



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Марко Николић

Интеграција сервиса за претрагу летова применом Enterprise Service Bus обрасца

ЗАВРШНИ РАД
- Основне академске студије -

Нови Сад, 2024.



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:		
Идентификациони број, ИБР:		
Тип документације, ТД:	Монографска документација	
Тип записа, ТЗ:	Текстуални штампани рад	
Врста рада, ВР:	Завршни (Bachelor) рад	
Аутор, АУ:	Марко Николић	
Ментор, МН:	др Жељко Вуковић, доцент	
Наслов рада, НР:	Интеграција сервиса за претрагу летова применом Enterprise Service Bus обрасца	
Језик публикације, ЈП:	Српски	
Језик извода, ЈИ:	Српски	
Земља публикавања, ЗП:	Република Србија	
Уже географско подручје, УГП:	Војводина	
Година, ГО:	2024.	
Издавач, ИЗ:	Ауторски репринт	
Место и адреса, МА:	Нови Сад, Трг Доситеја Обрадовића 6	
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	8/66/57/0/82/0/0	
Научна област, НО:	Електротехничко и рачунарско инжењерство	
Научна дисциплина, НД:	Примењене рачунарске науке и информатика	
Предметна одредница/Кључне речи, ПО:	Сервисно оријентисане архитектуре, интеграција сервиса	
УДК		
Чува се, ЧУ:	Библиотека Факултета техничких наука, Трг Д. Обрадовића 6, Нови Сад	
Важна напомена, ВН:		
Извод, ИЗ:	У овом раду описана је имплементација система за претрагу летова који интегрише системе различитих авио-компанија и помоћне системе у циљу централизоване претраге и резервације летова. Приказани су најзначајнији делови свих система као и начин на који су они међусобно повезани употребом Enterprise Service Bus обрасца.	
Датум прихватања теме, ДП:		
Датум одбране, ДО:	17.9.2024.	
Чланови комисије, КО:	Председник:	др Марко Марковић, доцент
	Члан:	др Јелена Сливка, ванредни професор
	Члан, ментор:	др Жељко Вуковић, доцент
		Потпис ментора



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	Monographic publication
Type of record, TR :	Textual printed material
Contents code, CC :	Bachelor Thesis
Author, AU :	Marko Nikolić
Mentor, MN :	Željko Vuković, PhD, assist. prof.
Title, TI :	Integration of flight search services using the Enterprise Service Bus pattern
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2024.
Publisher, PB :	Author's reprint
Publication place, PP :	Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	8/66/57/0/82/0/0
Scientific field, SF :	Electrical and computer engineering
Scientific discipline, SD :	Applied computer science and informatics
Subject/Key words, S/KW :	Service-oriented architectures, service integration
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	This paper describes the implementation of a flight search system that integrates various airline systems and auxiliary systems to enable centralized flight search and booking. The key components of all systems are presented, as well as the way they are interconnected using the Enterprise Service Bus pattern.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	17.9.2024.
Defended Board, DB :	President: Marko Marković, assist. Prof.
	Member: Jelena Slivka, assoc. Prof.
	Member, Mentor: Željko Vuković, assist. Prof.
	Mentor's sign



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ РАДА

Број:

Датум:

17.9.2024.

(Податке уноси предметни наставник - ментор)

Врста студија:	а) Основне академске студије б) Основне струковне студије
Студијски програм:	Рачунарство и аутоматика
Руководилац студијског програма:	др Милан Рапаић

Студент:	Марко Николић	Број индекса:	РА 69/2020
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	др Жељко Вуковић		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:			
<ul style="list-style-type: none">- проблем – тема рада;- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;- литература			

НАСЛОВ ЗАВРШНОГ (BACHELOR) РАДА:

Интеграција сервиса за претрагу летова применом Enterprise Service Bus обрасца

ТЕКСТ ЗАДАТКА:

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

Руководилац студијског програма	Ментор рада

Примерак за: о - Студента; о - Ментора

САДРЖАЈ

1. УВОД.....	1
2. ПРЕГЛЕД СЛИЧНИХ АПЛИКАЦИЈА И КОРИШЋЕНИХ СОФТВЕРСКИХ ТЕХНОЛОГИЈА.....	3
2.1 Преглед сличних апликација.....	3
2.2 Преглед коришћених софтверских технологија.....	6
3. СПЕЦИФИКАЦИЈА ЗАХТЕВА.....	9
3.1 Функционални захтеви.....	9
3.2 Нефункционални захтеви.....	11
4. СПЕЦИФИКАЦИЈА ДИЗАЈНА.....	12
4.1 Модел података.....	12
4.2 Архитектура система.....	21
5. ИМПЛЕМЕНТАЦИЈА СИСТЕМА.....	23
5.1 Систем авио-компаније 1.....	23
5.2 Систем авио-компаније 2.....	28
5.3 Систем авио-компаније 3.....	33
5.4 Систем за претрагу аеродрома.....	39
5.5 Систем за претрагу летова са председањем.....	40
5.6 Систем за претрагу летова.....	42
6. ДЕМОНСТРАЦИЈА.....	55
7. ЗАКЉУЧАК.....	61
8. ЛИТЕРАТУРА.....	63

1. УВОД

У данашње време, путовање авионом је постало незаменљив начин транспорта, како за пословне тако и за приватне потребе. Са растућим бројем авио-компанија различитих понуда, корисницима је постало све теже да пронађу најповољније и најприкладније летове. Проблем који се јавља у овом контексту је неефикасност и сложеност процеса претраге и упоређивања летова између више авио-компанија. Многи корисници троше знатно време и труд како би пронашли најбоље опције које одговарају њиховим потребама, што може бити изузетно фрустрирајуће и неефикасно.

Мотивација за решавање овог проблема лежи у жељи да се корисницима обезбеди једноставан и ефикасан начин претраге и резервације летова. Развој апликације која интегрисе више система за претрагу летова може значајно олакшати овај процес. Циљ је креирати јединствену платформу која омогућава корисницима да претражују и упоређују летове из више извора на једном месту, чиме се штеди време и новац и чиме се повећава задовољство корисника.

Без оваквих апликација, претрага летова би остала сложена и неефикасна, захтевајући од корисника да посећују више веб сајтова и ручно упоређују резултате. Ово је посебно изазовно у данашњем свету где су брзина и ефикасност кључни фактори. Историјски посматрано, претрага летова се развијала од директног одласка у једну од експозитур авио-компаније, затим контактирања авио-компанија путем телефона, до коришћења онлајн сервиса, али је и даље остала потреба за интеграцијом различитих извора у једну платформу.

Поред тога, велики број корисника се суочава са проблемима приликом резервације летова, као што су скривени трошкови, непрецизне информације о условима лета и ограничене могућности за упоређивање различитих понуда. Ова ситуација доприноси општем незадовољству и несигурности корисника, што је додатни разлог за потребу за једноставним и транспарентним решењем.

Решавање овог проблема допринеће већем задовољству корисника, ефикасности и транспарентности у процесу резервације летова. Увођење оваквих иновација у свет путовања може значајно унапредити корисничко искуство и поставити нове стандарде у индустрији авио-превоза.

Остатак рада је организован на следећи начин. У другом поглављу је дат преглед сличних апликација и коришћених софтверских технологија. Преглед сличних апликација обухвата анализу постојећих решења која нуде сличне функционалности, док преглед коришћених софтверских технологија објашњава технологије и алате који су коришћени у развоју овог пројекта. У трећем поглављу наведена је спецификација захтева, где су дефинисани функционални и нефункционални захтеви система. Функционални захтеви описују основне функције које систем треба да пружи, док нефункционални захтеви обухватају аспекте као што су перформансе, сигурност и корисничко искуство. У четвртом поглављу приказана је спецификација дизајна система. Ово поглавље садржи детаљан модел података који описује структуру и односе између података, као и архитектуру система која показује како су различити делови система међусобно повезани и како комуницирају. У петом поглављу објашњена је имплементација система. Ово поглавље покрива техничке детаље имплементације, укључујући опис коришћених програмских језика, радних оквира и алата, као и кључне аспекте кода и алгоритама који су примењени. У шестом поглављу представљена је

демонстрација система. Овде је приказано како систем функционише у пракси, укључујући примере корисничких сценарија. Седмо поглавље доноси закључна разматрања и правце будућих истраживања. У овом поглављу су сумирани главни резултати рада, разматрани су изазови и ограничења, и предложени су могући правци за даљи развој и унапређење система.

2. ПРЕГЛЕД СЛИЧНИХ АПЛИКАЦИЈА И КОРИШЋЕНИХ СОФТВЕРСКИХ ТЕХНОЛОГИЈА

У овом поглављу су наведена и објашњена сродна решења за претрагу и упоређивање летова између различитих авио-компанија. Преглед сличних апликација обухвата анализу постојећих решења која нуде сличне функционалности и истражује њихове предности и мане. Потом је дат преглед коришћених софтверских технологија, укључујући избор програмских језика, оквира, библиотека и алата који су коришћени у развоју овог система, уз објашњење разлога за њихов избор и њиховог доприноса укупној ефикасности и квалитету развијеног решења.

2.1 Преглед сличних апликација

У овом одељку су наведена и објашњена сродна решења за претрагу и упоређивање летова између различитих авио-компанија. Преглед сличних апликација обухвата анализу постојећих решења која нуде сличне функционалности и истражује њихове предности и мане. Ово укључује детаљан преглед водећих платформи које омогућавају претрагу летова. Нагласак је на функционалностима које нуде, употребљивости као и на корисничком искуству које пружају.

Google Flights [1] је једна од најпопуларнијих платформи за претрагу летова. Креирана од стране компаније Google [2], ова апликација омогућава корисницима да претражују и упоређују летове различитих авио-компанија. Google Flights на почетној страници нуди претрагу летова на основу следећих критеријума претраге: град поласка, дестинација, датум поласка, број путника подељен према старосним категоријама, класа летења. Такође могуће је назначити потребу за повратним летом при чему се додаје филтер за подешавање датума поласка повратног лета. Након успешне претраге летова, сајт води на страницу са интерактивном мапом на којој је назначен град поласка и дестинација са означеном ценом најјевтинијег пронађеног лета. Поред овога, мапа приказује и остале дестинације до којих се може летети из града поласка на означене датуме што омогућава увид у друге опције у случају да се кориснику свиди нека друга дестинација. Коначним одабиром дестинације приказују се сви летови који одговарају критеријумима претраге. Летове је могуће филтрирати на основу следећих критеријума (слика 1): број преседања, одабир авио-компанија које долазе у обзир, количина ручног пртљага која се може унети приликом уласка у авион, цена, максимална дужина лета и повратног лета, максимална дужина пауза приликом преседања, емисија штетних гасова. Додатне функционалности које Google Flights нуди су избор валуте, језика, препоруке о најјевтинијим датумима за летење, препоруке хотела. Кориснички интерфејс прилагођен је раду на рачунару као и на мобилном телефону. Према статистици, око 50% корисника Google Flights корисници мобилних телефона [3]. Лоша страна је што није понуђена опција за директну резервацију карата преко сајта, већ се за то врши редирекција на друге сајтове. Ово може допринети неконзистентом корисничком искуству приликом вишеструког коришћења Google Flights. Такође, не постоји мобилна апликација.





Round trip 1 Economy

Belgrade Barcelona Sat, Nov 16 Fri, Nov 22

All filters Stops Airlines Bags Price Times Emissions Connecting airports Duration

Track prices Nov 16 – 22 Any dates Date grid Price graph

Best departing flights
 Ranked based on price and convenience Prices include required taxes + fees for 1 adult. Optional charges and bag fees may apply. Passenger assistance info. Sort by: ↑

	6:40 AM – 9:20 AM Air Serbia - Air Europa	2 hr 40 min BEG-BCN	Nonstop	133 kg CO2e -43% emissions	RSD 11,449 round trip	▼
Avoids as much CO2e as 6,148 trees absorb in a day						
	2:35 PM – 7:15 PM SWISS	4 hr 40 min BEG-BCN	1 stop 1 hr ZRH	184 kg CO2e -21% emissions	RSD 18,059 round trip	▼
	11:05 AM – 7:45 PM Austrian	8 hr 40 min BEG-BCN	1 stop 5 hr 5 min VIE	214 kg CO2e -9% emissions	RSD 18,350 round trip	▼
	5:10 PM – 10:55 PM Lufthansa - Operated by Lufthansa CityLine	5 hr 45 min BEG-BCN	1 stop 2 hr 5 min MUC	231 kg CO2e Avg emissions	RSD 18,732 round trip	▼

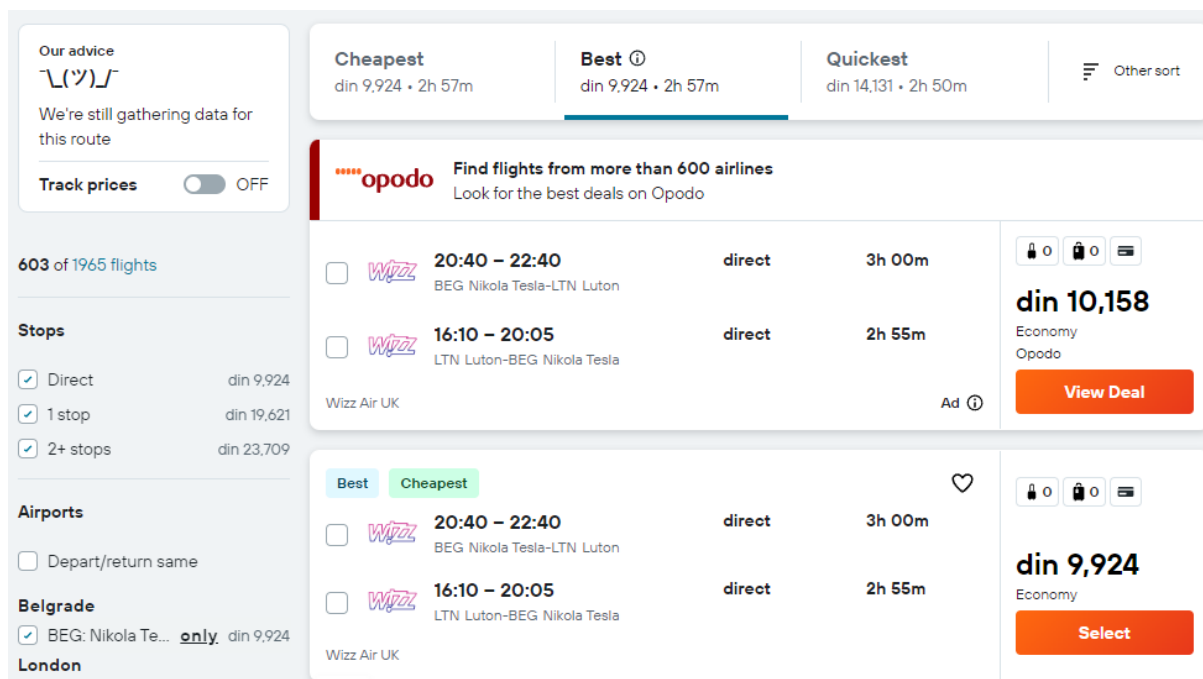
Слика 1 - Google Flights – филтрирање летова

Skyscanner [4] такође је једна од најпопуларнијих платформи за претрагу летова. Платформа је креирана 2002. године и омогућава корисницима да претражују и упоређују летове различитих авио-компанија. Skyscanner на почетној страници нуди претрагу летова на основу следећих критеријума претраге: град поласка, дестинација, датум поласка, датум поласка повратног лета, број путника подељен према старосним категоријама, класа летења. Могуће је назначити да није потребан повратни лет тако што се не одабере датум повратка. Такође, могуће је уз град поласка и дестинацију навести и да је прихватљиво претраживати и на основу околних аеродрома. Након успешне претраге летова, сајт води на страницу са резултатима претраге летова. Летове је могуће филтрирати на основу следећих критеријума (слика 2): број преседања, доб дана поласка, трајање лета, авио-компаније које долазе у обзир, аеродроми, емисија штетних гасова. Додатне функционалности које Skyscanner нуди су избор валуте, језика, укључивање обавештења о кретању цена летова, препоруке хотела и агенција за изнајмљивање аутомобила. Кориснички интерфејс прилагођен је раду на рачунару и на мобилном телефону, а постоји и мобилна апликација доступна на Android и iOS мобилним уређајима. Лоша страна је што није понуђена опција за директну резервацију карата преко сајта, већ се за то врши редирекција на друге сајтове. Такође, подржани су само неки језици за разлику од Google Flights који нуди већи избор језика. Информације о правилима о уносу и наплати пртљага нису доступне и корисници морају сами да поведу рачуна о њима.

The screenshot displays the Skyscanner interface. At the top, there's a 'Show whole month' link and a 'Get Price Alerts' button. The search results show '959 results' sorted by 'Best'. Three summary cards are visible: 'Best' at 17,472 Дин. (2h 58 average), 'Cheapest' at 17,472 Дин. (2h 58 average), and 'Fastest' at 71,329 Дин. (2h 50 average). On the left, a sidebar lists filters: Stops, Departure times, Journey duration, Airlines, Airports, and Flight emissions. The main content area features a Skyscanner advertisement for car hire with the text 'Ride in style (and save)' and 'Save 30% on average comparing car hire. Get that ultimate car hair feeling.' Below this, a flight comparison table shows two options: Wizz Air from BEG to LTN (20:40 to 22:40, 3h Direct) and Wizz Air from LTN to BEG (16:10 to 20:05, 2h 55 Direct). To the right of the table, it says '9 deals from 17,472 Дин.' with a 'Select' button. A small note at the bottom of the flight table states 'This flight emits 47% less CO2e than a typical flight on this route'.

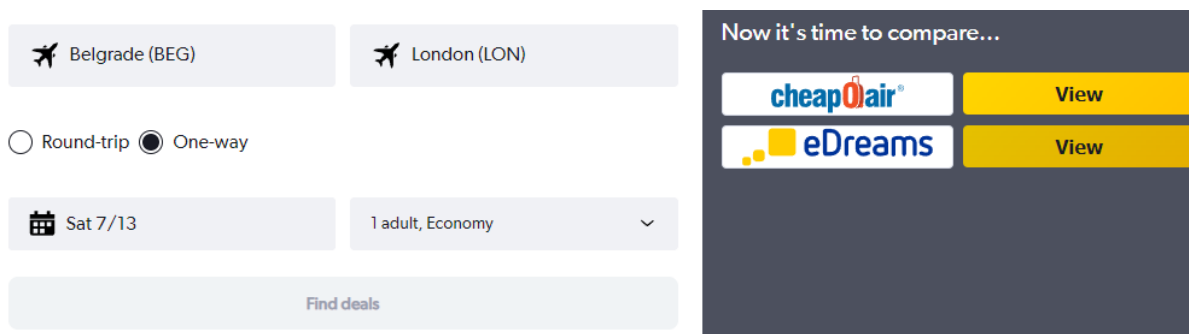
Слика 2 - Skyscanner – филтрирање летова

Kayak [5] је популарна платформа за претрагу летова, смештаја и за проналазак аутомобила за изнајмљивање. Ова апликација омогућава корисницима да претражују и упоређују летове различитих авио-компанија. Кауак на почетној страници нуди претрагу летова на основу следећих критеријума претраге: град поласка, дестинација, датум поласка, број путника подељен према старосним категоријама, класа летења, количина пртљага према категоријама. Такође могуће је назначити потребу за повратним летом при чему се додаје филтер за подешавање датума поласка повратног лета. Поред овакве претраге, понуђена је и опција претраге путем интерактивне мапе намењена корисницима који још нису сигурни где желе да путују. На мапи су приказане дестинације са најјевтинијим ценама лета, као и додатни критеријуми претраге: број преседања, буџет, трајање лета. Коначним одабиром дестинације приказују се сви летови који одговарају критеријумима претраге. Летове је могуће филтрирати на основу следећих критеријума (слика 3): број преседања, аеродроми, количина пртљага, начин плаћања, доб дана поласка, одабир авио-компанија које долазе у обзир, цена, трајање лета, класа летења, додатне авионске услуге, тип и модел авиона, сајтови преко којих је могуће резервисати. Додатне функционалности које Кауак нуди су избор валуте, језика, препоруке хотела и агенција за изнајмљивање аутомобила. Кориснички интерфејс прилагођен је раду на рачунару и на мобилном телефону, а постоји и мобилна апликација доступна на Android и iOS мобилним уређајима. Лоша страна је што није понуђена опција за директну резервацију карата преко сајта за све летове, већ се за то врши редирекција на друге сајтове, али ипак постоје и неки летови који се могу резервисати без редирекције.



Слика 3 - Кауак – филтрирање летова

Cheapflights [6] је платформа за претрагу летова, смештаја и за проналазак аутомобила за изнајмљивање. Ова апликација омогућава корисницима да претражују и упоређују летове различитих авио-компанија. Cheapflights на почетној страници нуди претрагу летова на основу следећих критеријума претраге: град поласка, дестинација, датум поласка, број путника подељен према старосним категоријама, класа летења. Могуће је назначити потребу за повратним летом при чему се додаје филтер за подешавање датума поласка повратног лета. Након претраге приказују се сајтови на којима су пронађени летови који одговарају критеријумима (слика 4). Корисник мора сам да прође кроз сваки сајт и упореди понуду летова. Лоша страна овога је што корисник мора да издвоји више времена за поређење и одабир летова јер не може да види целокупну понуду на једном месту.



Слика 4 - Cheapflights – претрага и резултати претраге

2.2 Преглед коришћених софтверских технологија

У овом одељку су наведене софтверске технологије коришћене у пројекту. Ове технологије су груписане по базама података, радним оквирима и програмским језицима за развој серверских апликација, фронтенду и радним оквирима и програмским језицима за развој серверских апликација, радним оквирима и програмским језицима за развој клијентских апликација, форматима за размену података, приступима комуникације између апликација, алатима за интеграцију и

системима за контролу верзија. Овај преглед обухвата све кључне компоненте које су коришћене за изградњу система.

Базе података:

- PostgreSQL [7] - објектно-релациона база података.
- Microsoft SQL Server [8] - релациона база података развијена од стране Microsoft-а [9].
- MongoDB [10] - NoSQL база података за складиштење и управљање подацима у JSON формату.

Радни оквири и програмски језици за развој серверских апликација:

- Node.js [11] – JavaScript [12] runtime окружење за извршавање JavaScript кода изван веб претраживача.
- Express [13] - Node.js радни оквир за изградњу веб апликација и API-ја.
- Spring [14] - радни оквир за развој Java [15] апликација.
- ASP .NET [16] - веб радни оквир за развој веб апликација користећи .NET и C# [17].
- Go [18] - програмски језик развијен од стране Google-а.

Радни оквири и програмски језици за развој клијентских апликација:

- Angular [19] - платформа и радни оквир за изградњу једностраних апликација коришћењем TypeScript-а [20] и JavaScript-а.

Формати за размена података:

- JSON (JavaScript Object Notation) [21] - текстуални формат за размену података, лак за читање и писање од стране људи и лак за парсирање и генерисање од стране машина.
- XML (eXtensible Markup Language) [22] - језик за означавање који дефинише правила за кодирање докумената у формату који је читљив за људе и машине.
- CSV (Comma-Separated Values) [23] - једноставан текстуални формат који се користи за чување и пренос табеларних података, где су вредности најчешће одвојене зарезом.

Приступи комуникације између апликација:

- REST (Representational State Transfer) [24] - архитектонски стил за дизајнирање мрежних апликација које користе HTTP [25] протокол за размену података. REST омогућава једноставну комуникацију између клијента и сервера путем захтева и одговора који укључују ресурсе представљене URL-овима.
- SOAP (Simple Object Access Protocol) [26] - протокол за размену структурираних информација у дистрибуираним окружењима, користећи XML формат. SOAP се

често користи за интеграцију између различитих апликација или сервиса путем мреже, пружајући строго дефинисану спецификацију за комуникацију.

- GraphQL [27] - упитни језик и окружење које омогућава клијентима да дефинишу тачно које податке желе да добију. GraphQL омогућава ефикаснију и флексибилнију размену података у односу на традиционалне REST API-је, јер клијенти имају начин да одаберу тачно која поља и везе желе да добију у једном захтеву.

Интеграција система:

- MuleSoft Anypoint Studio [28] - интеграциона платформа која омогућава развој, управљање и интеграцију апликација и сервиса у дистрибуираним окружењима. Anypoint Studio пружа алате за израду API-ја, интеграцију података, управљање сервисима и аутоматизацију пословних процеса. Платформа подржава различите протоколе за размену података и омогућава централизовано управљање интеграцијама кроз јединствену конзолу за управљање.

Систем за контролу верзија:

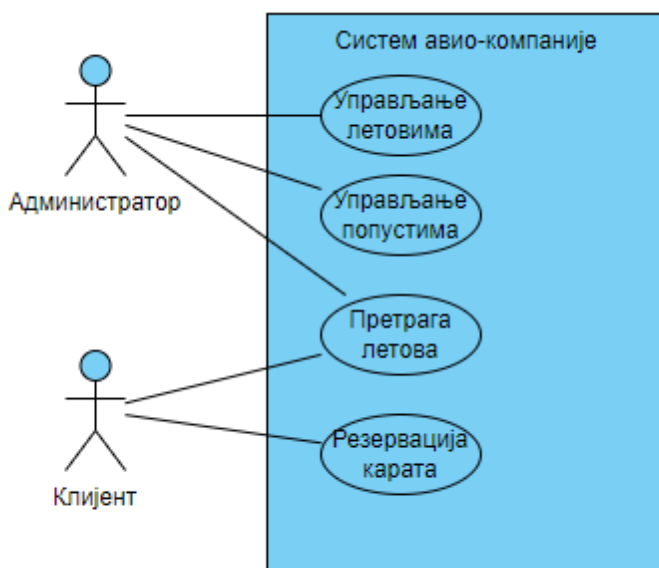
- Git [29] - систем за контролу верзија који омогућава програмерима да прате измене у свом коду током времена.

3. СПЕЦИФИКАЦИЈА ЗАХТЕВА

Овај одељак описује функционалне захтеве система за управљање летовима и резервацијом карата за авио-компанију. Дијаграми случајева коришћења илуструју функционалности. За креирање дијаграма случајева коришћења употребљен је VisualParadigm [30].

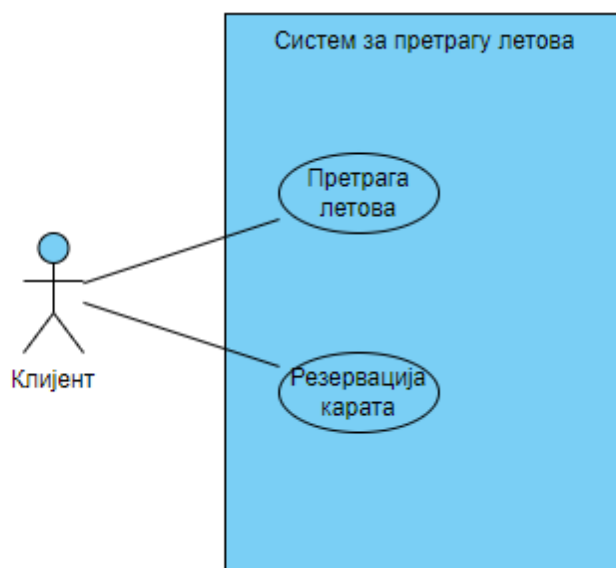
3.1 Функционални захтеви

Сва 3 система авио-компанија треба да омогуће следеће функционалности (слика 5):



Слика 5 - Дијаграм случајева коришћења система авио-компанија

Систем за претрагу летова треба да омогући следеће функционалности (слика 6):



Слика 6 - Дијаграм случајева коришћења система за претрагу летова

Креирање, преглед, измена и брисање летова:

- Предуслови: Администратор је пријављен на систем.
- Кораци:
 1. Администратор уноси информације о лету: место поласка, дестинација, датум и време поласка, датум доласка, број места за сваку класу (економска, пословна, прва), цене карата за сваку класу и правила о пртљагу.
 2. Систем проверава унете податке.
 3. Лет је успешно додат у систем.
- Резултат: Нови лет је креиран и приказан у систему.
- Изузеци: Унети подаци нису валидни.

Креирање, преглед, измена и брисање попушта за летове:

- Предуслови: Администратор је пријављен на систем.
- Кораци:
 1. Администратор бира лет за који жели да дода попуст и уноси информације о почетном и крајњем датуму важења попушта и проценту попушта.
 2. Систем проверава унете податке.
 3. Попуст је успешно додат за изабрани лет.
- Резултат: Нови попуст је додат и приказан у систему за изабрани лет.
- Изузеци: Унети подаци нису валидни.

Претрага летова (систем авио-компаније):

- Кораци:
 1. Корисник уноси критеријуме претраге: место поласка, дестинација, датум поласка, класа летења и број путника.
 2. Систем враћа летове који одговарају унетим критеријумима.
- Резултат: Приказују се сви летови који одговарају унетим критеријумима претраге.

Резервација карата (систем авио-компаније):

- Кораци:
 1. Клијент бира жељени лет, уноси класу летења, мејл адресу и уноси информације о сваком путнику за ког резервише карту (име, презиме, датум рођења, информације о пасошу).
 2. Систем проверава унете податке.

3. Систем резервише карте и шаље их клијенту на мејл у PDF формату.
 4. Карте су успешно резервисане за изабрани лет.
- Резултат: Клијенту се приказује потврда о резервацији карата за изабрани лет, а карте су му послате на мејл.
 - Изузеци: Унети подаци нису валидни или нема доступних места на изабраном лету.

Претрага летова (систем за претрагу летова):

- Кораци:
 1. Корисник уноси критеријуме претраге: место поласка, дестинација, датум поласка, класа летења и број путника, опционо уноси и датум повратног лета и информацију о потреби за искључиво директним летовима.
 2. Систем враћа летове из све 3 авио-компаније који одговарају унетим критеријумима.
- Резултат: Приказују се сви летови који одговарају унетим критеријумима претраге.

Резервација карата (систем за претрагу летова):

- Кораци:
 1. Клијент бира жељене летове, класу летења, мејл адресу и уноси информације о сваком путнику за ког резервише карту (име, презиме, датум рођења, информације о пасошу).
 2. Систем проверава унете податке.
 3. Систем резервише карте за све потребне летове из све 3 авио компаније.
 4. Карте су успешно резервисане за изабрани лет.
- Резултат: Клијенту се приказује потврда о резервацији карата за изабрани лет, а карте су му послате на мејл.

3.2 Нефункционални захтеви

Корисничко искуство (UX):

- Интерфејс треба да буде интуитиван за кориснике свих нивоа, укључујући и неискусне кориснике. Важно је да корисници лако могу да претражују летове и резервишу карте.

Кориснички интерфејс:

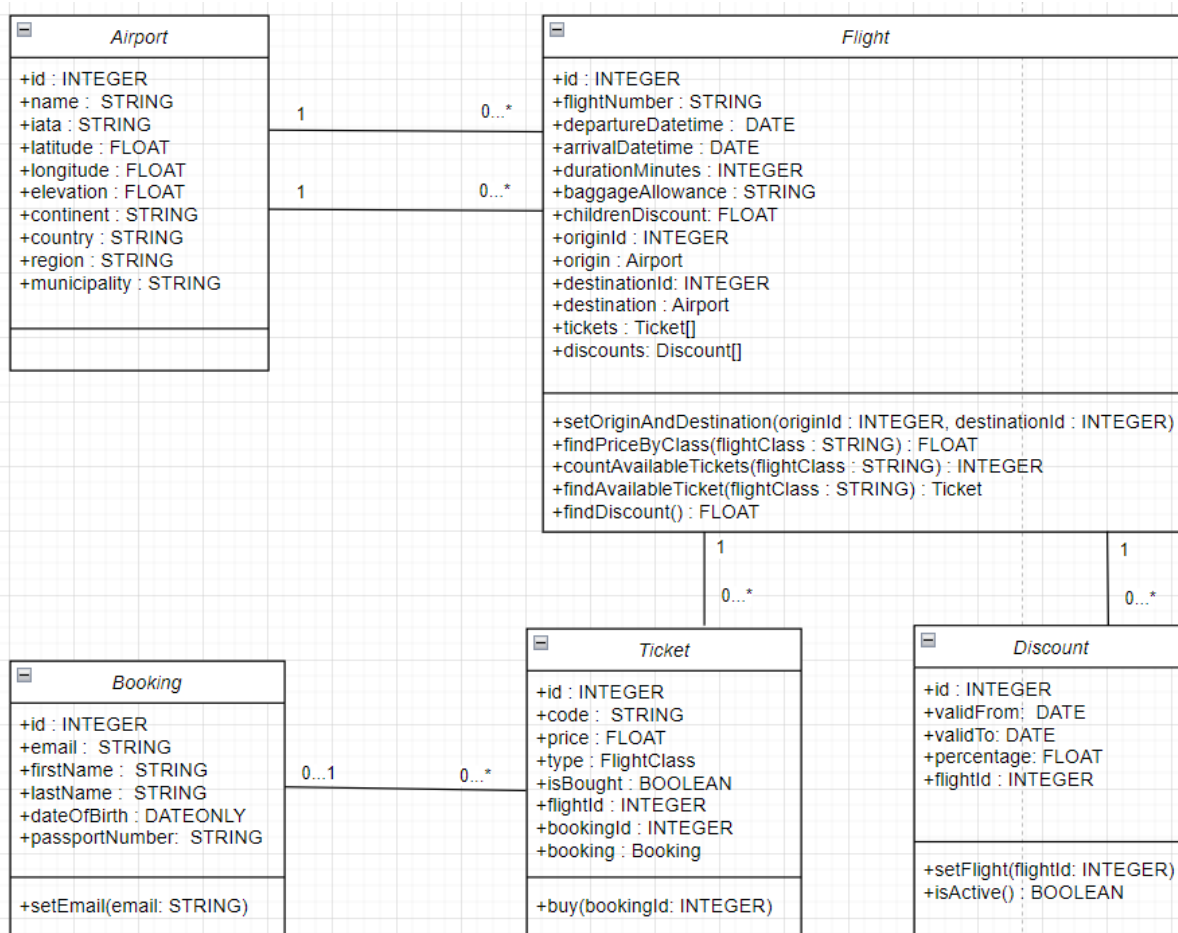
- Фронтенд апликација треба да има responsive дизајн, који се прилагођава различитим величинама екрана на различитим уређајима.

4. СПЕЦИФИКАЦИЈА ДИЗАЈНА

У овом поглављу су приказани модел података и архитектура система. Представљени ће бити модели за сва три система авио-компанија, за систем за претрагу аеродрома и за систем за претрагу летова са преседањем као и архитектура сваког од система. Поред тога, биће приказана и архитектура система за претрагу летова¹.

4.1 Модел података

Модел података прве авио-компаније приказан је на дијаграму (слика 7):



Слика 7 - Дијаграм класа система авио-компаније 1

Класа Airport:

- Атрибути:
 1. name – назив аеродрома.
 2. iata – јединствени идентификатор аеродрома од три слова.
 3. latitude – географска ширина.
 4. longitude – географска дужина.

¹За креирање дијаграма употребљен је Diagrams.net [31].

5. `elevation` – надморска висина.
6. `continent` – континент.
7. `country` – држава.
8. `region` – регион.
9. `municipality` – општина.

Класа `Flight`:

- Атрибути:
 1. `flightNumber` – серијски број лета.
 2. `departureDateTime` – датум и време поласка.
 3. `arrivalDateTime` – датум и време доласка.
 4. `durationMinutes` – трајање лета у минутима.
 5. `baggageAllowance` – правила о пртљагу.
 6. `childrenDiscount` – попуст за децу.
 7. `originId` – страни кључ полазног аеродрома.
 8. `origin` – полазни аеродром.
 9. `destinationId` – страни кључ аеродрома дестинације.
 10. `destination` – аеродром дестинације.
 11. `tickets` – листа карата.
 12. `discounts` – листа попушта.
- Методе:
 1. `setOriginAndDestination` – поставља релацију (почетни и крајњи аеродром).
 2. `findPriceByClass` – проналази цену карата по класи летења.
 3. `countAvailableTickets` – проналази број доступних карата по класи летења.
 4. `findAvailableTicket` – проналази слободну карту у класи летења.
 5. `findDiscount` – проналази активну вредност попушта.

Класа `Discount`:

- Атрибути:
 1. `validFrom` – датум почетка важења попушта.
 2. `validTo` – датум краја важења попушта.
 3. `percentage` – проценат попушта.
 4. `flightId` – страни кључ лета.
- Методе:

1. `setFlight` – поставља лет.
2. `isActive` – проверава да ли је попуст активан.

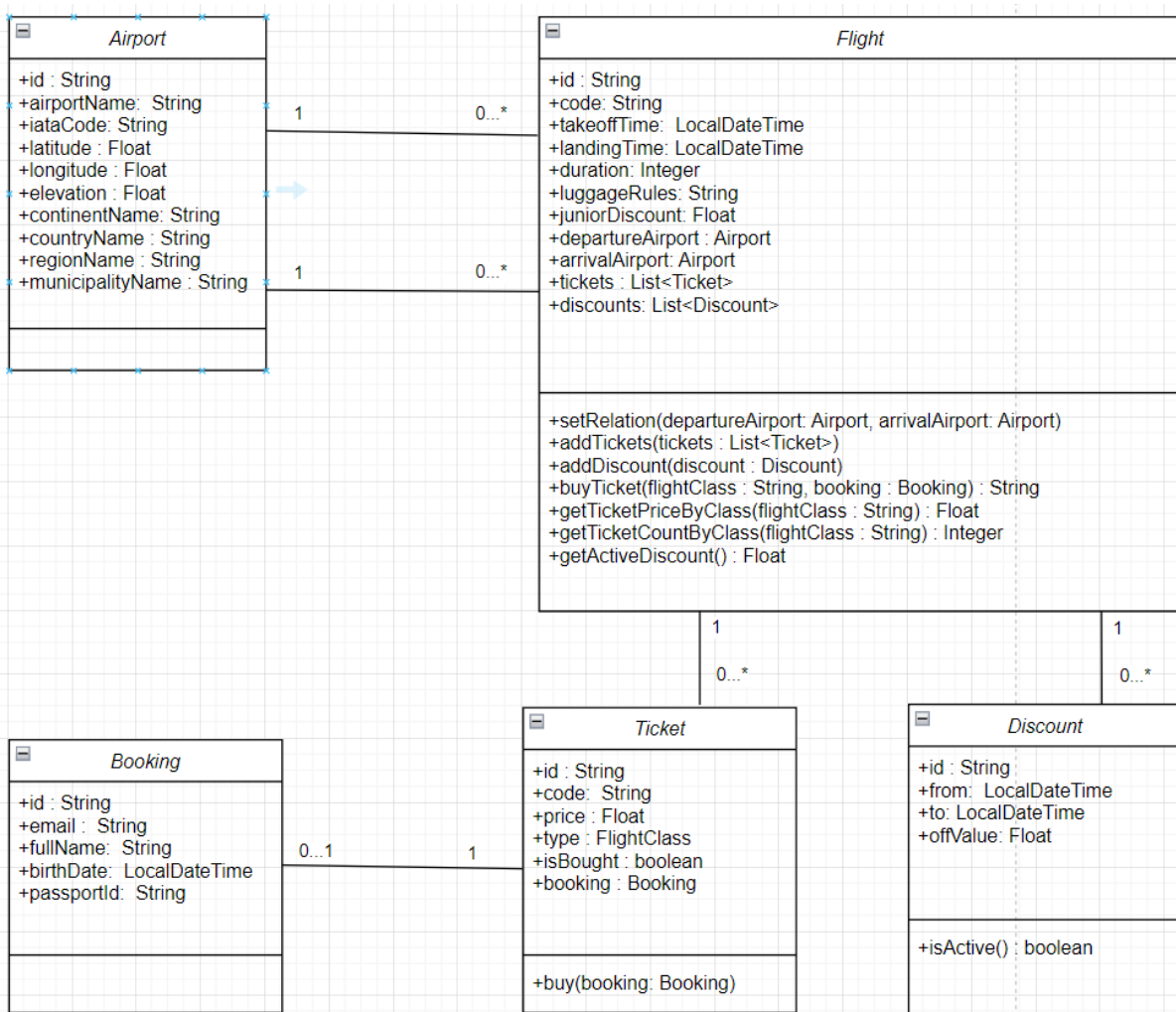
Класа `Ticket`:

- Атрибути:
 1. `code` – серијски број карте.
 2. `price` – цена.
 3. `type` – тип карте (класа летења).
 4. `isBought` – информација о доступности карте.
 5. `flightId` – страни кључ лета.
 6. `bookingId` – страни кључ резервације.
 7. `booking` – резервација.
- Методе:
 1. `buy` – означава карту као купљену.

Класа `Booking`:

- Атрибути:
 1. `email` – мејл купца.
 2. `firstName` – име купца.
 3. `lastName` – презиме купца.
 4. `dateOfBirth` – датум рођења купца.
 5. `passportNumber` – број пасоша купца.
- Методе:
 1. `setEmail` – поставља мејл купца.

Модел података друге авио-компаније приказан је на дијаграму (слика 8):



Слика 8 - Дијаграм класа система авио-компаније 2

Класа Airport:

- Атрибути:
 1. airportName – назив аеродрома.
 2. iataCode – јединствени идентификатор аеродрома од три слова.
 3. latitude – географска ширина.
 4. longitude – географска дужина.
 5. elevation – надморска висина.
 6. continentName – континент.
 7. countryName – држава.
 8. regionName – регион.
 9. municipalityName – општина.

Класа Flight:

- Атрибути:
 1. code – серијски број лета.

2. `takeoffTime` – датум и време поласка.
3. `landingTime` – датум и време доласка.
4. `duration` – трајање лета у минутима.
5. `luggageRules` – правила о пртљагу.
6. `juniorDiscount` – попуст за децу.
7. `departureAirport` – полазни аеродром.
8. `arrivalAirport` – аеродром дестинације.
9. `tickets` – листа карата.
10. `discounts` – листа попушта.

- Методе:

1. `setRelation` – поставља релацију (почетни и крајњи аеродром).
2. `addTickets` – додаје карте.
3. `addDiscount` – додаје попуст.
4. `buyTicket` – купује карту.
5. `getTicketPriceByClass` – проналази цену карата по класи летења.
6. `getTicketCountByClass` – проналази број доступних карата по класи летења.
7. `getActiveDiscount` – проналази активну вредност попушта.

Класа `Discount`:

- Атрибути:

1. `from` – датум почетка важења попушта.
2. `to` – датум краја важења попушта.
3. `offValue` – проценат попушта.

- Методе:

1. `isActive` – проверава да ли је попуст активан.

Класа `Ticket`:

- Атрибути:

1. `code` – серијски број карте.
2. `price` – цена.
3. `type` – тип карте (класа летења).
4. `isBought` – информација о доступности карте.
5. `booking` – резервација.

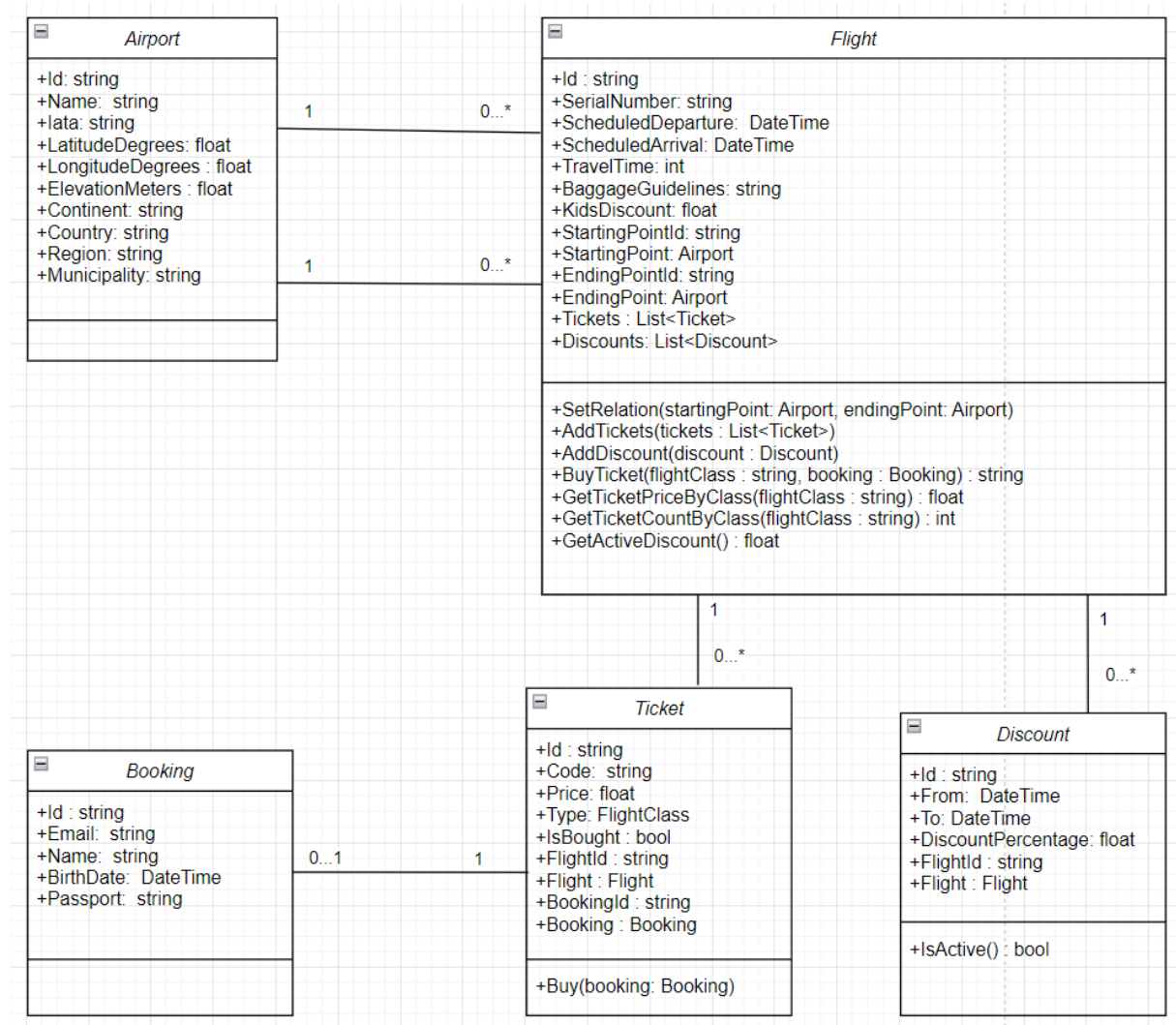
- Методе:

1. `buy` – означава карту као купљену.

Класа Booking:

- Атрибути:
 1. email – мејл купца.
 2. fullName – пуно име купца.
 3. birthDate – датум рођења купца.
 4. passportId – број пасоша купца.

Модел података треће авио-компаније приказан је на дијаграму (слика 9):



Слика 9 - Дијаграм класа система авио-компаније 3

Класа Airport:

- Атрибути:
 1. Name – назив аеродрома.
 2. Iata – јединствени идентификатор аеродрома од три слова.
 3. LatitudeDegrees – географска ширина.
 4. LongitudeDegrees – географска дужина.

5. `ElevationMeters` – надморска висина.
6. `Continent` – континент.
7. `Country` – држава.
8. `Region` – регион.
9. `Municipality` – општина.

Класа `Flight`:

- Атрибути:
 1. `SerialNumber` – серијски број лета.
 2. `ScheduledDeparture` – датум и време поласка.
 3. `ScheduledArrival` – датум и време доласка.
 4. `TravelTime` – трајање лета у минутима.
 5. `BaggageGuidelines` – правила о пртљагу.
 6. `KidsDiscount` – попуст за децу.
 7. `StartingPointId` – страни кључ полазног аеродрома.
 8. `StartingPoint` – полазни аеродром.
 9. `EndingPointId` – страни кључ аеродрома дестинације.
 10. `EndingPoint` – аеродром дестинације.
 11. `Tickets` – листа карата.
 12. `Discounts` – листа попушта.
- Методе:
 1. `SetRelation` – поставља релацију (почетни и крајњи аеродром).
 2. `AddTickets` – додаје карте.
 3. `AddDiscount` – додаје попуст.
 4. `BuyTicket` – купује карту.
 5. `GetTicketPriceByClass` – проналази цену карата по класи летења.
 6. `GetTicketCountByClass` – проналази број доступних карата по класи летења.
 7. `GetActiveDiscount` – проналази активну вредност попушта.

Класа `Discount`:

- Атрибути:
 1. `From` – датум почетка важења попушта.
 2. `To` – датум краја важења попушта.
 3. `DiscountPercentage` – проценат попушта.
 4. `FlightId` – страни кључ лета.

5. Flight – лет.

- Методе:

1. IsActive – проверава да ли је попуст активан.

Класа Ticket:

- Атрибути:

1. Code – серијски број карте.
2. Price – цена.
3. Type – тип карте (класа летења).
4. IsBought – информација о доступности карте.
5. FlightId – страни кључ лета.
6. Flight – лет.
7. BookingId – страни кључ резервације.
8. Booking – резервација.

- Методе:

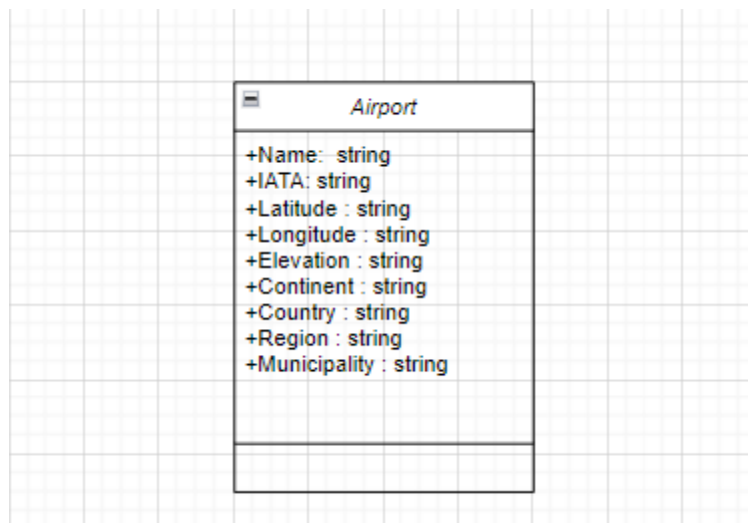
1. Buy – означава карту као купљену.

Класа Booking:

- Атрибути:

1. Email – мејл купца.
2. Name – пуно име купца.
3. BirthDate – датум рођења купца.
4. Passport – број пасоша купца.

Модел података система за претрагу аеродрома приказан је на дијаграму (слика 10):

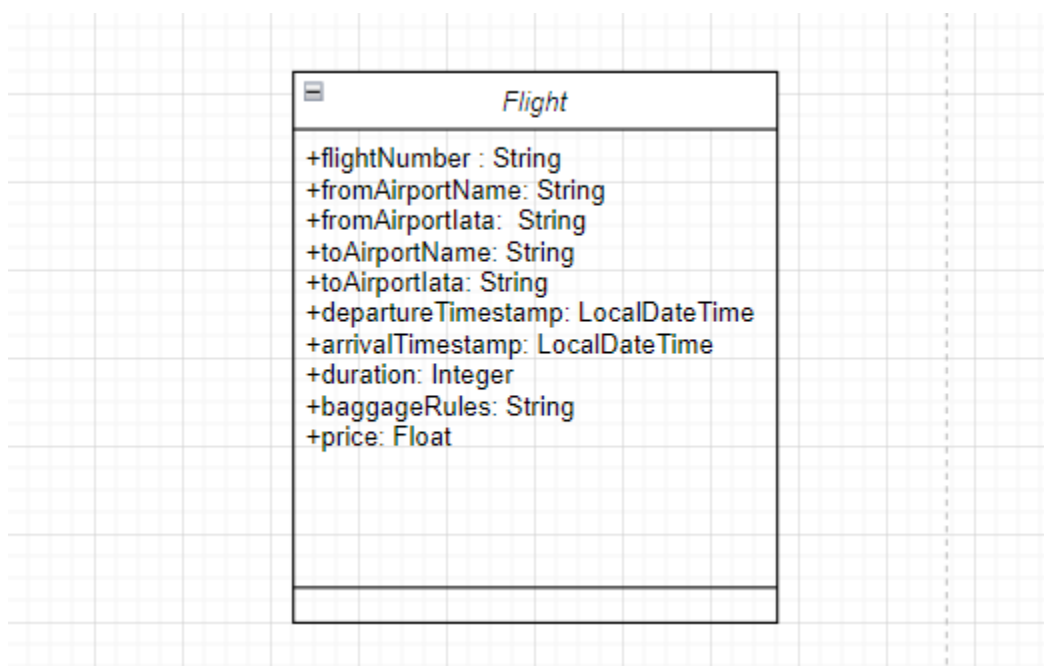


Слика 10 - Дијаграм класа система за претрагу аеродрома

Класа Airport:

- Атрибути:
 1. Name – назив аеродрома.
 2. IATA – јединствени идентификатор аеродрома од три слова.
 3. Latitude – географска ширина.
 4. Longitude – географска дужина.
 5. Elevation – надморска висина.
 6. Continent – континент.
 7. Country – држава.
 8. Region – регион.
 9. Municipality – општина.

Модел података система за претрагу летова са преседањем приказан је на дијаграму (слика 11):



Слика 11 - Дијаграм класа система за претрагу летова са преседањем

Класа Flight:

- Атрибути:
 1. flightNumber – серијски број лета.
 2. departureTimestamp – датум и време поласка.
 3. arrivalTimestamp – датум и време доласка.
 4. duration – трајање лета у минутима
 5. baggageRules – правила о пртљагу..

6. price – цена.
7. fromAirportName – назив полазног аеродрома.
8. fromAirportIata – јединствени идентификатор полазног аеродрома од три слова.
9. toAirportName – назив аеродрома дестинације.
10. toAirportIata – јединствени идентификатор аеродрома дестинације од три слова.

Систем за претрагу летова не ради директно са класама већ са различитим текстуалним форматима репрезентације података. Овај систем од осталих система добија податке и врши конверзију тако да подаци добављени из свих система имају један униформни формат.

4.2 Архитектура система

Архитектура система укључује три апликације авио компаније које нуде различите функционалности за претрагу и резервацију летова. Први систем је базиран на REST API-ју и користи JSON формат за размену података. Други систем користи SOAP API и размењује податке у SOAP Envelope XML формату. Трећи систем користи GraphQL API који такође користи JSON формат за размену података. Сваки од ових система пружа услуге за претрагу летова и њихову резервацију.

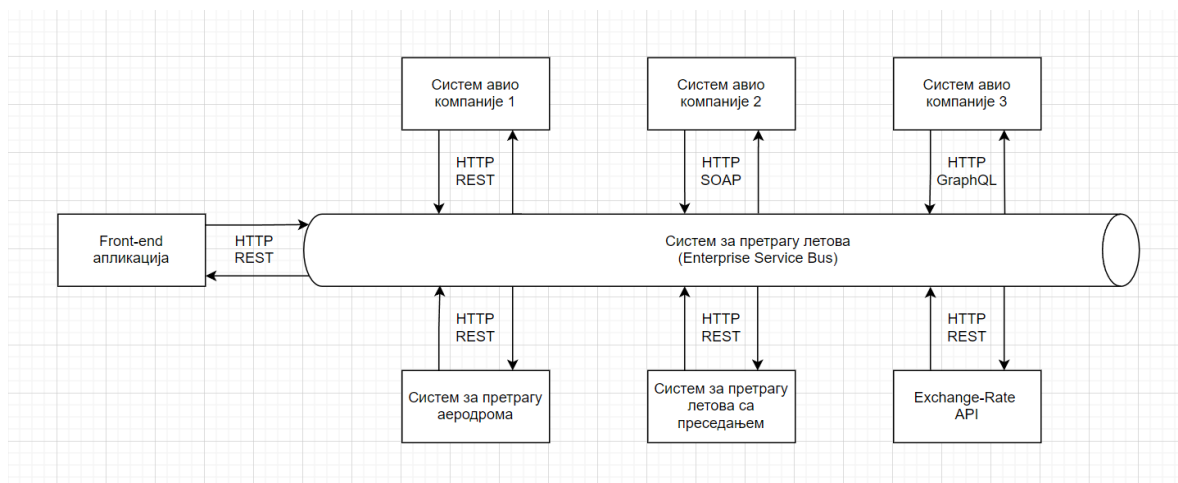
Поред ових система, постоји и систем за претрагу аеродрома који користи CSV формат за размену података преко REST API-ја и нуди функционалност за претрагу аеродрома.

Такође, постоји систем за претрагу летова са преседањем који преко REST API-ја омогућава корисницима да претражују путање које укључују летове са преседањем.

Сви ови системи су интегрисани кроз систем за претрагу и резервацију летова који обједињује функционалности система све три авио-компаније, система за претрагу аеродрома и система за претрагу летова са преседањем. Ова интеграција пружа корисницима већи избор летова и могућност претраге индиректних путања. Поред тога, овај интеграциони систем је повезан и са Exchange-Rate API-јем [32] како би се омогућила конверзија валута у реалном времену.

Систем за претрагу летова имплементиран је користећи ESB (Enterprise Service Bus) шаблон [33]. Enterprise Service Bus је софтверска архитектура која омогућава комуникацију и интеракцију између различитих апликација. Он функционише као посредник који омогућава повезивање и интеграцију различитих система и сервиса, без обзира на њихову платформу, технологију, коришћене протоколе или формате података. ESB олакшава управљање подацима, трансформацију порука, рутирање, и оркестрацију сервиса, што доприноси бољој флексибилности, скалабилности и одрживости система. Користећи ESB, организације могу лакше интегрисати нове апликације и сервисе у постојећу инфраструктуру, смањујући трошкове и сложеност интеграције.

Дијаграм на слици 12 пружа увид у архитектуру целокупног система на високом нивоу апстракције.



Слика 12 - Архитектура система

5. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

У овом поглављу је представљена имплементација 3 система авио-компанија, система за претрагу аеродрома, система за претрагу летова са преседањем и система за претрагу летова и интеграцију осталих поменутих система.

5.1 Систем авио-компаније 1

У овом одељку представљена је имплементација првог система авио-компаније и његови кључни делови. Овај систем омогућава администрацију летова и попушта, као и претрагу летова и резервацију карата за кориснике. У наставку је детаљни опис кључних делова система и технологија које су коришћене у имплементацији.

Систем авио-компаније 1 имплементиран је уз помоћ следећих технологија: Node.js, Express и PostgreSQL. За тестирање REST захтева коришћен је Swagger UI [34].

PostgreSQL је систем за управљање релационим базама података. У систему авио-компаније 1, PostgreSQL се користи за складиштење свих релевантних података о аеродромима, летовима, картама, попустима, резервацијама. Сваки од ових ентитета се чува у одвојеним табелама у бази података.

Node.js служи као платформа за извршавање JavaScript кода на серверу и пружа широк спектар библиотека и модула који олакшавају развој сложених апликација. Express је радни оквир за Node.js који пружа флексибилан начин за креирање и управљање HTTP рутама. Ово омогућава дефинисање REST API крајњих тачака за различите функције система. Node.js и Express су коришћени за имплементацију серверског дела система авио-компаније 1, при чему је изграђен REST API.

За размену података REST API користи JSON формат.

Имплементација система авио-компаније 1 је подељена на неколико слојева како би се обезбедила боља организација кода. Ови слојеви су: модел, репозиторијум, сервис, контролер, рутирање, конфигурација и помоћни слој.

Модел слој садржи дефиниције ентитета који се користе у систему, објекте за пренос података као и логику за мапирање између ентитета и објеката за пренос података. Овај слој комуницира директно са PostgreSQL базом података, користећи библиотеке за објектно-релационо мапирање. За имплементацију овог система, у сврху објектно-релационог мапирања, употребљена је библиотека Sequelize [35]. Ентитети који се чувају у бази података су следећи: аеродром (ентитет Airport), лет (ентитет Flight), карта (ентитет Ticket), попуст (ентитет Discount), резервација (ентитет Booking). На листингу 1 приказан је скраћени поглед на ентитет Booking. Модел Booking и његови атрибути дефинисани су у оквиру Sequelize фрејмворка, користећи његове могућности за спровођење интегритета података и валидацију. Такође, дефинисане су и прототипске методе као што је setEmail. На сличан начин дефинисани су сви остали ентитети овог система.

```

1  const Sequelize = require('sequelize');
2  const sequelize = require('../configuration/database');
3
4  const Booking = sequelize.define('Booking', {
5    id: {
6      type: Sequelize.INTEGER,
7      autoIncrement: true,
8      primaryKey: true,
9    },
10 > email: { ...
21 > },
22 > firstName: { ...
30 > },
31 > lastName: { ...
39 > },
40 > dateOfBirth: { ...
51 > },
52 > passportNumber: { ...
60 > },
61 > }, { ...
63 > });
64
65 Booking.prototype.setEmail = function (email) {
66   this.email = email;
67 };
68
69 module.exports = { Booking };

```

Листинг 1 - Модел Booking

Објекти за пренос података имају 2 улоге. Прва улога је енкапсулирање података које корисник шаље апликацији и који су потребни апликацији да обради захтев. Друга улога је енкапсулирање података које апликација шаље као одговор. Ове класе се директно мапирају на тело HTTP захтева и одговора у JSON формату. На листингу 2 приказан је пример једног објекта за пренос података, BookingDTO.

```

1  const { PassengerDTO } = require("../PassengerDTO");
2
3  class BookingDTO {
4    constructor(flightNumber, flightClass, email, passengers) {
5      this.flightNumber = flightNumber;
6      this.flightClass = flightClass;
7      this.email = email;
8      this.passengers = passengers.map(passenger => new PassengerDTO(
9        passenger.firstName,
10       passenger.lastName,
11       passenger.dateOfBirth,
12       passenger.passportNumber
13     ));
14   }
15 }
16
17 module.exports = { BookingDTO };

```

Листинг 2 - Објекат за пренос података BookingDTO

Логика за мапирање између ентитета и објекта за пренос података имплементирана је помоћу Sequelize build методе која за разлику од create методе омогућава да се ентитет креира без чувања у базу. Овим се избегавају ситуације где ће захтеви са грешкама довести до непотребних и лоших промена у бази података. На листингу 3 дат је пример мапирања између објекта за пренос података BookingDTO и ентитета Booking.

```
1  const { Booking } = require('../domain/Booking');
2
3  function fromDTO(bookingDTO) {
4    return Booking.build({
5      firstName: bookingDTO.firstName,
6      lastName: bookingDTO.lastName,
7      dateOfBirth: bookingDTO.dateOfBirth,
8      passportNumber: bookingDTO.passportNumber,
9    });
10 }
11
12 module.exports = {
13   fromDTO
14 };
```

Листинг 3 - Објекат за пренос података BookingDTO

Репозиторијумски слој пружа апстракцију између модел слоја и сервис слоја. Репозиторијуми садрже логику за приступ подацима, као што су операције за креирање, читање, измену и брисање података. На листингу 4 приказан је ticketRepository са методама за креирање и измену. Параметар options омогућава трансакције над базом података. На сличан начин имплементирани су сви остали репозиторијуми.

```
1  async function create(ticket, options = {}) {
2    return await ticket.save({ options });
3  }
4
5  async function update(ticket, options = {}) {
6    return await ticket.save({ options });
7  }
8
9  module.exports = {
10    create,
11    update
12 };
```

Листинг 4 - Репозиторијум ticketRepository

Сервис слој садржи пословну логику апликације. Овај слој користи репозиторијуме за добијање и манипулацију подацима, позива мапирање за конверзију података и примењује пословна правила. У овом слоју се налази логика везана за све кључне функционалности овог система.

За креирање летова задужен је сервис flightService и метода create (листинг 5). Ова метода започиње трансакцију над базом података и проверава податке из захтева тако што провери да ли у систему постоје аеродроми из релације лета, да ли је датум лета у будућности, а осталу валидацију препушта валидационим правилима дефинисаним помоћу Sequelize библиотеке у моделу. Након тога креира лет у бази података и позива методу createAllTickets из сервиса ticketService који је задужен за креирање карата. Методи се прослеђује и текућа трансакција. Метода

createAllTickets за сваку класу летења креира тражени број карата са траженом ценом. У случају грешке у овом процесу, све измене у бази података се поништавају.

```
9  async function create(flightInformation) {
10    let transaction;
11    try {
12      transaction = await sequelize.transaction();
13
14      const originAirport = await airportService.checkIfExists(flightInformation.originIATA);
15      const destinationAirport = await airportService.checkIfExists(flightInformation.destinationIATA);
16
17      let flight = flightMapper.fromDTO(flightInformation);
18      flight.setOriginAndDestination(originAirport.id, destinationAirport.id);
19
20      dateValidator.isFutureDate(flight.departureDatetime, true);
21
22      flight = await flightRepository.create(flight, { transaction });
23
24      const ticketPricing = ticketPricingMapper.fromDTO(flightInformation);
25      await ticketService.createAllTickets(flight.id, ticketPricing, transaction);
26
27      await transaction.commit();
28    } catch (error) {
29      if (transaction) await transaction.rollback();
30      throw error;
31    }
32  }
```

Листинг 5 - Метода create сервиса flightService

За претрагу летова је такође задужен сервис flightService и његове методе search (листинг 6) као и помоћне методе searchByPassengerInformation, searchByDateRange, searchByRelation. Метода search добавља све летове преко репозиторијума, позива остале 3 методе једну за другом и на крају мапира листу летова на листу објеката за пренос података. Остале 3 методе позивају помоћне методе, одређују да ли је параметар претраге валидан и да ли постоји и филтрирају листу летова помоћу Array.prototype.filter() методе.

```
50  async function search(from, to, origin, destination, flightClass, passengerCount) {
51    let flights = await flightRepository.findAll();
52
53    if (!flights || flights.length === 0) {
54      return [];
55    }
56
57    flights = await searchByDateRange(flights, from, to);
58    flights = await searchByRelation(flights, origin, destination);
59    flights = await searchByPassengerInformation(flights, flightClass, passengerCount);
60
61    const flightDTOs = flightMapper.toDTOList(flights);
62
63    return flightDTOs;
64  }
```

Листинг 6 - Метода search сервиса flightService

За резервацију летова задужен је сервис bookingService и методе bookAll (листинг 7) и bookOne. Сервис врши валидацију захтева, и за сваког путника дефинисаног у захтеву покушава да резервише једну карту за тражени лет, у траженој класи летења. За ово се отвара трансакција над базом података. У случају грешке, у бази података неће бити промена. Након успешне резервације, на мејл купца шаљу се карте у PDF формату. PDF фајл генерише се попуњавањем HTML шаблона и његовом конверзијом у PDF формат употребом html-pdf [36] библиотеке. Након конверзије, PDF

се шаље на мејл купца уз попуњавање HTML шаблона за мејл поруку, везивање PDF фајла за поруку и слање поруке употребом nodemailer [37] библиотеке.

```
9  async function bookAll(bookings) {
10    const flight = await flightService.findByFlightNumber(bookings.flightNumber);
11
12    const validFlightClasses = ['Economy', 'Business', 'First'];
13    if (!bookings.flightClass || !validFlightClasses.includes(bookings.flightClass)) { ...
14  }
15
16
17    let transaction;
18    try {
19      transaction = await sequelize.transaction();
20
21      const ticketCodes = []
22      for (const passenger of bookings.passengers) {
23        const ticketCode = await bookOne(bookings.flightNumber, bookings.flightClass, bookings.email, passenger, transaction);
24        ticketCodes.push(ticketCode)
25      }
26      await transaction.commit();
27      generatePdfAndSendEmail(bookings.email, flight, bookings.flightClass, bookings.passengers, ticketCodes);
28    } catch (error) {
29      if (transaction) await transaction.rollback();
30      throw error;
31    }
32  }
```

Листинг 7 - Метода bookAll сервиса bookingService

Остале функционалности које сервисни слој пружа јесте дефинисање попушта за летове и претрага аеродрома у систему.

Контролер слој је задужен за обраду HTTP захтева и враћање одговора клијенту. Контролери користе сервисе за обављање пословне логике и формирање одговора.

Слој за рутирање управља прослеђивањем HTTP захтева од клијената ка одговарајућим контролерима у систему. На листингу 8 приказане су дефиниције рута за резервацију карата.

```
1  const express = require('express');
2  const router = express.Router();
3  const bookingController = require('../controller/bookingController');
4
5  > /** ...
11
12 > /** ...
60
61 > /** ...
81  router.post('/bookings', bookingController.book);
82
83  module.exports = router;
```

Листинг 8 - Руте за резервацију карата

Конфигурациони слој је одговоран за управљање свим конфигурационим подешавањима система, као што су параметри базе података, дефиниција односа између ентитета, као и подешавање Swagger UI алата за тестирање HTTP захтева. Овај слој обезбеђује централизовану локацију за сва конфигурациона подешавања, што олакшава управљање и одржавање система.

У помоћном слоју налазе се компоненте које нису директно везане за пословну логику, али су потребне сервисима. У овом слоју налази се логика за генерисање PDF документа из HTML шаблона, за слање мејлова.

5.2 Систем авио-компаније 2

У овом одељку представљена је имплементација другог система авио-компаније и његови кључни делови. Овај систем омогућава администрацију летова и попушта, као и претрагу летова и резервацију карата за кориснике. У наставку је детаљни опис кључних делова система и технологија које су коришћене у имплементацији.

Систем авио-компаније 1 имплементиран је уз помоћ следећих технологија: Java, Spring и MongoDB. За тестирање SOAP захтева коришћен је SOAP UI [38].

MongoDB је NoSQL база података за складиштење и управљање подацима у JSON формату. У систему авио-компаније 2, MongoDB се користи за складиштење свих релевантних података о аеродромима, летовима, картама, попустима, резервацијама. Сваки од ових ентитета се чува у засебним колекцијама или као део другог ентитета кроз асоцијације.

Spring је радни оквир за Java програмски језик који пружа флексибилан начин за креирање и управљање различитим аспектима апликација. Ово омогућава дефинисање SOAP веб сервис крајњих тачака за различите функције система. Java и Spring су коришћени за имплементацију серверског дела система авио-компаније 2, при чему је изграђен SOAP веб сервис.

За размену података SOAP веб сервис користи XML SOAP Envelope формат.

Имплементација система авио-компаније 2 је подељена на неколико слојева како би се обезбедила боља организација кода. Ови слојеви су: модел, репозиторијум, сервис, слој крајњих тачака, конфигурација, помоћни слој.

Модел слој садржи дефиниције ентитета који се користе у систему. Ентитети који се чувају у бази података су следећи: аеродром (ентитет Airport), лет (ентитет Flight), карта (ентитет Ticket), попуст (ентитет Discount), резервација (ентитет Booking). На листингу 9 приказан је скраћени поглед на ентитет Booking. Модел Booking и његови атрибути дефинисани су у оквиру Jakarta Validation API-ја [39], користећи његове могућности за спровођење интегритета података и валидацију. Сваки ентитет има имплементирану методу за валидацију. На сличан начин дефинисани су сви остали ентитети овог система. Ентитети се чувају у бази података као колекције, при чему постоје колекције за аеродроме и летове. Сви остали ентитети чувају се угњеждено унутар летова.

```

9 public class Booking extends BaseEntity {
10     @NotNull(message = "Validation: Full name is required")
11     @NotEmpty(message = "Validation: Full name cannot be empty")
12     private String fullName;
13
14     @NotNull(message = "Validation: Email name is required")
15     @NotEmpty(message = "Validation: Email name cannot be empty")
16     @Email(message = "Validation: Email should be valid")
17     private String email;
18
19     @NotNull(message = "Validation: Birth date is required")
20     private LocalDateTime birthDate;
21
22     @NotNull(message = "Validation: Passport ID is required")
23     @NotEmpty(message = "Validation: Passport ID cannot be empty")
24     private String passportId;
25
26 > public Booking(String fullName, String email, LocalDateTime birthDate, String passportId) { ...
32
33 > public String getFullName() { ...
36
37 > public LocalDateTime getBirthDate() { ...
40
41 > public String getPassportId() { ...
44
45     @Override
46 > public void validate() { ...
55
56 }

```

Листинг 9 - Модел Booking

Репозиторијумски слој пружа апстракцију између модел слоја и сервис слоја. Репозиторијуми садрже логику за приступ подацима, као што су операције за креирање, читање, измену и брисање података. На листингу 10 приказан је FlightRepository. Репозиторијум наслеђује MongoRepository интерфејс из Spring Data MongoDB. Додатне методе које нису предвиђене интерфејсом се морају додати. На сличан начин имплементирани су сви остали репозиторијуми.

```

10 @Repository
11 public interface FlightRepository extends MongoRepository<Flight, String> {
12     Optional<Flight> findByCode(String code);
13 }

```

Листинг 10 - Репозиторијум FlightRepository

Сервис слој садржи пословну логику апликације. Овај слој користи репозиторијуме за добијање и манипулацију подацима, позива мапирање за конверзију података и примењује пословна правила. За сваки сервис у овом систему постоји интерфејс и имплементација. У овом слоју се налази логика везана за све кључне функционалности овог система.

За креирање летова задужен је сервис FlightService и метода create (листинг 11). Ова метода проверава податке из захтева тако што провери да ли у систему постоје аеродроми из релације лета, а осталу валидацију препушта валидационим правилима дефинисаним помоћу Jakarta Validation API-ја у моделу. Након тога генерише серијски број лета и позива методу generateTickets из сервиса TicketService који је задужен за креирање карата. Метода generateTickets за сваку класу летења креира тражени број карата са траженом ценом. Након успешног генерисања карата, карте се додају у лет и заједно са њим чувају у бази података.

```

35  @Override
36  public String create(Flight flight, String departureIata, String arrivalIata, TicketPricing pricing) {
37      try {
38          Airport departureAirport = airportRepository.findByIataCode(departureIata).orElseThrow();
39          Airport arrivalAirport = airportRepository.findByIataCode(arrivalIata).orElseThrow();
40
41          flight.setRelation(departureAirport, arrivalAirport);
42
43          String code = generateFlightCode(flight);
44          flight.setCode(code);
45
46          List<Ticket> tickets = ticketService.generateTickets(code, pricing);
47          flight.addTickets(tickets);
48
49          flight.validate();
50          flightRepository.save(flight);
51
52          return "Flight created.";
53      } catch (Exception e) {
54          return e.getMessage();
55      }
56  }

```

Листинг 11 - Метода **create** сервиса **FlightService**

За претрагу летова је такође задужен сервис **FlightService** и његове методе **search** (листинг 12) као и помоћне методе **searchByPassengerInformation**, **searchByDateRange**, **searchByRelation**. Метода **search** добавља све летове преко репозиторијума, позива остале 3 методе једну за другом и на крају мапира листу летова на листу објеката за пренос података. Остале 3 методе позивају помоћне методе, одређују да ли је параметар претраге валидан и да ли постоји и филтрирају листу летова помоћу **Array.prototype.filter()** методе.

```

65  public List<Flight> search(LocalDateTime from, LocalDateTime to,
66      String departureAirport, String arrivalAirport, String flightClass, Integer passengerCount)
67  {
68      List<Flight> flights = flightRepository.findAll();
69
70      if (flights.isEmpty()) {
71          return List.of();
72      }
73
74      flights = searchByDateRange(flights, from, to);
75      flights = searchByRelation(flights, departureAirport, arrivalAirport);
76      flights = searchByPassengerInformation(flights, flightClass, passengerCount);
77
78      return flights;
79  }

```

Листинг 12 - Метода **search** сервиса **FlightService**

За резервацију летова задужен је сервис **BookingService** и методе **book** (листинг 13) и **bookOne**. Сервис врши валидацију захтева, и за сваког путника дефинисаног у захтеву покушава да резервише једну карту за тражени лет, у траженој класи летења. Након успешне резервације, на мејл купца шаљу се карте у PDF формату. PDF фајл генерише се попуњавањем HTML шаблона и његовом конверзијом у PDF формат употребом **iText PDF** [40] библиотеке. Након конверзије, PDF се шаље на мејл купца уз попуњавање HTML шаблона за мејл поруку, везивање PDF фајла за поруку и слање поруке употребом **Jakarta Mail** [41] библиотеке.

Остале функционалности које сервисни слој пружа јесте дефинисање попушта за летове и претрага аеродрома у систему.

```

29 @Override
30 public String book(String flightCode, String flightClass, String email, List<Booking> bookings) {
31     try {
32         Flight flight = flightRepository.findByCode(flightCode).orElseThrow();
33
34         FlightClass flightClassFilter = validateClass(flightClass);
35
36         List<String> ticketCodes = new ArrayList<>();
37         for(Booking booking : bookings) {
38             String ticketCode = bookOne(flight, flightClassFilter, booking);
39             ticketCodes.add(ticketCode);
40         }
41
42         flightRepository.save(flight);
43
44         PdfGenerator pdfGenerator = new PdfGenerator();
45         byte[] pdf = pdfGenerator.generateTicketsPDF(flight, flightClass, bookings, ticketCodes);
46
47         emailSender.sendTicketsEmail(email, pdf);
48
49         return "Booking created.";
50     } catch(Exception e) {
51         return e.getMessage();
52     }
53 }

```

Листинг 13 - Метода book сервиса BookingService

Слој крајњих тачака (endpoint слој) је одговоран за руковање SOAP захтевима и одговорима у систему. У овом слоју се налазе дефиниције SOAP захтева и одговора, адаптери података који служе за трансформацију података између различитих формата, као и саме endpoint класе које обрађују долазне захтеве и враћају одговоре. Endpoint класе комуницирају са сервисима како би испунили пословну логику и обезбедили исправне одговоре на захтеве клијената. Овај слој обезбеђује интерфејс за клијенте који користе SOAP за комуникацију са системом.

SOAP захтеви и одговори дефинисани су помоћу Jakarta XML Binding [42] библиотеке (листинг 14). За сваку класу и њена поља дефинисана је XML структура, додати потребни адаптери за податке као и мапирање на класе из моделског слоја.

```

13 @XmlAccessorType(XmlAccessType.FIELD)
14 @XmlRootElement(name = "Schedule")
15 public class FlightScheduleResponse {
16
17     @XmlElement(name = "TakeOffTime")
18     @XmlJavaTypeAdapter(LocalDateAdapter.class)
19     private LocalDateTime takeoffTime;
20
21     @XmlElement(name = "LandingTime")
22     @XmlJavaTypeAdapter(LocalDateAdapter.class)
23     private LocalDateTime landingTime;
24
25     @XmlElement(name = "Duration")
26     private Integer duration;
27
28 > public FlightScheduleResponse() { ...
30
31 > public FlightScheduleResponse(Flight flight) { ...
36 }

```

Листинг 14 - Класа FlightScheduleResponse

Endpoint класе комуницирају са сервисима како би испунили пословну логику и обезбедили исправне одговоре на захтеве клијената. Сваки endpoint специјализован је за обраду једне врсте SOAP захтева. Endpoint класе парсирају SOAP захтеве и претварају их у инстанце класа SOAP захтева. Ове инстанце се након тога мапирају на модел и дају сервису за даљу обраду. На листингу 15 приказан је endpoint за резервисање карата.

```
37 @PayloadRoot(namespace = NAMESPACE_URI, localPart = "BookFlightRequest")
38 @ResponsePayload
39 public JAXBElement<BookFlightResponse> bookFlight(@RequestPayload Source request) throws TransformerException {
40     BookFlightRequest bookRequest = parseRequest(request);
41     String result = bookingService.book(bookRequest.getFlightCode(),
42         bookRequest.getFlightClass(), bookRequest.getCustomerEmail(),
43         bookRequest.mapToBookingDomain());
44
45     BookFlightResponse response = new BookFlightResponse(result);
46     QName qname = new QName(localPart:"BookFlightResponse");
47     JAXBElement<BookFlightResponse> jaxbResponse = new JAXBElement<BookFlightResponse>(qname,
48         declaredType:BookFlightResponse.class, response);
49
50     return jaxbResponse;
51 }
52
```

Листинг 15 - Endpoint BookFlightEndpoint

Конфигурациони слој је одговоран за управљање свим конфигурационим подешавањима система, као што су параметри базе података као и подешавање SOAP протокола. Овај слој обезбеђује централизовану локацију за сва конфигурациона подешавања, што олакшава управљање и одржавање система.

Подешавање SOAP протокола урађено је у класи SoapConfiguration (листинг 16). @EnableWs анотација означава да је ова класа конфигурација за подршку SOAP веб сервисима у Spring апликацији.

MessageDispatcherServlet је сервлет који прима све SOAP захтеве и управља њима. Конфигурише се као Spring Bean помоћу messageDispatcherServlet методе. Ова метода такође омогућава трансформацију локација WSDL докумената.

ServletRegistrationBean користи се за регистрацију сервлета MessageDispatcherServlet-a у Spring контексту. Дефинисано је да сервлет ослушкује захтеве на путањи /ws/*.

DefaultWsdll11Definition дефинише WSDL (Web Services Description Language) [43] за SOAP веб сервис. Овај Bean наводи назив порта, циљни namespace, локацију сервиса и XSD шему која описује структуру података која се користи у веб сервису.

XsdSchema представља XSD (XML Schema Definition) шему [44] која описује структуру података која се користи у SOAP веб сервису. У овом случају, шема се учитава из ресурса апликације.

```

14 @EnableWs
15 @Configuration
16 public class SoapConfiguration {
17
18     @Bean
19     public ServletRegistrationBean<MessageDispatcherServlet> messageDispatcherServlet(ApplicationContext context) {
20         MessageDispatcherServlet messageDispatcherServlet = new MessageDispatcherServlet();
21         messageDispatcherServlet.setApplicationContext(context);
22         messageDispatcherServlet.setTransformWsdlLocations(true);
23         return new ServletRegistrationBean<MessageDispatcherServlet>(messageDispatcherServlet, "/ws/*");
24     }
25
26     @Bean(name = "airline-company-2")
27     public DefaultWsd11Definition defaultWsd11Definition(XsdSchema airlineCompanySchema) {
28         DefaultWsd11Definition wsdl11Definition = new DefaultWsd11Definition();
29         wsdl11Definition.setPortTypeName("AirlineCompany2Port");
30         wsdl11Definition.setTargetNamespace("http://localhost:8082/");
31         wsdl11Definition.setLocationUri("/ws");
32         wsdl11Definition.setSchema(airlineCompanySchema);
33         return wsdl11Definition;
34     }
35
36     @Bean
37     public XsdSchema airlineCompanySchema() {
38         return new SimpleXsdSchema(new ClassPathResource(path:"xsd/airline-company-2.xsd"));
39     }
40 }
41

```

Листинг 16 - Конфигурација SoapConfiguration

У помоћном слоју налазе се ставри које нису директно везане за пословну логику, али су потребне сервисима. У овом слоју налази се логика за генерисање PDF документа из HTML шаблона, за слање мејлова.

5.3 Систем авио-компаније 3

У овом одељку предстаљена је имплементација трећег система авио-компаније и његови кључни делови. Овај систем омогућава администрацију летова и попушта, као и претрагу летова и резервацију карата за кориснике. У наставку је детаљни опис кључних делова система и технологија које су коришћене у имплементацији.

Систем авио-компаније 3 имплементиран је помоћу следећих технологија: C#, ASP .NET, Microsoft SQL Server и Postman [45] за тестирање GraphQL захтева.

Microsoft SQL Server је систем за управљање релационим базама података. У систему авио-компаније 3, Microsoft SQL Server се користи за складиштење свих релевантних података о аеродромима, летовима, картама, попустима, резервацијама. Сваки од ових ентитета се чува у одвојеним табелама у бази података.

ASP .NET је оквир за програмски језик C# који обезбеђује флексибилан начин за креирање и управљање различитим аспектима апликација. Ово омогућава дефинисање GraphQL крајњих тачака за различите функције система. C# и ASP.NET су коришћени за имплементацију серверског дела система авио-компаније 3, при чему је изграђен GraphQL API.

За размену података GraphQL API користи JSON формат.

Имплементација система авио-компаније 3 је подељена на неколико слојева како би се обезбедила боља организација кода. Ови слојеви су: модел, репозиторијум, сервис, слој крајњих тачака, конфигурација, помоћни слој.

Модел слој садржи дефиниције ентитета који се користе у систему, објекте за пренос података као и логику за мапирање између ентитета и објеката за пренос података. За

имплементацију овог система, у сврху објектно-релационог мапирања, употребљен је Entity Framework [46]. Ентитети који се чувају у бази података су следећи: аеродром (ентитет Airport), лет (ентитет Flight), карта (ентитет Ticket), попуст (ентитет Discount), резервација (ентитет Booking). На листингу 17 приказан је скраћени поглед на ентитет Booking. Модел Booking и његови атрибути дефинисани су у оквиру Entity Framework-a, користећи његове могућности за спровођење интегритета података и валидацију. Сваки ентитет има имплементирану методу за валидацију. На сличан начин дефинисани су сви остали ентитети овог система.

```

6      public class Booking : BaseEntity
7      {
8          [Required(ErrorMessage = "Validation: Name is required")]
9          public string Name { get; set; }
10
11         [Required(ErrorMessage = "Validation: Email name is required")]
12         [EmailAddress(ErrorMessage = "Validation: Email should be valid")]
13         public string Email { get; set; }
14
15         [Required(ErrorMessage = "Validation: Birth date is required")]
16         public DateTime BirthDate { get; set; }
17
18         [Required(ErrorMessage = "Validation: Passport is required")]
19         public string Passport { get; set; }
20
21         public Booking() : base()
22         {
23         }
24
25         public override void Validate()...
34     }

```

Листинг 17 - Модел Booking

Објекти за пренос података имају 2 улоге. Прва улога је енкапсулирање података које корисник шаље апликацији и који су потребни апликацији да обради захтев. Друга улога је енкапсулирање података које апликација шаље као одговор. Ове класе се директно мапирају на тело GraphQL захтева и одговора у JSON формату. На листингу 18 приказан је пример једног објекта за пренос података, BookingCreationDto.

```

3      public class BookingCreationDto
4      {
5          public string FlightSerialNumber { get; set; }
6          public string FlightClass { get; set; }
7          public string Email { get; set; }
8
9          public List<PassengerDto> Passengers { get; set; }
10     }

```

Листинг 18 - Објекат за пренос података BookingCreationDto

Логика за мапирање између ентитета и објекта за пренос података имплементирана је помоћу AutoMapper [47] пакета. На листингу 19 дат је пример мапирања између објекта за пренос података `DiscountCreationDto` и ентитета `Discount`.

```
14 private void CreateMaps()  
15 {  
16     CreateMap<DiscountCreationDto, Discount>();  
17 }
```

Листинг 19 - Мапирање

Репозиторијумски слој пружа апстракцију између модел слоја и сервис слоја. Репозиторијуми садрже логике за приступ подацима, као што су операције за креирање, читање, измену и брисање података. На листингу 20 приказан је генерички интерфејс `IRepository`. Он предвиђа методе за читање, креирање и измену ентитета. За читање ентитета омогућава и коришћење ламбда израза. Сваки репозиторијум у систему имплементира `IRepository` тако што наслеђује његову имплементацију `Repository`. Додатне методе које нису предвиђене генеричким интерфејсом се морају додати кроз нови интерфејс. На сличан начин имплементирани су сви остали репозиторијуми.

```
5 public interface IRepository<T> where T : class  
6 {  
7     4 references  
    List<T> GetAll(Expression<Func<T, bool>>? filter = null, string? includedProperties = null);  
8     7 references  
    T Get(Expression<Func<T, bool>> filter, string? includedProperties = null, bool orElseThrow = false);  
9     2 references  
    void Create(T entity);  
10    3 references  
    void Update(T entity);  
11 }  
12 }
```

Листинг 20 - Интерфејс `IRepository`

У репозиторијумском слоју се такође налази и `ServerDatabaseContext` која наслеђује класу `DbContext` из Entity Framework-a. `ServerDatabaseContext` садржи дефиниције табела које се чувају у бази података, као и типови веза између ентитета. Свака табела представљена је атрибутом типа `DbSet`.

Сервис слој садржи пословну логику апликације. Овај слој користи репозиторијуме за добијање и манипулацију подацима, позива мапирање за конверзију података и примењује пословна правила. За сваки сервис у овом систему постоји интерфејс и имплементација. У овом слоју се налази логика везана за све кључне функционалности овог система.

За креирање летова задужен је сервис `FlightService` и метода `Create` (листинг 21). Ова метода проверава податке из захтева тако што провери да ли у систему постоје аеродроми из релације лета, а осталу валидацију препушта валидационим правилима дефинисаним помоћу Entity Framework-a у моделу. Након тога генерише серијски број лета и позива методу `GenerateTickets` из сервиса `TicketService` који је задужен за креирање карата. Метода `GenerateTickets` за сваку класу летења креира тражени број карата са траженом ценом. Након успешног генерисања карата, карте се додају у лет и заједно са њим чувају у бази података.


```

28 public MessageDto Create(FlightCreationDto flightDto)
29 {
30     try
31     {
32         Flight flight = _mapper.Map<Flight>(flightDto);
33
34         Airport startingPoint = _airportRepository.Get(filter: a => a.Iata == flightDto.StartingPointIata, orElseThrow: true);
35         Airport endingPoint = _airportRepository.Get(filter: a => a.Iata == flightDto.EndingPointIata, orElseThrow: true);
36         flight.SetRelation(startingPoint, endingPoint);
37
38         flight.SerialNumber = GenerateSerialNumber(flight);
39
40         TicketPricing pricing = _mapper.Map<TicketPricing>(flightDto);
41         List<Ticket> tickets = _ticketService.GenerateTickets(flight.Id, flight.SerialNumber, pricing);
42         flight.AddTickets(tickets);
43
44         flight.Validate();
45         _flightRepository.Create(flight);
46
47         return new MessageDto("Flight created.");
48     }
49     catch (Exception ex)
50     {
51         return new MessageDto(ex.Message);
52     }
53 }

```

Листинг 21 - Метода Create сервиса FlightService

За претрагу летова је такође задужен сервис FlightService и његове методе Search (листинг 22) као и помоћне методе SearchByPassengerInformation, SearchByDateRange, SearchByRelation. Метода Search добавља све летове преко репозиторијума, позива остале 3 методе једну за другом и на крају мапира листу летова на листу објеката за пренос података. Остале 3 методе позивају помоћне методе, одређују да ли је параметар претраге валидан и да ли постоји и филтрирају листу летова помоћу лямбда израза.

```

55 public List<FlightDto> Search(FlightSearchDto searchFilter)
56 {
57     List<Flight> flights = _flightRepository.GetAll(includedProperties: "StartingPoint,EndingPoint,Tickets,Discounts");
58
59     if (flights.IsNullOrEmpty())
60     {
61         return new List<FlightDto>();
62     }
63
64     if(searchFilter != null)
65     {
66         flights = SearchBySerialNumber(flights, searchFilter.SerialNumber);
67         flights = SearchByDateRange(flights, searchFilter.From, searchFilter.To);
68         flights = SearchByRelation(flights, searchFilter.StartingPointIata, searchFilter.EndingPointIata);
69         flights = SearchByPassengerInformation(flights, searchFilter.FlightClass, searchFilter.PassengerCount);
70     }
71
72     List<FlightDto> flightDtos = _mapper.Map<List<FlightDto>>(flights);
73
74     return flightDtos;
75 }

```

Листинг 22 - Метода Search сервиса FlightService

За резервацију летова задужен је сервис BookingService и методе Book (листинг 23) и BookOne. Сервис врши валидацију захтева, и за сваког путника дефинисаног у захтеву покушава да резервише једну карту за тражени лет, у траженој класи летења. Након успешне резервације, на мејл купца шаљу се карте у PDF формату. PDF фајл генерише се попуњавањем HTML шаблона и његовом конверзијом у PDF формат употребом itext7.pdfhtml пакета [48]. Након конверзије, PDF се шаље на мејл купца уз попуњавање HTML шаблона за мејл поруку, везивање PDF фајла за поруку и слање поруке употребом SmtplibClient класе [49].

Остале функционалности које сервисни слој пружа јесте дефинисање попушта за летове и претрага аеродрома у систему.

```

25 public MessageDto Book(BookingCreationDto bookingDto)
26 {
27     try
28     {
29         List<Booking> bookings = _mapper.Map<List<Booking>>(bookingDto);
30
31         Flight flight = _flightRepository.Get(f => f.SerialNumber == bookingDto.FlightSerialNumber,
32         includedProperties: "StartingPoint,EndingPoint,Tickets,Discounts", orElseThrow: true);
33
34         FlightClass flightClass = ValidateClass(bookingDto.FlightClass);
35
36         List<String> ticketCodes = new List<String>();
37         foreach (Booking booking in bookings)
38         {
39             String ticketCode = BookOne(flight, flightClass, booking);
40             ticketCodes.Add(ticketCode);
41         }
42
43         _flightRepository.Update(flight);
44         PdfGenerator pdfGenerator = new PdfGenerator();
45         byte[] pdf = pdfGenerator.GenerateTicketsPDF(flight, bookingDto.FlightClass, bookings, ticketCodes);
46
47         _emailSender.SendTicketsEmailAsync(bookingDto.Email, pdf);
48
49         return new MessageDto("Booking created.");
50     }
51     catch (Exception ex)
52     {
53         return new MessageDto(ex.Message);
54     }
55 }

```

Листинг 23 - Метода Book сервиса BookingService

Слој крајњих тачака (endpoint слој) је одговоран за руковање GraphQL захтевима и одговорима у систему. У овом слоју се налазе дефиниције GraphQL упита, мутација и типова података, као и класе које обрађују долазне захтеве и враћају одговоре. Endpoint класе комуницирају са сервисима како би испунили пословну логику и обезбедили исправне одговоре на захтеве клијената. Овај слој обезбеђује интерфејс за клијенте који користе GraphQL за комуникацију са системом.

GraphQL упити, мутације и типови података дефинисани су употребом HotChocolate пакета [50]. GraphQL упити, мутације и типови података су кључни елементи у GraphQL API-ју који омогућавају ефикасну комуникацију и манипулацију подацима у систему.

GraphQL упити представљају начин на који клијенти захтевају податке из система. Упити су дизајнирани тако да клијенти могу специфицирати тачно које податке желе да добију, што омогућава прецизну и оптимизовану размену података. На пример, клијент може затражити детаље о одређеном лету користећи упит који је енкапсулиран класом FlightQuery (листинг 24). Овом класом описана је структура упита, аргументи упита (FlightSearchType), endpoint класа одговорна за позивање сервисног слоја (FlightResolver) као и структура одговора који се очекује (ListType<NonNullType<FlightType>>).

```

5 public class FlightQuery
6 {
7     1 reference
8     public static void Configure(IObjectTypeDescriptor descriptor)
9     {
10         descriptor.Field("GetFlights")
11             .Argument("filter", a => a.Type<FlightSearchType>().DefaultValue(null))
12             .ResolveWith<FlightResolver>(r => r.GetFlights(default, default))
13             .Type<ListType<NonNullType<FlightType>>>();
14     }
15 }

```

Листинг 24 - Упит FlightQuery

GraphQL мутације омогућавају клијентима да врше промене у подацима у систему. Кроз мутације, клијенти могу креирати, ажурирати или брисати податке. На пример, корисник може користити мутацију да резервише лет користећи мутацију која је енкапсулирана класом `BookingMutation` (листинг 25). Овом класом описана је структура мутације, аргументи мутације (`NonNullType<BookingType>`), endpoint класа одговорна за позивање сервисног слоја (`BookingResolver`) као и структура одговора који се очекује (`MessageType`).

```
6 public class BookingMutation
7 {
8     1 reference
9     public static void Configure(IObjectTypeDescriptor descriptor)
10    {
11        descriptor.Field("BookFlight")
12            .Argument("input", a => a.Type<NonNullType<BookingType>>())
13            .ResolveWith<BookingResolver>(r => r.BookFlight(default, default))
14            .Type<MessageType>();
15    }
16 }
```

Листинг 25 - Мутација `FlightQuery`

GraphQL типови података дефинишу структуру података који се размењују у систему. Типови података укључују примитивне типове као што су `String`, `Int`, и `Boolean`, као и сложене типове као што су објекти и ентитети специфични за домен апликације, као што су резервације летова (листинг 26). Ове класе се директно мапирају на објекте за пренос података и њихове атрибуте.

```
5 public class BookingType : InputObjectType<BookingCreationDto>
6 {
7     0 references
8     protected override void Configure(IInputObjectTypeDescriptor<BookingCreationDto> descriptor)
9     {
10        descriptor.Name("Booking");
11        descriptor.Field(f => f.FlightSerialNumber).Type<NonNullType<StringType>>();
12        descriptor.Field(f => f.FlightClass).Type<NonNullType<StringType>>();
13        descriptor.Field(f => f.Email).Type<NonNullType<StringType>>();
14        descriptor.Field(f => f.Passengers).Type<NonNullType<ListType<PassengerType>>>();
15    }
16 }
```

Листинг 26 - Тип података `BookingType`

Класе које обрађују долазне захтеве и враћају одговоре су кључни део GraphQL API-ја који је одговоран за обраду упита и мутација. Оне мапирају GraphQL операције на пословну логику позивом метода сервисног слоја. На листингу 27 дат је пример класе која обрађује упите и мутације везане за летове.

```

6      public class FlightResolver
7      {
8          1 reference
9          public MessageDto CreateFlight(FlightCreationDto input,
10             [Service(ServiceKind.Synchronized)] IFlightService flightService)
11          {
12              MessageDto Message = flightService.Create(input);
13              return Message;
14          }
15
16          1 reference
17          public List<FlightDto> GetFlights(FlightSearchDto filter,
18             [Service(ServiceKind.Synchronized)] IFlightService flightService)
19          {
20              List<FlightDto> flights = flightService.Search(filter);
21              return flights;
22          }
23      }

```

Листинг 27 - Тип података **BookingType**

У помоћном слоју налазе се модули који нису директно везани за пословну логику, али су потребни сервисима.

5.4 Систем за претрагу аеродрома

У овом одељку представљена је имплементација система за претрагу аеродрома и његови кључни делови. Овај систем омогућава добављање информација о аеродромима као и њихову претрагу. У наставку је детаљни опис кључних делова система и технологија које су коришћене у имплементацији.

Систем за претрагу аеродрома имплементиран је уз помоћ програмског језика Go. За тестирање REST захтева коришћен је Postman.

За складиштење података систем користи CSV фајл систем. Сви подаци су статички и налазе се у CSV фајловима.

Go програмски језик је коришћен за имплементацију серверског дела система авио-компаније 3, при чему је изграђен REST API.

За размену података REST API користи CSV формат.

Имплементација система за претрагу аеродрома је подељена на неколико слојева како би се обезбедила боља организација кода. Ови слојеви су: модел, репозиторијум, сервис, контролер.

Модел слој садржи дефиниције ентитета који се користе у систему. У систему постоји само један ентитет, аеродром (ентитет Airport). На листингу 28 приказан је поглед на ентитет Airport. Он садржи следеће информације о аеродромима: назив, IATA код, географска ширина, географска дужина, надморска висина, континент, држава, регион, општина.

```

1  package model
2
3  type Airport struct {
4      Name      string
5      IATA      string
6      Latitude  string
7      Longitude string
8      Elevation string
9      Continent string
10     Country   string
11     Region    string
12     Municipality string
13 }
14

```

Листинг 28 - Модел **Airport**

Репозиторијумски слој пружа апстракцију између модел слоја и сервис слоја. Репозиторијуми садрже логику за приступ подацима. Улога репозиторијума у овом систему је да прочита CSV фајл који садржи информације о аеродромима и да те информације претвори у инстанце модела `Airport` и врати их као повратну вредност.

Сервис слој садржи пословну логику апликације. Овај слој користи репозиторијуме за добијање и манипулацију подацима, позива мапирање за конверзију података и примењује пословна правила. У овом слоју се налази логика везана за претрагу летова. Метода `GetAirports` позива репозиторијумски слој и филтрира листу аеродрома на основу параметра уколико тај параметар није празан (листинг 29).

```

21 func (s *airportService) GetAirports(search string) ([]model.Airport, error) {
22     if search == "" {
23         return s.repo.FindAll()
24     }
25
26     search = strings.ToLower(search)
27     airports, err := s.repo.FindAll()
28     if err != nil {
29         return nil, err
30     }
31
32     var result []model.Airport
33     for _, airport := range airports {
34         if strings.Contains(strings.ToLower(airport.Name), search) || strings.Contains(strings.ToLower(airport.IATA), search) {
35             result = append(result, airport)
36         }
37     }
38
39     return result, nil
40 }

```

Листинг 29 - Метода `GetAirports` сервиса **AirportService**

Контролер слој је задужен за обраду HTTP захтева и враћање одговора клијенту. Контролери користе сервисе за обављање пословне логике и формирање одговора. У овом систему контролер је задужен и за серијализацију објеката `Airport` класе у одговарајући CSV формат.

5.5 Систем за претрагу летова са председањем

У овом одељку представљена је имплементација система за претрагу летова са председањем и његови кључни делови. Овај систем омогућава претрагу летова и проналажење путања за долазак од полазног аеродрома до аеродрома дестинације директним летовима или летовима за председање. У наставку је детаљни опис кључних делова система и технологија које су коришћене у имплементацији.

Систем за претрагу летова са преседањем имплементиран је помоћу следећих технологија: Python [51], Flask [52]. За тестирање REST захтева коришћен је Postman.

Flask је радни оквир за Python програмски језик који пружа флексибилан начин за креирање и управљање HTTP рутама. Ово омогућава дефинисање REST API крајњих тачака за различите функције система. Python и Flask су коришћени за имплементацију серверског дела система за претрагу летова са преседањем, при чему је изграђен REST API.

За размену података REST API користи JSON формат.

Имплементација система за претрагу летова са преседањем своди се на имплементацију BFS (Breadth-First Search) алгоритма [53] за претрагу графова и стабала.

BFS (Breadth-First Search) је алгоритам за претраживање или обилазак графова и стабала који почиње од коренског чвора и истражује све његове суседне чворове на тренутном нивоу пре него што пређе на чворове на следећем нивоу. BFS користи ред за праћење чворова које треба истражити. Када посети чвор, BFS додаје све његове непосредне суседе у ред, осигуравајући да се сви чворови на једном нивоу истраже пре него што пређе на следећи ниво.

Систем као улазни параметар прима листу летова са минимално информацијама о IATA коду полазног аеродрома, IATA коду аеродрома дестинације, времену поласка, времену доласка и цени. Као остале улазне параметре прима жељени датум поласка, жељени полазни аеродром и жељени аеродром дестинације. Након парсирања покреће се алгоритам за претрагу летова.

Функција `build_graph` конструише граф од датих података о летовима где је сваки аеродром чвор, а сваки лет је усмерена грана између два чвора (аеродрома). Гране садрже информације о времену поласка и доласка летова. Ова метода приказана је на листингу 30.

```
10 def build_graph(flights):
11     graph = defaultdict(list)
12     for flight in flights:
13         departure_time = datetime.fromisoformat(flight['departureTimestamp'].replace('Z', '+00:00'))
14         arrival_time = datetime.fromisoformat(flight['arrivalTimestamp'].replace('Z', '+00:00'))
15         graph[flight['fromAirportIata']].append((flight, departure_time, arrival_time))
16     return graph
```

Листинг 30 - Функција `build_graph`

Функција `find_flights_with_stops` (листинг 31) спроводи BFS алгоритам да истражи све могуће путање летова од полазног аеродрома до аеродрома дестинације. Започиње од полазног аеродрома и истражује све доступне аеродроме, водећи рачуна о пређеном путу, укупној цени и укупном трајању. Користи ред за истраживање сваког аеродрома ниво по ниво. За сваки аеродром разматра све могуће излазне летове. За сваки лет проверава да ли датум поласка лета одговара датом датуму и да ли је време преседања између претходног доласка и тренутног поласка унутар важећег опсега (од 30 минута до 4 сата). Ако важећа путања стигне до аеродрома дестинације, додаје се на листу путања.

```

18 def find_flights_with_stops(flights, date, from_iata, to_iata):
19     try:
20         date = datetime.fromisoformat(date.replace('Z', '+00:00'))
21     except ValueError as e:
22         logging.error(f"Invalid date format: {date} - Error: {e}")
23         return []
24
25     graph = build_graph(flights)
26     paths = []
27     queue = deque([(from_iata, [], datetime.min, 0, 0)])
28
29     while queue:
30         current_airport, path, prev_arrival, total_price, total_duration = queue.popleft()
31         for flight, departure_time, arrival_time in graph[current_airport]:
32             if departure_time.date() != date.date():
33                 continue
34             if prev_arrival != datetime.min and (departure_time - prev_arrival < timedelta(minutes=30)
35             or departure_time - prev_arrival > timedelta(hours=4)):
36                 continue
37             new_path = path + [flight]
38             new_price = total_price + flight['price']
39             new_duration = total_duration + flight['duration']
40             if flight['toAirportIata'] == to_iata:
41                 paths.append((new_path, new_price, new_duration))
42             else:
43                 queue.append((flight['toAirportIata'], new_path, arrival_time, new_price, new_duration))
44
45     return paths

```

Листинг 31 - Функција `find_flights_with_stops`

Након прикупљања свих могућих путања, систем филтрира и сортира важеће путање на основу цене (најнижа прва), броја летова у путањи (најмањи прво) и укупног трајања лета (најкраће прво).

5.6 Систем за претрагу летова

У овом одељку представљена је имплементација система за претрагу летова и његови кључни делови. Овај систем спаја све претходно описане системе и омогућава претрагу директних летова и летова са преседањем, једносмерних и повратних, као и резервацију карата. Систем комуницира са системима авио-компанија и проналази летове користећи податке из све 3 авио-компаније, а са системом за претрагу летова са преседањем комуницира за налажење путања за долазак од полазног аеродрома до аеродрома дестинације директним летовима или летовима за преседање. У наставку је детаљни опис кључних делова система и технологија које су коришћене у имплементацији.

Систем за претрагу летова са преседањем имплементиран је уз помоћ следећих технологија: Mulesoft Anypoint Studio, DataWeave [54]. За тестирање REST захтева коришћен је Postman.

Mulesoft Anypoint Studio је развојно окружење (IDE) које омогућава корисницима да креирају и тестирају API-је и интегришу их у своје апликације. Развијен од стране Mulesoft-a, овај алат нуди графички интерфејс који поједностављује процес интеграције тако што омогућава корисницима да визуелно дизајнирају протоке података и API-је, без потребе за кодирањем.

Anypoint Studio користи DataWeave, језик за трансформацију података, који олакшава манипулацију и конверзију података из једног формата у други. Овај алат је користан за развој интеграционих решења која повезују различите системе и апликације, које користе различите формате података.

Anypoint Studio нуди палету компоненти [55] које се слажу у токове података у циљу интеграције система и имплементације пословне логике. Компоненте које су коришћене у овом систему су следеће:

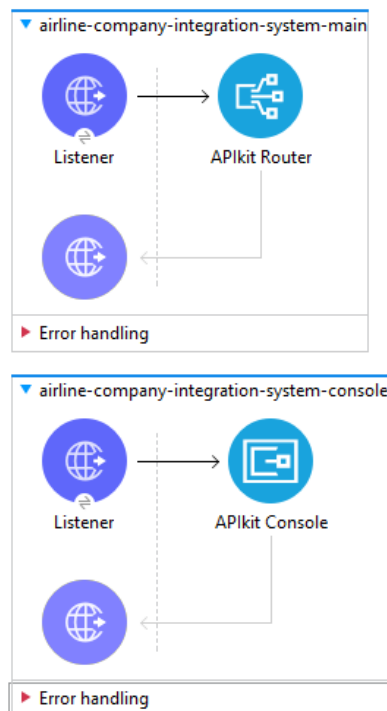
- Flow Reference – позива други ток података. Може се упоредити са позивом функције или методе.
- Logger – логује податке.
- Transform Message – трансформише податке из једног облика други уз опциону употребу DataWeave.
- Error Handler – обрађује грешке. Може се упоредити са catch блоком у неким програмским језицима.
- On Error Continue – обрађује грешку и враћа се у ток података који је позвао ток у ком је настала грешка.
- On Error Propagate – прослеђује грешку току података који је позвао ток у ком је настала грешка. Уколико тај ток података нема Error Handler који може да обради ту врсту грешке, грешка се шаље даље.
- Choice – Може се упоредити са if и else блоковима. На основу услова извршава један о 2 могућа тока података.
- Scatter-Gather – Извршава више паралелних токова података и групише њихов резултат.
- Async – ток података унутар Async блока се извршава асинхроно.
- Parallel For Each – паралелна for петља. Итерира кроз колекцију и за сваки елемент извршава ток података.
- Set Payload – подешава вредност payload дела Mule поруке. Payload је главно поље Mule поруке које служи за чување информација кроз више токова података, а моће да служи и као радна површина за неке од наведених компоненти.
- Set Variable – подешава вредност променљиве унутар Mule поруке. Mule променљиве се не чувају кроз токове података и локалне су.
- Listener – ослушкује HTTP захтеве.
- Request – шаље HTTP захтев и чека одговор.

Систем за претрагу летова имплементиран је користећи ESB (Enterprise Service Bus) шаблон. Enterprise Service Bus је софтверска архитектура која омогућава комуникацију и интеракцију између различитих апликација. Он функционише као посредник који омогућава повезивање и интеграцију различитих система и сервиса, без обзира на њихову платформу, технологију, коришћене протоколе или формате података. ESB олакшава управљање подацима, трансформацију порука, рутирање, и оркестрацију сервиса, што доприноси бољој флексибилности, скалабилности и одрживости система. Користећи ESB, организације могу лакше интегрисати нове апликације и сервисе у постојећу инфраструктуру, смањујући трошкове и сложеност интеграције.

Системи које систем за претрагу летова повезује, као и њихове карактеристике и разлике су следећи:

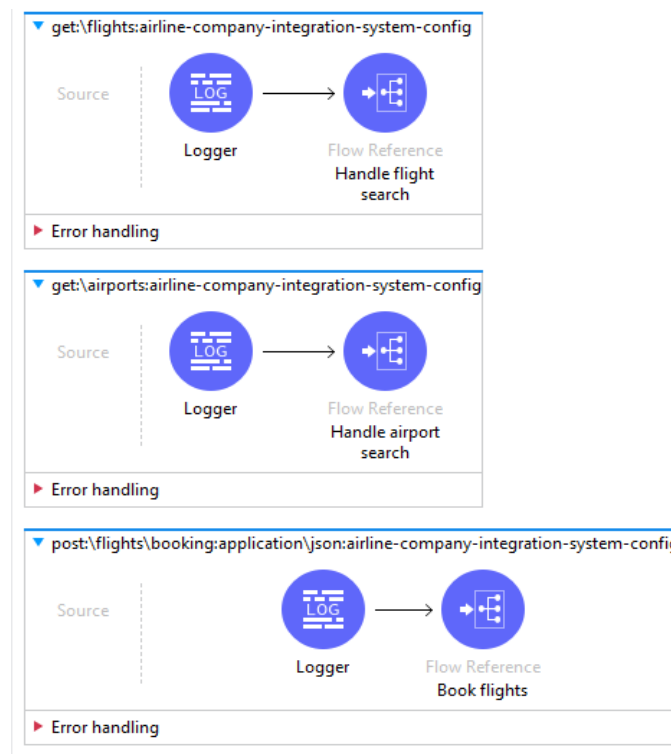
- Систем авио-компаније 1 који пружа REST API и комуницира преко JSON формата порука. Цена лета је изражена у еврима.
- Систем авио-компаније 2 који пружа SOAP API и комуницира преко SOAP Envelope XML формата порука. Такође сами подаци имају другачију структуру угњеждавања у односу на податке из авио-компаније 1, као и другачије називе и формате атрибута. Цена лета је изражена у динарима.
- Систем авио-компаније 3 који пружа GraphQL API и комуницира преко JSON формата порука. Такође сами подаци имају другачију структуру угњеждавања у односу на податке из авио-компаније 1 и авио-компаније 2, као и другачије називе и формате атрибута. Цена лета је изражена у америчким доларима.
- Систем за претрагу аеродрома који пружа REST API и враћа податке у CSV формату.
- Систем за претрагу летова са преседањем који пружа REST API и комуницира преко JSON формата порука.
- Exchange-Rate API дизајниран од стране AYR Tech (Pty) Ltd [56]. API нуди функционалности за конверзију валута. У питању је REST API који комуницира преко JSON формата порука.

Као полазна тачка имплементације дефинисан је RAML [57] фајл. RAML (RESTful API Modeling Language) је језик за моделовање API-ја који омогућава дефиницију и документацију RESTful API-је на структуриран начин. RAML дефинише структуру и понашање API-ја, укључујући руте, параметре, захтеве, одговоре и примере одговора. Уз помоћ RAML-а, искоришћена је Mulesoft Anypoint Studio функционалност која аутоматски генерише код и конфигурације потребне за имплементацију API-ја. На овај начин генерисан је API контролер (слика 13). Компонента Listener ослушкује захтеве док APIkit Router мапира захтеве на одговарајуће токове података.



Слика 13 - API контролер

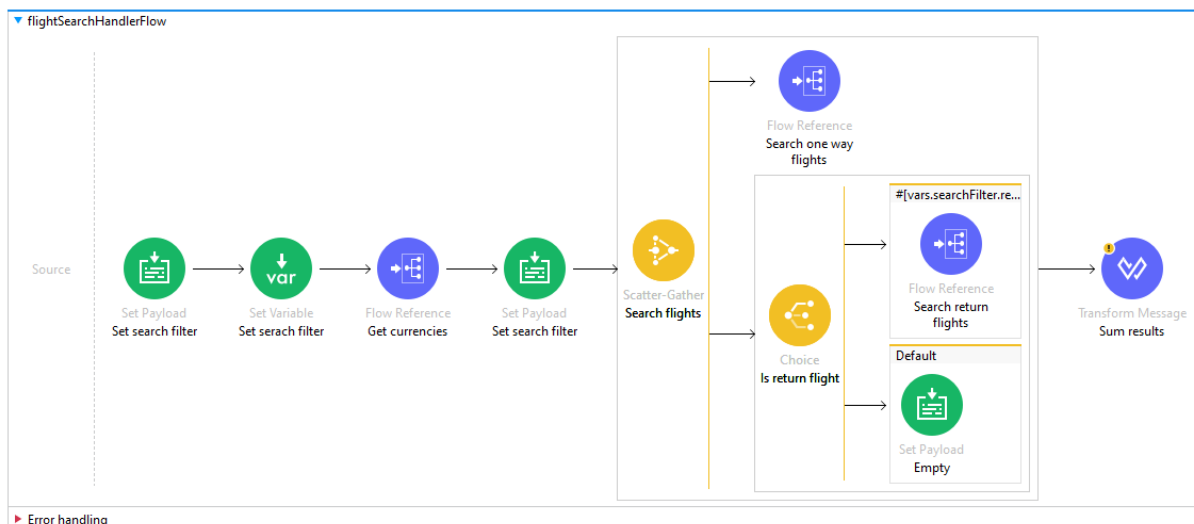
API нуди 3 крајње тачке: претрага летова, резервација летова и претрага аеродрома (потребна приликом претраге летова). Токови података за ове 3 крајње тачке су аутоматски генерисани и приказани на слици 14. Крајње тачке логују захтев и позивају ток података који је задужен за одговарајућу пословну логику.



Слика 14 - Крајње тачке

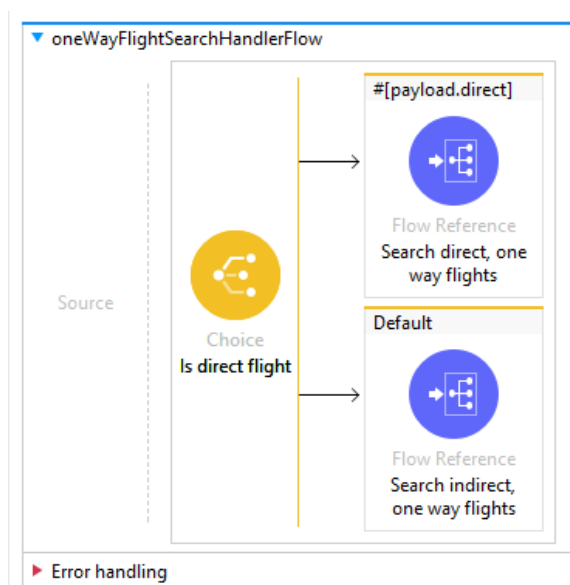
Претрага летова је најсложенија функционалност система. У зависности од параметара претраге могуће је претраживати директне летове и летове са преседањем. Такође постоји и опција претраге једносмерних или повратних летова. Потребно је прво одредити шта је кориснику потребно.

Главни ток података ове функционалности је `flightSearchHandlerFlow`. Његов задатак је обрада параметара претраге, добављање података о валутама и преусмеравање на један од два тока података у зависности од тога да ли се траже једносмерни или повратни летови. Након извршавања једног од 2 тока података форматира добијене листе летова. Ово је приказано на слици 15.



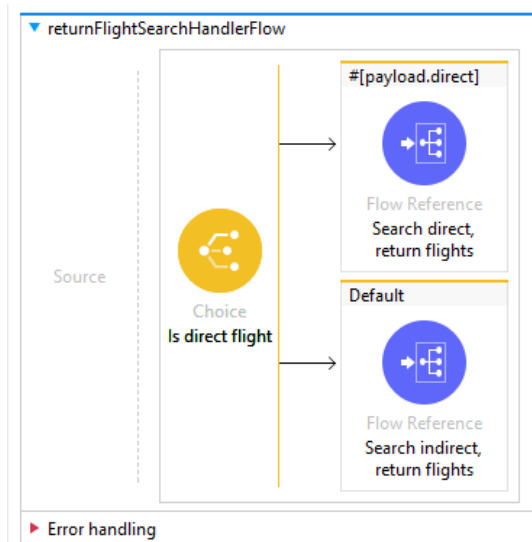
Слика 15 - Ток података `flightSearchHandlerFlow`

У случају да су у питању једносмерни летови, позива се ток података `oneWayFlightSearchHandlerFlow`. Његова улога је преусмеравање на један од два тока података у зависности од тога да су тражени искључиво директни летови или и летови са преседањем. На слици 16 приказан је ток података `oneWayFlightSearchHandlerFlow`.



Слика 16 - Ток података `oneWayFlightSearchHandlerFlow`

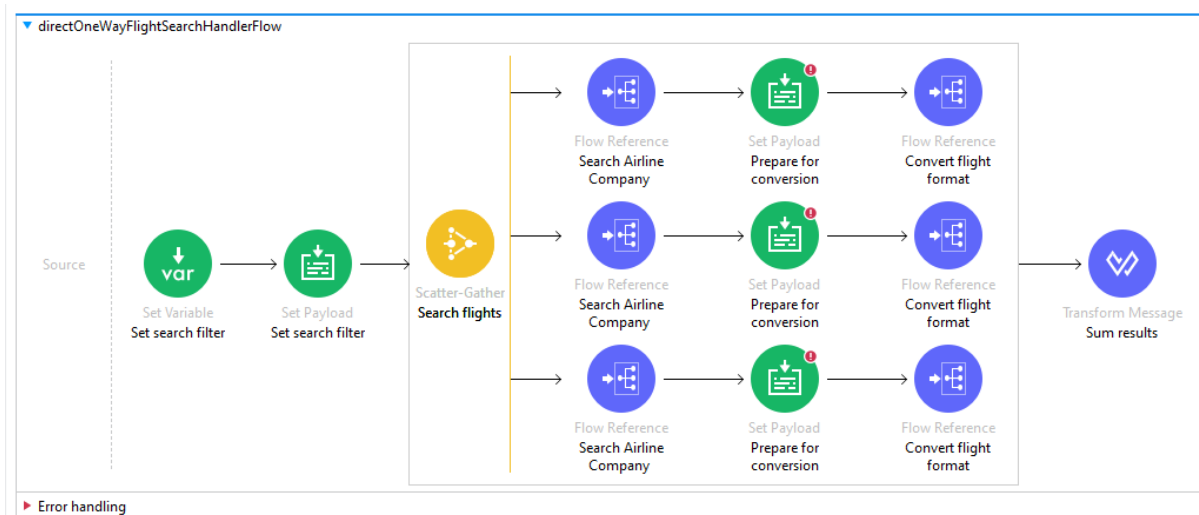
Ток података `returnFlightSearchHandlerFlow` има исту улогу у случају претраге повратних летова и позива одговарајуће токове података за тај случај (слика 17).



Слика 17 - Ток података **returnFlightSearchHandlerFlow**

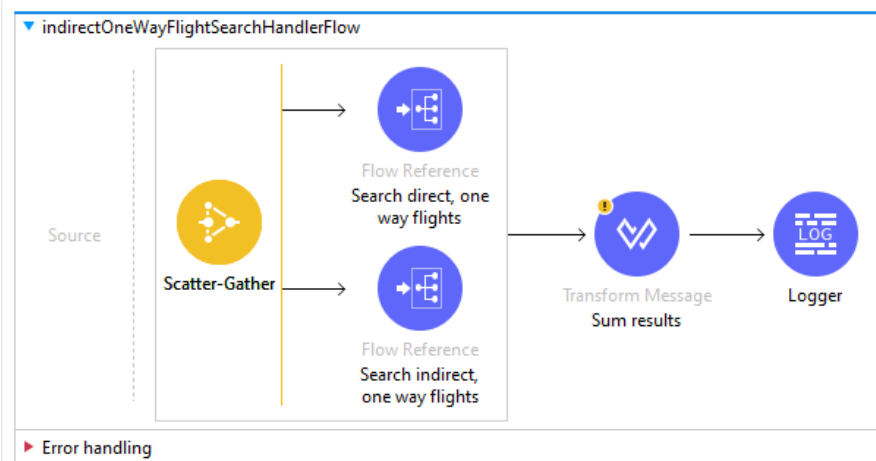
Ток података `directOneWayFlightSearchHandlerFlow` (слика 18) позива се за тражење директних, једносмерних летова. Ток припрема `payload` део Mule поруке и у њега смешта критеријуме претраге. Након тога помоћу компоненте `Scatter-Gather` позива токове података за добављање летова из авио-компаније након чега се позива ток података за конверзију формата добијених летова у јединствен JSON формат. Резултати се сумирају употребом `Transform Message` компоненте.

Слично функционише и ток података `directReturnFlightSearchHandlerFlow` који служи за тражење директних, повратних летова. Једина разлика је у прослеђеним критеријумима претраге који одговарају повратном лету.



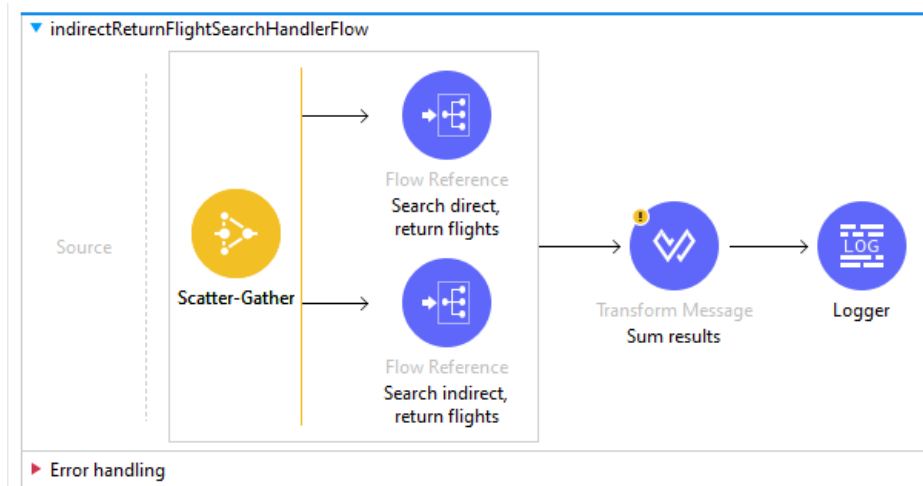
Слика 18 - Ток података **directOneWayFlightSearchHandlerFlow**

Ток података `indirectOneWayFlightSearchHandlerFlow` (слика 19) позива се за тражење једносмерних летова, директних и са преседањем. Ток помоћу компоненте `Scatter-Gather` позива претходно описани ток података за добављање директних, једносмерних летова као и ток за добављање једносмерних летова са преседањем. Резултати се сумирају употребом `Transform Message` компоненте.



Слика 19 - Ток података **indirectOneWayFlightSearchHandlerFlow**

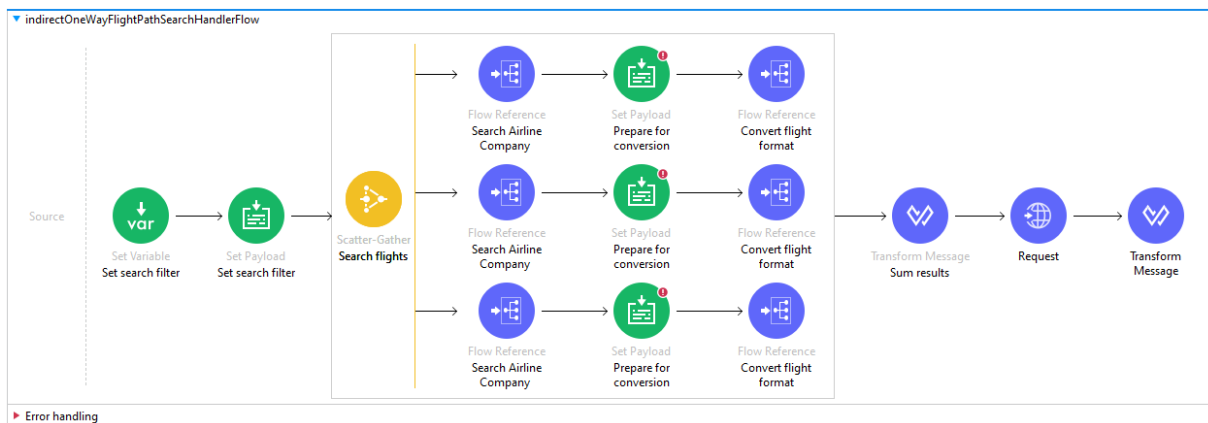
Ток података **indirectReturnFlightSearchHandlerFlow** има исту улогу у случају претраге повратних летова, директних и са преседањем. Позива одговарајуће токове података за тај случај (слика 20).



Слика 20 - Ток података **indirectReturnFlightSearchHandlerFlow**

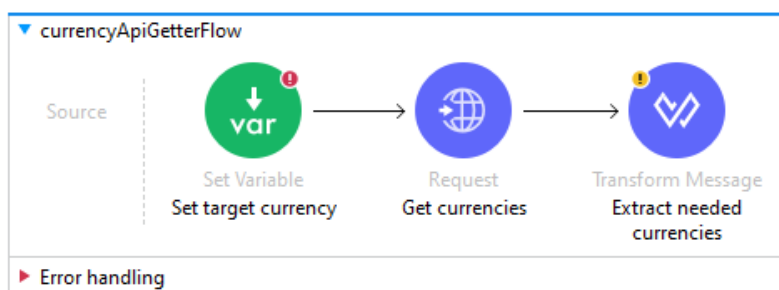
Ток података **indirectOneWayFlightPathSearchHandlerFlow** (слика 21) позива се за тражење једносмерних летова са преседањем. Ток припрема payload део Mule поруке и у њега смешта критеријуме претраге. Након тога помоћу компоненте Scatter-Gather позива токове података за добављање летова из авио-компаније након чега се позива ток података за конверзију формата добијених летова у јединствен JSON формат. Резултати се сумирају употребом Transform Message компоненте. Након овога летови се прослеђују систему за претрагу летова са преседањем.

Слично ради ток података **indirectReturnFlightPathSearchHandlerFlow** који служи за тражење повратних, летова са преседањем. Једина разлика је у прослеђеним критеријумима претраге који одговарају повратном лету.



Слика 21 - Ток података `indirectOneWayFlightPathSearchHandlerFlow`

Ток података за добављање информација о валутама (слика 22) као улазни параметар узима тражену валуту. Шаље се захтев ка Exchange Rates API-ју и у компоненти Transform Message се издвајају информације о потребним валутама.



Слика 22 - Ток података `currencyApiGetterFlow`

Токови података за добављање летова као улазни параметар узимају критеријуме претраге. Састоје се из једне Transform Message компоненте која је задужена за писање тела захтева и једне Request компоненте која је задужена за слање захтева ка системима авио-компаније. Приликом формирања тела захтева коришћен је DataWeave.

Формирање REST тела захтева за авио-компанију 1 је приказано на листингу 32. Заглавље `application/json` назначава да је излазна вредност компоненте JSON. Вредности се преузимају из `payload-a`.

```
%dwr 2.0
output application/json
---
{
  direct: payload.direct,
  from: if (payload.return) payload.payload.returnFrom else payload.payload.from,
  to: if (payload.return) payload.payload.returnTo else payload.payload.to,
  origin: if (payload.return) upper(payload.payload.toAirportIATA) else upper(payload.payload.fromAirportIATA),
  destination: if (payload.return) upper(payload.payload.fromAirportIATA) else upper(payload.payload.toAirportIATA),
  flightClass: payload.payload.flightClass,
  passengerCount: (payload.payload.adultPassengerCount as Number) + (payload.payload.childPassengerCount as Number)
}
```

Листинг 32 - Формирање тела захтева за претрагу летова авио-компаније 1

Формирање SOAP Envelope XML тела захтева за авио-компанију 2 је приказано на листингу 33. Заглавље `text/xml` назначава да је излазна вредност компоненте XML. Вредности се преузимају из `payload-a`.

```
%dw 2.0
output text/xml
ns soapenv http://schemas.xmlsoap.org/soap/envelope/
ns loc http://localhost:8082/
---
soapenv#Envelope: {
  soapenv#Header: {},
  soapenv#Body: {
    loc#SearchFlightsRequest: {
      loc#From: if (payload.return) payload.payload.returnFrom else payload.payload.from,
      loc#To: if (payload.return) payload.payload.returnTo else payload.payload.to,
      loc#DepartureAirportIATA: if(payload.direct) if (payload.return) upper(payload.payload.toAirportIATA)
        else upper(payload.payload.fromAirportIATA) else '',
      loc#ArrivalAirportIATA: if(payload.direct) if (payload.return) upper(payload.payload.fromAirportIATA)
        else upper(payload.payload.toAirportIATA) else '',
      loc#FlightClass: payload.payload.flightClass,
      loc#PassengerCount: (payload.payload.adultPassengerCount as Number) + (payload.payload.childPassengerCount as Number)
    }
  }
}
```

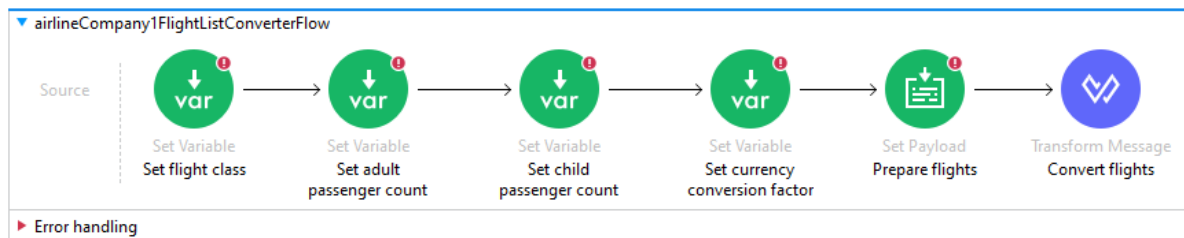
Листинг 33 - Формирање тела захтева за претрагу летова авио-компаније 2

Формирање GraphQL тела захтева за авио-компанију 3 је приказано на листингу 34. Заглавље application/json назначава да је излазна вредност компоненте JSON. Вредности се преузимају из payload-a.

```
%dw 2.0
output application/json
var from = if (payload.return) payload.payload.returnFrom else payload.payload.from as String default ""
var to = if (payload.return) payload.payload.returnTo else payload.payload.to as String default ""
var fromAirportIATA = if(payload.direct) if (payload.return) lower(payload.payload.toAirportIATA)
  else lower(payload.payload.fromAirportIATA) as String default "" else ""
var arrivalAirportIATA = if(payload.direct) if (payload.return) lower(payload.payload.fromAirportIATA)
  else lower(payload.payload.toAirportIATA) as String default "" else ""
var flightClass = payload.payload.flightClass as String default ""
var adultPassengerCount = payload.payload.adultPassengerCount as Number default 0
var childPassengerCount = payload.payload.childPassengerCount as Number default 0
var passengerCount = (adultPassengerCount + childPassengerCount) as String
---
{
  query: "query GetFlights {
    GetFlights(
      filter: {
        from: \"\" ++ from ++ \"\",
        to: \"\" ++ to ++ \"\",
        startingPointIata: \"\" ++ lower(fromAirportIATA) ++ \"\",
        endingPointIata: \"\" ++ lower(arrivalAirportIATA) ++ \"\",
        flightClass: \"\" ++ flightClass ++ \"\",
        passengerCount: \"\" ++ passengerCount ++ \"\"
      }
    ){
      flightInformation {
        serialNumber
        scheduledDeparture
        scheduledArrival
        travelTime
        baggageGuidelines
        startingPointIata
        endingPointIata
        startingPointName
        endingPointName
      }
      flightPrice {
        economyCount
        economyPrice
        businessCount
        businessPrice
        firstCount
        firstPrice
        kidsDiscountPercentage
        discountPercentage
      }
    }
  }"
}
```

Листинг 34 - Формирање тела захтева за претрагу летова авио-компаније 2

Токови података за конверзију формата летова као улазни параметар узимају листу летова и фактор множења за тражену валуту. Главна компонента је једна Transform Message компонента која је задужена за Dataweave конверзију летова из формата добијеног из одговора авио-компанија у униформни JSON формат. Осим овога, за тражени број путника и њихове узрасте обрачунава укупну цену лета за задату валуту. Ток података за конверзију летова авио-компаније 1 приказан је на слици 23. Токови података за остале 2 авио-компаније су слични са разликама у DataWeave изразима које проистичу из разлика у формату података летова. Након конверзије, добављени летови из све 3 авио-компаније имају један, униформни JSON формат што омогућава лакше руковање подацима о летовима.



Слика 23 - Ток података `airlineCompany1FlightListConverterFlow`

Интеграција са системом за претрагу летова са председањем није извршена у посебном току података него употребом једне Request компоненте. У оквиру ове компоненте употребом DataWeave се формира тело захтева које садржи летове у униформном JSON формату и информације о критеријумима претраге. DataWeave израз за формирање тела овог захтева приказан је на листингу 35.

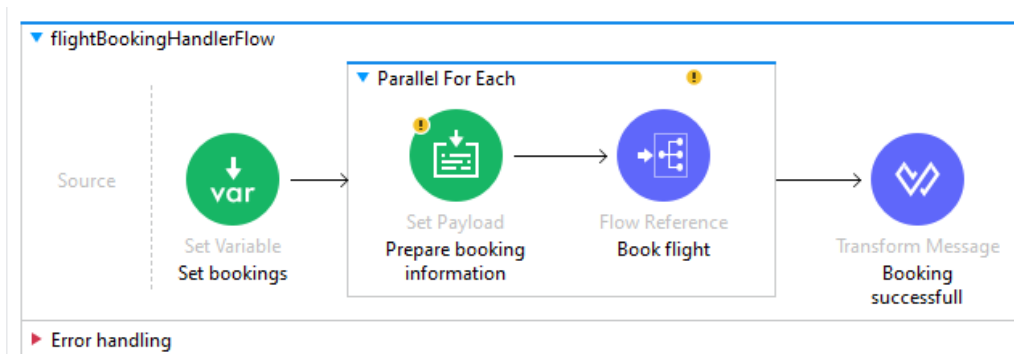
```

%dw 2.0
output application/json
---
{
  flights: payload.flights,
  fromIata: upper(vars.searchFilter.toAirportIATA),
  toIata: upper(vars.searchFilter.fromAirportIATA),
  date: vars.searchFilter.returnFrom
}

```

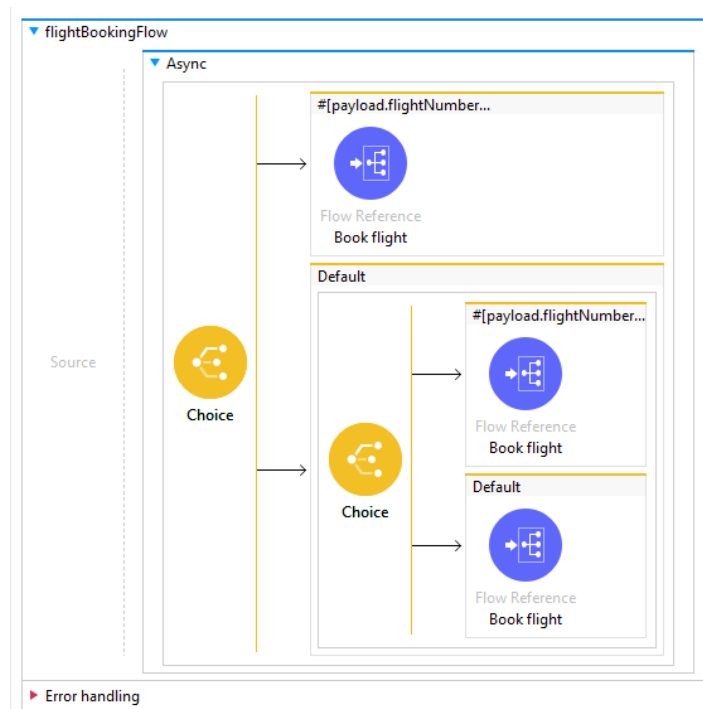
Листинг 35 - Формирање тела захтева за претрагу летова са председањем

Резервација летова такође спаја више система. Главни ток података ове функционалности је `flightBookingHandlerFlow` (слика 24). Његов задатак је појединачна обрада захтева за резервацију летова. Припрема листу резервација и итерира кроз њу у оквиру Parallel For Each компоненте која позива ток података `flightBookingFlow`.



Слика 24 - Ток података `flightBookingHandlerFlow`

Ток података `flightBookingFlow` (слика 25) за сваку резервацију одређује којој авио-компанији припада тражени лет у оквиру две угњеждане `Choice` компоненте. На основу тога захтев се шаље оном току података који одговара датој компанији ради резервације карата.



Слика 25 - Ток података `flightBookingFlow`

Токови података за добављање летова као улазни параметар узимају информације о резервацији. Састоје се из једне `Transform Message` компоненте која је задужена за писање тела захтева и једне `Request` компоненте која је задужена за слање захтева ка системима авио-компаније. Приликом формирања тела захтева коришћен је `DataWeave` на сличан начин као и приликом претраге летова. Ови токови података имају и дефинисан `Error Handler` и `On Error Continue` обрађивач грешке који грешку логује и исписује у конзолу.

Формирање `REST` тела захтева за авио-компанију 1 је приказано на листингу 36. Заглавље `application/json` назначава да је излазна вредност компоненте `JSON`. Вредности се преузимају из `payload-a`.

```
%dw 2.0
output application/json
import * from dw::core::Dates
---
{
    flightNumber: payload.flightNumber,
    flightClass: payload.flightClass,
    email: payload.email,
    passengers: payload.passengers map ( passenger , indexOfPassenger ) -> {
        passportNumber: passenger.passportNumber,
        firstName: passenger.firstName,
        lastName: passenger.lastName,
        dateOfBirth: passenger.birthDate
    }
}
```

Листинг 36 - Формирање тела захтева за резервацију лета авио-компаније 1

Формирање SOAP Envelope XML тела захтева за авио-компанију 2 је приказано на листингу 37. Заглавље text/xml назначава да је излазна вредност компоненте XML. Вредности се преузимају из payload-a.

```
%dw 2.0
output text/xml
ns soapenv http://schemas.xmlsoap.org/soap/envelope/
ns loc http://localhost:8082/
---
soapenv#Envelope: {
  soapenv#Header: {},
  soapenv#Body: {
    loc#BookFlightRequest: {
      loc#FlightCode: payload.flightNumber,
      loc#Class: payload.flightClass,
      loc#CustomerEmail: payload.email,
      loc#Passengers: payload.passengers map ( passenger , indexOfPassenger ) -> {
        loc#FullName: passenger.firstName ++ " " ++ passenger.lastName,
        loc#BirthDate: passenger.birthDate,
        loc#PassportID: passenger.passportNumber
      }
    }
  }
}
```

Листинг 37 - Формирање тела захтева за резервацију лета авио-компаније 2

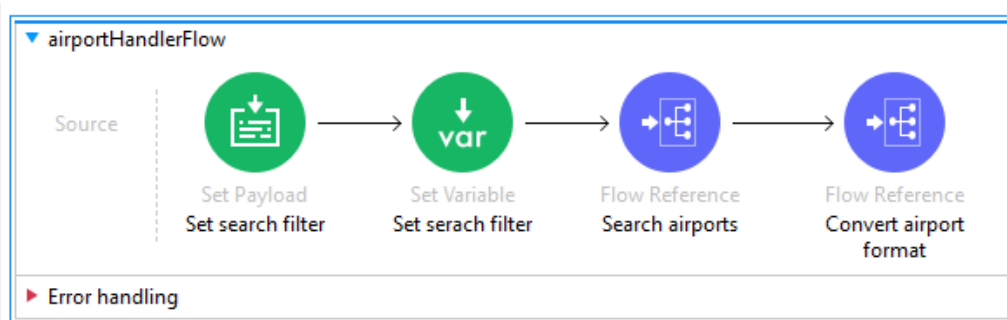
Формирање GraphQL тела захтева за авио-компанију 3 је приказано на листингу 38. Заглавље application/json назначава да је излазна вредност компоненте JSON.

```
%dw 2.0
output application/json
var passengersList = payload.passengers map (passenger) ->
  "{ name: \"\" ++ passenger.firstName ++ \" \" ++ passenger.lastName ++ \"\", birthDate: \"\" ++ passenger.birthDate ++ \"\", passport: \"\" ++ passenger.passportNumber ++ \"\" }"

var passengersString = "[" ++ (passengersList joinBy ", ") ++ "]"
---
{
  query: "mutation BookFlight {
    BookFlight(
      input: {
        flightSerialNumber: \"\" ++ payload.flightNumber ++ \"\",
        flightClass: \"\" ++ payload.flightClass ++ \"\",
        email: \"\" ++ payload.email ++ \"\",
        passengers: \"\" ++ passengersString ++ \"\"
      }
    ) {
      message
    }
  }"
}
```

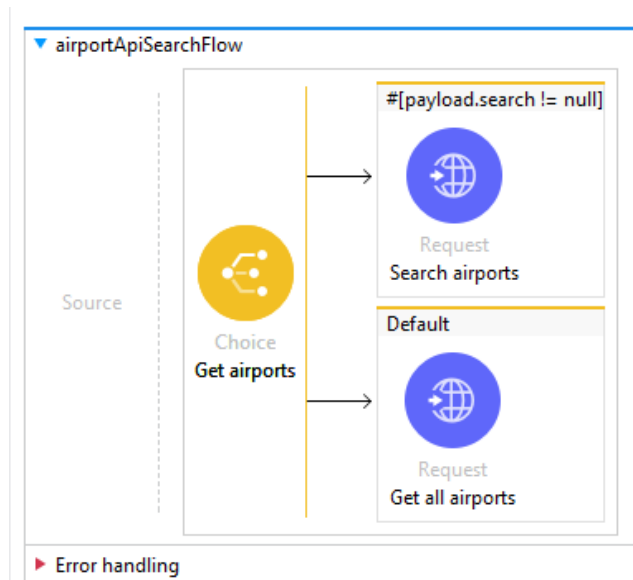
Листинг 38 - Формирање тела захтева за резервацију лета авио-компаније 3

Претрага аеродрома је помоћна функционалност система за претрагу летова. Њена главна намена је да омогући кориснику увид у све аеродроме који су предвиђени у систему. Главни ток података ове функционалности је airportHandlerFlow (слика 26). Његов задатак припрема критеријума претраге позивање тока података који добавља аеродроме и позивање тока података који врши конверзију формата података. Као излазни параметар враћа листу аеродрома у JSON формату.



Слика 26 - Ток података airportHandlerFlow

Ток података за добављање аеродрома као улазни параметар узима критеријум претраге. У случају да је критеријум претраге празан, од система за претрагу летова се захтевају сви аеродроми. У случају да није празан, од система за претрагу аеродрома се захтева претрага аеродрома. Овај ток података приказан је на слици 27.



Слика 27 - Ток података `airportApiSearchFlow`

Ток података за конверзију формата аеродрома као улазни параметар узимају листу аеродрома у CSV формату. Главна компонента је једна Transform Message компонента која је задужена за Dataweave конверзију аеродрома из CSV формата у JSON формат. DataWeave израз који је употребљен за ову конверзију дат је на листингу 39.

```

%dw 2.0
output application/json
---
payload map ( payload01 , indexOfPayload01 ) -> {
    elevation: payload01.Elevation as Number,
    continent: payload01.Continent,
    country: payload01.Country,
    Iata: payload01.IATA,
    latitude: payload01.Latitude as Number,
    name: payload01.Name,
    municipality: payload01.Municipality,
    region: payload01.Region,
    longitude: payload01.Longitude as Number
}

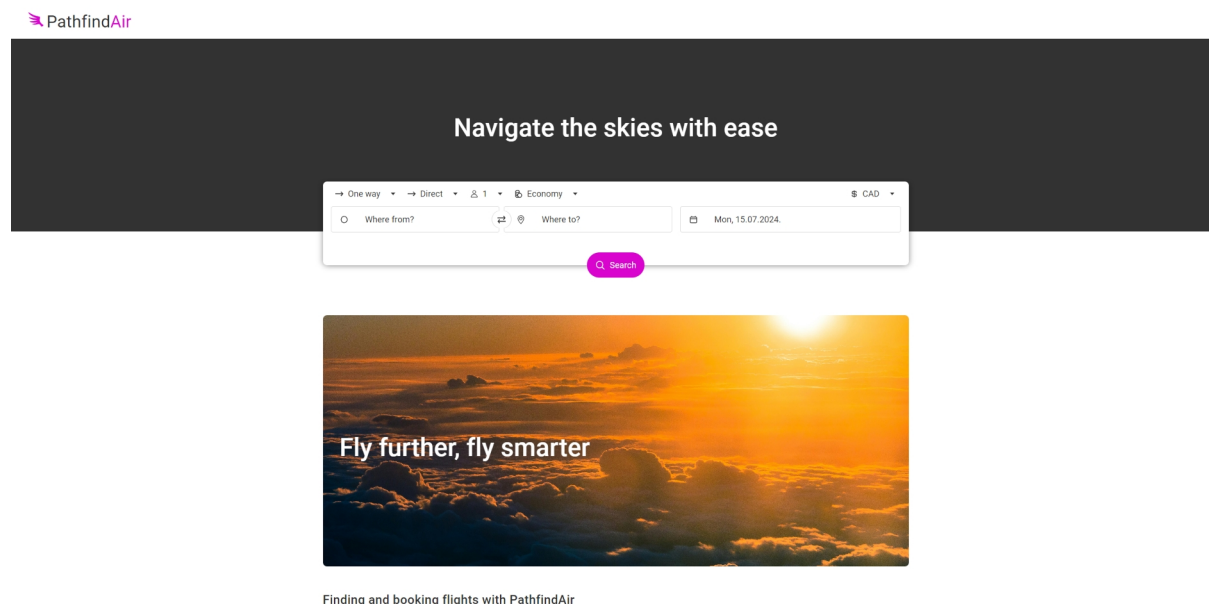
```

Листинг 39 - Конверзија формата летова

6. ДЕМОНСТРАЦИЈА

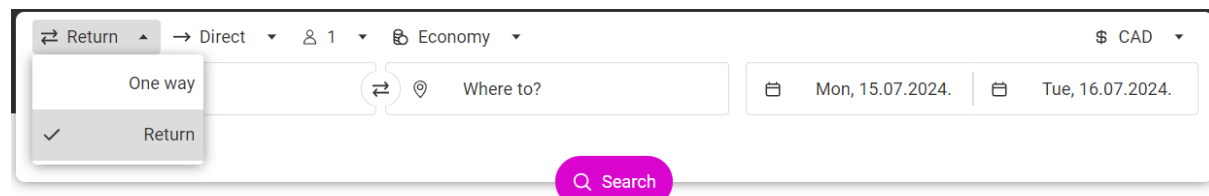
Апликација за претрагу и резервацију летова омогућава корисницима да на једном месту претражују летове из више авио компанија, као и да резервишу карте. У овом одељку, демонстрирани су битни елементи коришћења апликације кроз детаљно упутство које обухвата корак по корак интеракцију корисника са апликацијом. Кроз један сценарио коришћења приказано је како корисници могу претраживати летове, аеродроме, и резервисати карте, уз слике које приказују релевантне фазе у коришћењу апликације.

Корисник улази на почетну страницу апликације и на њој уочава форму за претрагу летова (слика 28). На форми се налазе поља за унос следећих параметара: почетни аеродром, аеродром дестинације, датум поласка, класа летења, број путника, тип лета (повратни или једносмерни, директни или са преседањем), валута. У случају да се изабере повратни лет потребно је унети и датум повратка.



Слика 28 - Почетна страница

Корисник прво бира да ли жели једносмерни или повратни лет преко падајуће листе. Овај део форме приказан је на слици 29. У овом сценарију коришћења изабран је повратни лет.



Слика 29 - Избор типа лета (једносмерни или повратни)

Корисник затим бира да ли жели искључиво директне летове или му је прихватљиво и да преседа. Овај део форме приказан је на слици 30. У овом сценарију коришћења изабрани су летови са преседањем.

Слика 30 - Избор типа лета (директни или са преседањем)

Корисник затим уноси информације о броју путника приликом чега посебно уноси број одраслих путника и број деце. Одраслим путницима се сматрају особе које имају 12 или више година. Овај део форме приказан је на слици 31. У овом сценарију коришћења корисник путује са једним дететом.

Слика 31 - Унос броја путника

Корисник затим бира класу летења из падајуће листе. Може да изабере једну од 3 опције: економска, бизнис или прва класа. Овај део форме приказан је на слици 32. У овом сценарију коришћења изабрана је прва класа.

Слика 32 - Избор класе летења

Корисник затим бира аеродроме. Почетни и крајњи аеродром се могу претраживати у оквиру падајуће листе. Изабрани аеродроми морају бити различити. Овај део форме приказан је на слици 33. У овом случају коришћења траже се летови из Њу Јорка (JFK) до Атланте (ATL).

Слика 33 - Избор аеродрома

Корисник затим бира датуме поласка и повратка путем форме, при чему датуми не смеју бити у прошлости, а датум поласка мора бити након датума повратка. У случају једносмерних летова бира се само један датум. Форма за избор датума је приказан на слици 34. У овом случају коришћења потребан је лет који полеће 1. децембра 2025. године са повратком 2. децембра 2025. године.

Return With stops 2 First \$ CAD

New York (JFK) Atlanta (ATL) Mon, 01.12.2025

Search

Return

Day 02

Month 12

Year 2025

Слика 34 - Избор датума

Корисник затим опционо из падајуће листе бира валуту у којој жели да му буду приказане цене. Овај део форме приказан је на слици 35. У овом случају коришћења изабран је евро.

Return With stops 2 First \$ EUR

New York (JFK) Atlanta (ATL) Mon, 01.12.2025 Tue, 02.12.2025

Search

EUR

FJD

FKP

Слика 35 - Избор валуте

Након уноса свих информација корисник кликом на дугме за претрагу покреће претрагу летова. На следећој страници приказана му је листа летова приказана на слици 36.

[Back to search](#)




Best departing flights · JFK - ATL · Mon, 01.12.2025.

Ranked based on price and convenience. Listed prices include the price of tickets for every passenger. Optional charges and bag fees may apply.

	Departure · Mon, 01.12.2025.	8 hr 0 min JFK - ATL	0 stops 0 hr 0 min	Select flight	EUR 32.5	▼
	Departure · Mon, 01.12.2025.	9 hr 20 min JFK - ATL	2 stops 2 hr 40 min	Select flight	EUR 97.5	▼
	Departure · Mon, 01.12.2025.	1 hr 0 min JFK - ATL	0 stops 0 hr 0 min	Select flight	EUR 3489.54	▼
	Departure · Mon, 01.12.2025.	8 hr 40 min JFK - ATL	1 stops 4 hr 0 min	Select flight	EUR 3522.04	▼
	Departure · Mon, 01.12.2025.	9 hr 20 min JFK - ATL	2 stops 4 hr 0 min	Select flight	EUR 3554.54	▼
	Departure · Mon, 01.12.2025.	8 hr 45 min JFK - ATL	1 stops 3 hr 0 min	Select flight	EUR 3832.5	▼
	Departure · Mon, 01.12.2025.	1 hr 40 min JFK - ATL	1 stops 2 hr 0 min	Select flight	EUR 6979.08	▼

Слика 36 - Приказ полазних летова

Уколико жели да види детаљне информације о лету, корисник кликом на картицу лета може да је прошири. У случају летова са преседањем приказују се информације о сваком лету у оквиру путање, као и информације о периодима чекања на следећи лет. Ово је приказано на слици 37.

	Departure · Mon, 01.12.2025.	8 hr 45 min JFK - ATL	1 stops 3 hr 0 min	Select flight	EUR 3832.5 ^
	<ul style="list-style-type: none"> 9:00 AM · John F Kennedy International Airport (JFK) Travel time: 8 hr 0 min 5:00 PM · Dallas Fort Worth International Airport (DFW) 				\$ EUR 32.5 First 20kg per passenger
Horizon Airways · AL2-A8B1508594					
3 hr 0 min layover · Dallas Fort Worth International Airport (DFW)					
	<ul style="list-style-type: none"> 8:00 PM · Dallas Fort Worth International Airport (DFW) Travel time: 0 hr 45 min 8:45 PM · Hartsfield Jackson Atlanta International Airport (ATL) 				\$ EUR 3800 First 20kg
Eclipse Air · AL1-127DF889AB					





Слика 37 - Детаљан приказ лета

Корисник бира лет кликом на дугме за избор лета. Након овога му се на сличан начин приказује листа повратних летова приказана на слици 38.

[Back to departing flights](#)

Best returning flights · ATL - JFK · Tue, 02.12.2025.

Ranked based on price and convenience. Listed prices include the price of tickets for every passenger. Optional charges and bag fees may apply.



	Return · Tue, 02.12.2025.	4 hr 0 min ATL - JFK	1 stops 2 hr 0 min	Select flight	EUR 65 ▾
	Return · Tue, 02.12.2025.	4 hr 0 min ATL - JFK	1 stops 2 hr 0 min	Select flight	EUR 65 ▾
	Return · Tue, 02.12.2025.	8 hr 0 min ATL - JFK	0 stops 0 hr 0 min	Select flight	EUR 3489.54 ▾
	Return · Tue, 02.12.2025.	9 hr 0 min ATL - JFK	1 stops 1 hr 0 min	Select flight	EUR 6979.08 ▾

Слика 38 - Приказ повратних летова

Корисник бира повратни лет кликом на дугме за избор лета. Након овога кориснику се још једном приказују изабрани летови као и форма за попуњавање информација о путницима. Ова страница је приказана на слици 39.


[← Back to returning flights](#)

Selected flights


 Departure · Mon, 01.12.2025.	8 hr 45 min JFK - ATL	1 stops 3 hr 0 min	EUR 3832.5 ▾
 Return · Tue, 02.12.2025.	8 hr 0 min ATL - JFK	0 stops 0 hr 0 min	EUR 3489.54 ▾


Passengers

1. Passenger (adult)


 First name

Last name


 Passport number


 Birthdate: 15.07.2012.

2. Passenger (child)


 First name


Last name

 Passport number

 Birthdate: 15.07.2024.

Contact

 Email




Слика 39 - Страница за резервацију летова


Корисник затим уноси информације о свим путницима које укључују име, презиме, број пасоша и датум рођења. Такође, уноси и мејл на који ће га контактирати све авио компаније које учествују у изабраним летовима. Попуњена форма је приказана на слици 40.


Passengers

1. Passenger (adult)


 Marko

Nikolić


 67723308


 Birthdate: 21.12.2001.

2. Passenger (child)


 Petar


Petrović

 9048826

 Birthdate: 15.07.2015.

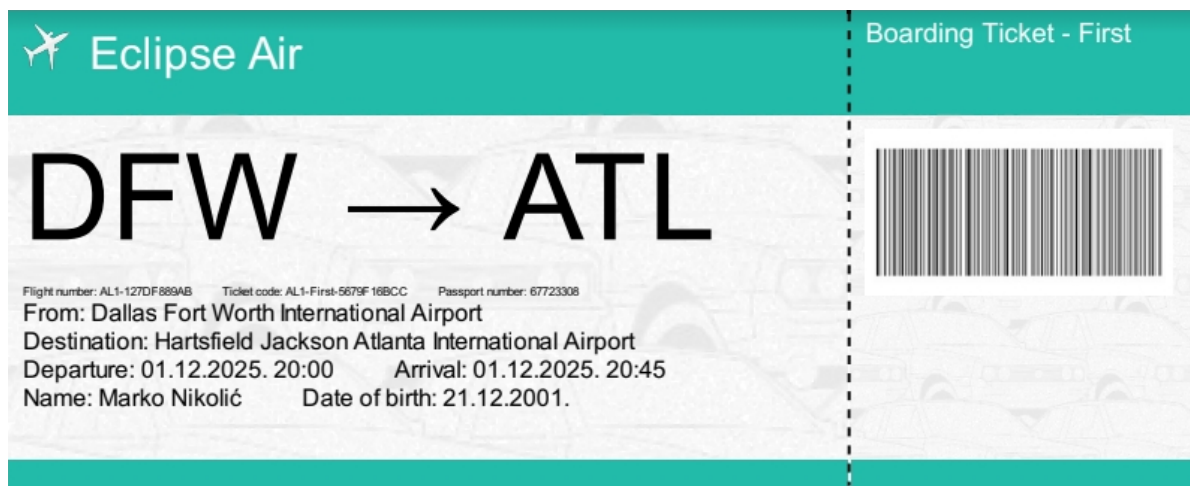
Contact

 nikolicmarko1243@gmail.com



Слика 40 - Форма за информације о путницима

Након уноса информација о путницима, корисник резервише карте кликом на дугме. Добија потврду и информације о следећим корацима. На мејл добија све потребне карте у PDF формату. Карте из различитих авио компанија приказане су на сликама 41, 42, и 43.



Слика 41 - Карта авио-компаније 1



Слика 42 - Карта авио-компаније 2



Слика 43 - Карта авио-компаније 3

7. ЗАКЉУЧАК

У овом раду је представљено софтверско решење којим се омогућава интеграција и управљање претрагом и резервацијом летова из три различите авио-компаније, система за претрагу аеродрома, система за претрагу летова са преседањем и Exchange-Rate API-ја. Дата је спецификација свих компоненти система, укључујући све поменуте системе, као и функционалности за претрагу и резервацију летова. Објашњена је интеграција система, која укључује употребу Enterprise Service Bus (ESB) архитектурног обрасца за интеграцију различитих система. Представљена је имплементација описаног решења уз пролаз кроз програмски код. Кроз детаљан сценарио коришћења апликације описано је како корисници могу претраживати летове и резервисати карте.

Предности решења:

- Интеграција - систем обједињује више апликација и сервиса, што корисницима омогућава приступ већем броју опција за путовање.
- Флексибилност - подршка за различите API стандарде (REST, SOAP, GraphQL) омогућава интеграцију са разноврсним системима.
- Скалабилност - употреба ESB архитектуре омогућава додавање нових сервиса и функционалности без значајних промена у систему.

Мане решења:

- Перформансе - усложњавање система кроз интеграцију више сервиса може утицати на перформансе и брзину одговора система.
- Зависност од спољних система - систем зависи од доступности и поузданости спољних API-ја и сервиса, што може утицати на стабилност и поузданост апликације.

У поређењу са сродним решењима, апликација пружа једноставнији начин да се резервишу карте из више авио-компанија истовремено. Већина сродних система ослања се на редирекцију приликом резервације где је неко друго решење одговорно за резервацију док је систем одговоран само за претрагу летова. Приликом резервисања летова више пута, често се деси да је за резервацију карата за различите компаније одговоран различит сервис за резервацију карата што може довести до неконзистентног корисничког искуства. Систем описан у овом раду се не ослања на редирекцију и има уграђену функционалност за резервацију карата из свих компанија које су подржане у систему. Ово значи да приликом вишеструке употребе система, корисник има конзистентније искуство и мање простора за грешку.

Са друге стране, сродна решења имају напредније функционалности за претрагу и филтрирање летова што омогућава кориснику да на лакши начин пронађе свој идеалан лет. Такође, у сродна решења је интегрисан велики број авио-компанија што проширује понуду летова. Још једна предност је што сродна решења имају подршку за стране језике што повећава број потенцијалних корисника.

Пожељна особина будућих апликација из ове области било би и укључивање напредне функције као што су персонализоване препоруке на основу претходних претрага и резервисања, предиктивна анализа цена и аутоматизовани алати за оптимизацију путовања. Ове функције би могле додатно побољшати корисничко искуство и

омогућити корисницима да доносе информисане одлуке при одабиру најповољнијих и најприкладнијих летова.

У будућности би се мане описаног решења могле отклонити кроз:

- Интеграцију са већим бројем авио-компанија – интеграција нових авио-компанија у систем би проширила понуду летова.
- Побољшање претраге летова - увођење нових критеријума претраге и филтера резултата омогућила би кориснику да лакше и брже пронађе лет који му највише одговара.
- Унапређење алгоритма за претрагу летова са преседањем – оптимизација алгоритма ради бољих перформанси система.
- Унапређење интеграције – побољшање, паралелизација и оптимизација токова података ради бољих перформанси система.
- Побољшање стабилности - увођење механизма за надгледање и управљање грешкама како би се осигурала већа поузданост и стабилност система.

8. ЛИТЕРАТУРА

- [1] Kyle Wilson, "How to use Google Flights: Find cheap flight options, search multiple airlines at once, and track flight prices", доступно на: <https://www.businessinsider.com/google-flights> (посећено 01.07.2024.)
- [2] Google, "About Google", доступно на: <https://about.google/> (посећено 01.07.2024.)
- [3] Think With Google, "Consumer trends", <https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/google-flight-visitor-statistics/> (посећено 01.07.2024.)
- [4] Skyscanner, "About us", <https://www.skyscanner.net/about-us> (посећено 01.07.2024.)
- [5] Kayak, "About", <https://www.kayak.com/about> (посећено 01.07.2024.)
- [6] Cheapflights, "About Cheapflights", доступно на: <https://www.cheapflights.com/about> (посећено 01.07.2024.)
- [7] PostgreSQL, "PostgreSQL: The World's Most Advanced Open Source Relational Database", доступно на: <https://www.postgresql.org/> (посећено 02.07.2024.)
- [8] Rahul Awati, Adam Hughes, Craig Stedman, "What is Microsoft SQL Server?", доступно на: <https://www.techtarget.com/searchdatamanagement/definition/SQL-Server#:~:text=Microsoft%20SQL%20Server%20is%20a,applications%20in%20corporate%20IT%20environments> (посећено 02.07.2024.)
- [9] Microsoft, "About Microsoft", доступно на: <https://www.microsoft.com/en-us/about> (посећено 02.07.2024.)
- [10] MongoDB, "What Is MongoDB?", доступно на: <https://www.mongodb.com/company/what-is-mongodb> (посећено 02.07.2024.)
- [11] Node.js, "Node.js documentation", доступно на: <https://nodejs.org/docs/latest/api/> (посећено 02.07.2024.)
- [12] W3Schools, "What is JavaScript?", доступно на: https://www.w3schools.com/whatis/whatis_js.asp (посећено 02.07.2024.)
- [13] Express, "What is JavaScript?", доступно на: <https://expressjs.com/> (посећено 02.07.2024.)
- [14] Spring, "Spring Framework Documentation", доступно на: <https://docs.spring.io/spring-framework/reference/index.html> (посећено 02.07.2024.)
- [15] Java, "Learn Java", доступно на: <https://dev.java/learn/> (посећено 02.07.2024.)
- [16] Microsoft Learn, "ASP.NET documentation", доступно на: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0> (посећено 02.07.2024.)
- [17] Microsoft Learn, "C# documentation", доступно на: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/> (посећено 02.07.2024.)
- [18] Go, "Documentation", доступно на: <https://go.dev/doc/> (посећено 02.07.2024.)
- [19] Angular, "What is Angular?", доступно на: <https://angular.dev/overview> (посећено 02.07.2024.)

- [20] TypeScript, "TypeScript Documentation", доступно на: <https://www.typescriptlang.org/fr/docs/> (посећено 02.07.2024.)
- [21] W3Schools, "What is JSON?", доступно на: https://www.w3schools.com/whatis/whatis_json.asp (посећено 02.07.2024.)
- [22] W3Schools, "Introduction to XML", доступно на: https://www.w3schools.com/xml/xml_whatis.asp (посећено 02.07.2024.)
- [23] Chris Hoffman, "What Is a CSV File, and How Do I Open It?", доступно на: <https://www.howtogeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it/> (посећено 02.07.2024.)
- [24] Codecademy, "What is REST?", доступно на: <https://www.codecademy.com/article/what-is-rest> (посећено 02.07.2024.)
- [25] W3Schools, "What is HTTP?", доступно на: https://www.w3schools.com/whatis/whatis_http.asp (посећено 02.07.2024.)
- [26] GeeksForGeeks, "Basics of SOAP – Simple Object Access Protocol", доступно на: <https://www.geeksforgeeks.org/basics-of-soap-simple-object-access-protocol/> (посећено 02.07.2024.)
- [27] GraphQL, "Introduction to GraphQL", доступно на: <https://graphql.org/learn/> (посећено 02.07.2024.)
- [28] MuleSoft, "Anypoint Studio", доступно на: <https://docs.mulesoft.com/studio/latest/> (посећено 02.07.2024.)
- [29] Git, "Documentation", доступно на: <https://git-scm.com/doc> (посећено 02.07.2024.)
- [30] Visual Paradigm, "Ideal Modeling & Diagramming Tool for Agile Team Collaboration", доступно на: <https://www.visual-paradigm.com/> (посећено 03.07.2024.)
- [31] Diagrams.net, "Diagram Software and Flowchart Maker", доступно на: <https://www.diagrams.net/> (посећено 03.07.2024.)
- [32] Exchange-Rate API, "The Accurate & Reliable Exchange Rate API", доступно на: <https://www.exchangerate-api.com/> (посећено 03.07.2024.)
- [33] MuleSoft, "What is an ESB?", доступно на: <https://www.mulesoft.com/resources/esb/what-esb> (посећено 03.07.2024.)
- [34] Swagger, "Swagger UI", доступно на: <https://swagger.io/tools/swagger-ui/> (посећено 04.07.2024.)
- [35] Sequelize, "Getting Started", доступно на: <https://sequelize.org/docs/v6/getting-started/> (посећено 04.07.2024.)
- [36] npm, "html-pdf-node", доступно на: <https://www.npmjs.com/package/html-pdf-node> (посећено 04.07.2024.)
- [37] npm, "Nodemailer", доступно на: <https://www.npmjs.com/package/nodemailer> (посећено 04.07.2024.)
- [38] SoapUI, "Comprehensive Testing to Accelerate API Quality", доступно на: <https://www.soapui.org/> (посећено 05.07.2024.)

- [39] javadoc, "Jakarta Validation API", доступно на: <https://javadoc.io/doc/jakarta.validation/jakarta.validation-api/latest/index.html> (посећено 05.07.2024.)
- [40] Rijul Dahiya, "Creating a PDF Export API with iText PDF in Java Spring Boot", доступно на: <https://medium.com/@rijuldahiya/creating-a-pdf-export-api-with-itext-pdf-in-java-spring-boot-a0bdf5f8f57> (посећено 05.07.2024.)
- [41] Alfrick Opidi, "Jakarta Mail Tutorial", доступно на: <https://mailtrap.io/blog/jakarta-mail-tutorial/> (посећено 05.07.2024.)
- [42] Jakarta EE, "Jakarta XML Binding", доступно на: <https://jakarta.ee/specifications/xml-binding/> (посећено 05.07.2024.)
- [43] W3Schools, "XML WSDL", доступно на: https://www.w3schools.com/xml/xml_wsdl.asp (посећено 05.07.2024.)
- [44] W3Schools, "XML Schema Tutorial", доступно на: https://www.w3schools.com/xml/schema_intro.asp (посећено 05.07.2024.)
- [45] Postman, "What is Postman?", доступно на: <https://www.postman.com/product/what-is-postman/> (посећено 06.07.2024.)
- [46] Microsoft Learn, "Entity Framework documentation hub", доступно на: <https://learn.microsoft.com/en-us/ef/> (посећено 06.07.2024.)
- [47] AutoMapper, "AutoMapper", доступно на: <https://docs.automapper.org/en/stable/> (посећено 06.07.2024.)
- [48] NuGet, "itext7.pdfhtml", доступно на: <https://www.nuget.org/packages/itext7.pdfhtml> (посећено 06.07.2024.)
- [49] Microsoft Learn, "SmtpClient Class", доступно на: <https://learn.microsoft.com/en-us/dotnet/api/system.net.mail.smtpclient?view=net-8.0> (посећено 06.07.2024.)
- [50] ChilliCream, "HotChocolate Documentation", доступно на: <https://chillicream.com/docs/hotchocolate/v12/defining-a-schema/documentation> (посећено 06.07.2024.)
- [51] Python, "Python documentation", доступно на: <https://docs.python.org/3/> (посећено 07.07.2024.)
- [52] Flask, "User's Guide", доступно на: <https://flask.palletsprojects.com/en/3.0.x/> (посећено 07.07.2024.)
- [53] GeeksForGeeks, "Breadth First Search or BFS for a Graph", доступно на: <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/> (посећено 07.07.2024.)
- [54] MuleSoft, "DataWeave Overview", доступно на: <https://docs.mulesoft.com/dataweave/latest/> (посећено 08.07.2024.)
- [55] MuleSoft, "Mule Components", доступно на: <https://docs.mulesoft.com/mule-runtime/latest/mule-components> (посећено 08.07.2024.)
- [56] AYR Tech (Pty) Ltd, "AYR Tech (Pty) Ltd", доступно на: [https://ayr.co.za/#:~:text=AYR%20Tech%20\(Pty\)%20Ltd%20is,and%20developer's%20ease%20of%20use](https://ayr.co.za/#:~:text=AYR%20Tech%20(Pty)%20Ltd%20is,and%20developer's%20ease%20of%20use) (посећено 08.07.2024.)

- [57] RAML, "The simplest way to model APIs", доступно на: <https://raml.org/>
(посећено 08.07.2024.)

БИОГРАФИЈА

Марко Николић је рођен 21. децембра 2001. године у Суботици. Од 2006. године живи у Новом Саду. Основно образовање стекао је у Основној школи "Милош Црњански" у Новом Саду коју је похађао од 2008. до 2016. године. Током школовања, освојио је дипломе из српског језика и књижевности, биологије и историје. Био је одликован титулом ђака генерације и добитник дипломе "Вук Караџић".

Након завршене основне школе, уписао је природно-математички смер у Гимназији "Јован Јовановић Змај" у Новом Саду коју је похађао од 2016. до 2020. године. И у гимназији је био успешан ученик и добитник дипломе "Вук Караџић".

2020. године, Марко се уписао на Факултет техничких наука у Новом Саду на смер Рачунарство и аутоматика. Током студија, определио се за студијски модул Примењене рачунарске науке и информатика. Положио је све испите предвиђене планом и програмом и стекао услов за одбрану завршног рада.