су објашњени концепти израде веб апликације са фокусом на радне оквире засноване на Java и JavaScript технологијама. Примењен је ефикасан начин за чување података, примењени су софтверски патерни, имплементирана је заштита веб апликације и извршен је развој корисничког интерфејса који прати добре праксе софтверског инжењерства. Невезано од саме технологије је и проучен приступ изради било каквог софтверског система методом која користи итеративно инкрементaлан модел са дијаграмима случаја коришћења, секвенци и класа на разумљив и логичан начин.

Сам доменски проблем је изабран због своје једноставности и широке распрострањености што га чини погодним за лако праћење и разумевање завршног рада. Још један разлог одабира ове теме је била велика лична љубав према филмовима и критикама. Доменски проблем се своди на израду система који ће чувати и приказивати податке о филмовима, као и омогућити изабраним корисницима да пишу рецензије за исте, и тиме шире своја мишљења и љубав према најмодернијој уметности.

Апликација дозвољава приступ свим подаци о филмовима, док корисници могу филмове и да сачувају у личну листу жеља. Администратор може да мења податке било ког филма, док критичари остављају рецензије са детаљним описима и оценама сваког филма, заједно са могућношћу да мењају те исте рецензије.

Као и сваки софтверски систем, и овај систем има потенцијала за даља унапређења. Могуће је хоризонтално скалирати апликацију додавањем већег броја функционалности. Једна од њих би била детаљнија и дубља манипулација са свим постојећим ентитетима где би се додала функционалност прављења више листи за гледање. Друга би била интеграција са сервисима који пружају преглед филмских реклама, дигитално изнајмљивање филмова или куповину карата у биоскопима. Следећа је функционалност која се назива „пријава једним кликом“ (енгл. *Single Sign On - SSO*) где корисник не мора ручно да прави налог на апликацији већ може искористити постојећи налог са независног система. Како су праћене све добре праксе и принципи пројектовања скалабилног софтверског система, додавање нових функционалности или интеграција са екстерним системима неће бити никакав проблем.

5. Литература

1. Стојановић, Д. (1978). Теорија Филма. Нолит.
2. Влајић, С. (2015). Пројектовање софтвера - скрипта. Златни пресек.
3. Abadi, M., & Cardelli, L. (1996). *A theory of objects*. Springer.
4. Kendal, S. (2009). Object oriented programming using Java. Bookboon.
5. Hoeller, J., Beams, C., Poutsma, A., Laddad, R., Brannen, S., Fisher, M., … Donald, K. (2014). Retrieved from https://docs.spring.io/spring-framework/docs/4.0.x/spring-framework-reference/html/overview.html (Access Date: July 2nd 2023).
6. Влајић, С. (2014). Софтверски патерни (Software patterns). Златни пресек.
7. Freeman, E., & Robson, E. (2014). *Head first design patterns: A brain-friendly guide*. O'Reilly, Edition: 10th Anniversary ed.
8. Yang, H. Y., Tempero, E., & Melton, H. (2008, March). An empirical study into use of dependency injection in java. In 19th Australian Conference on Software Engineering (aswec 2008) (pp. 239-247). IEEE.
9. Hodges, N. (2019). Retrieved from https://betterprogramming.pub/the-3-types-of-dependency-injection-141b40d2cebc (Access Date: July 2nd 2023).
10. Atzeni, P., & De Antonellis, V. (1993). Relational database theory. Benjamin-Cummings Publishing Co., Inc.
11. O'Neil, E. J. (2008, June). Object/relational mapping 2008: hibernate and the entity data model (edm). In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 1351-1356).
12. Bauer, C. (2005). Hibernate in action. Manning.
13. Kainulainen, P. (2012). Spring Data Standard Guide. Packt Publishing Ltd.
14. Derived Query Methods in Spring Data JPA Repositories. (2023). Baeldung. Retrieved from https://www.baeldung.com/spring-data-derived-queries (Access Date: July 2nd 2023).
15. Rodriguez, A. (2008). Restful web services: The basics. IBM developerWorks, 33, 18.
16. Shingala, K. (2019). Json web token (jwt) based client authentication in message queuing telemetry transport (mqtt). arXiv preprint arXiv:1903.02895.
17. Mularien, P. (2010). Spring Security 3. Birmingham, England: Packt Publishing.
18. Antonov, A. (2015). Spring Boot Cookbook. Packt Publishing Ltd.