УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Милица Пантелић

Симулација функционалности протокола за преузимање електронске поште

ДИПЛОМСКИ РАД

Основне академске студије

Нови Сад, 2025

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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска публикација
Тип записа, **ТЗ**:		Текстуална штампана документација
Врста рада, **ВР**:		Дипломски рад
Аутор, **АУ**:		Милица Пантелић
Ментор, **МН**:		доц. др. Милана Бојанић
Наслов рада, **НР**:		Симулација функционалности протокола за преузимање електронске поште
Језик публикације, **ЈП**:		Српски
Језик извода, **ЈИ**:		Српски/Енглески
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2025.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Факултет техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		5/25/11/1/8/0/0
Научна област, **НО**:		Електротехничко и рачунарско инжењерство
Научна дисциплина, **НД**:		Примењено софтверско инжењерство
Предметна одредница/Кључне речи, **ПО**:		електронска пошта, IMAP протокол, сервер, клијент
УДК		
Чува се, **ЧУ**:		Библиотека ФТН, Нови Сад, Трг Доситеја Обрадовића 6
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У оквиру овог рада анализирано је понашање и главне функционалности протокола за преузимање електронске поште. За потребе ове симулације имплементирана је апликација која прима поруке од стране пошиљаоца и смешта их у поштанско сандуче примаоца. Прималац има могућност да управља пристиглим порукама, премешта их из једног фолдера у други, брише и означава поруке непрочитаним.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		18.12.2025.
Чланови комисије, **КО**:	Председник:	доц. др Себастијан Стоја
	Члан:	доц. др Бојан Јелачић
	Члан:	Потпис ментора
	Члан, ментор:	доц. др Милана Бојанић

UNIVERSITY OF NOVI SAD●**FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, TrgDositejaObradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual material, printed	
Contents code, **CC**:		Bachelor thesis	
Author, **AU**:		Milica Pantelić	
Mentor, **MN**:		Milana Bojanić, Ph.D.	
Title, **TI**:		Simulation of email retrieval protocol functionality	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian/English	
Country of publication, **CP**:		Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2025.	
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)		5/25/11/1/8/0/0	
Scientific field, **SF**:		Electrical and computer engineering	
Scientific discipline, **SD**:		Applied Software Engineering	
Subject/Key words, **S/KW**:		email, IMAP protocol, server, client	
UC			
Holding data, **HD**:		Library of Faculty of Technical Sciences, Novi Sad, Trg Dositeja Obradovića 6	
Note, **N**:			
Abstract, **AB**:		Within this paper, the behavior of email retrieving protocols has been analyzed. For the purpose of the simulation, an application was implemented that receives messages from a sender and stores them in the recipient's mailbox. The recipient is able to manage the received messages, move them between folders, delete them, and mark previously read messages as unread.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:		18.12.2025.	
Defended Board, **DB**:	President:	Asst. prof. Sebastijan Stoja,Ph.D.	
	Member:	Asst. prof. Bojan Jelačić, Ph.D.	
	Member:		Menthor's sign
	Member, Mentor:	Asst. prof. Milana Bojanić, Ph.D.	

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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

(Податке уноси предметни наставник - ментор)

Студијски програм:	Примењено софтверско инжењерство		
Студент:	Милица Пантелић	Број индекса:	ПР 65/2020
Степен и врста студија:	Основне академске студије		
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	Др Милана Бојанић, доцент		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ ЗАВРШНОГ РАДА:

Симулација функционалности протокола за преузимање електронске поште

ТЕКСТ ЗАДАТКА:

Проучити карактеристике и функционисање протокола везаних за слање и преузимање електронске поште. Обезбедити имплементацију најважнијих функционалности протокола за преузимање е-поште по узору на IMAP протокол. У имплементацији покрити следеће: слање имејла ка изабраном кориснику, а на пријемној страни преглед свих пристиглих порука, читање изабраног имејла, брисање или означавање поруке као непрочитане. Тестирати решење у различитим тестним сценаријима комуникације клијента са мејл сервером приликом приступа е-пошти.

Руководилац студијског програма:	Ментор рада:

Примерак за: o - Студента; o -Ментора

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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Изјављујем да нисам у сукобу интереса у односу ментор – кандидат и да нисам члан породице (супружник или ванбрачни партнер, родитељ или усвојитељ, дете или усвојеник), повезано лице (крвни сродник ментора/кандидата у правој линији, односно у побочној линији закључно са другим степеном сродства, као ни физичко лице које се према другим основама и околностима може оправдано сматрати интересно повезаним са ментором или кандидатом), односно да нисам зависан/на од ментора/кандидата, да не постоје околности које би могле да утичу на моју непристрасност, нити да стичем било какве користи или погодности за себе или друго лице било позитивним или негативним исходом, као и да немам приватни интерес који утиче, може да утиче или изгледа као да утиче на однос ментор-кандидат.

У Новом Саду, дана _____

Ментор

_____

Кандидат

_____

Садржај

1. Увод

Брз развој информационих технологија и све већа доступност интернета довели су до тога да електронска пошта постане најчешће коришћен облик комуникације савременог доба. Предности које овај вид комуникације има у односу на традиционалне начине размене информација јесу његова брзина, поузданост и приступачност.

Електронска пошта ради помоћу неколицине протокола који контролишу слање и размену података између клијената и сервера. Најбитнији међу њима су **SMTP**(*Simple Mail Transfer Protocol*), који је задужен за слање порука и **IMAP**(*Internet Message Access Protocol*), који је задужен да омогући корисницима да преузимају и управљају порукама које се налазе на серверу.

Овај рад нам омогућава увид у симулацију рада протокола за пријем електронске поште са фокусом на карактеристике IMAPпротокола. Реализована апликација нам на практичан начин приказује принцип функционисања система електронске поште у којем више корисника може да прима, чита, означава и брише поруке.

Апликација је имплементирана у **C#** програмском језику у оквиру **.NET** платформе, помоћу **WPF** (*Windows Presentation Foundation*) технологије која нам омугућава креирање корисничког интерфејса. Овај начин нам омогућава да на јасан и једноставан начин прикажемо рад клијент-сервер архитектуре и тако симулирамо процес пријема електронске поште.

Поред практичне имплементације, рад обухвата и теоријску анализу електронске поште, структуру и принцип рада IMAPпротокола, као и опис технологија које су коришћене у реализацији апликације.

2. Теоријска основа

2.1 Електронска пошта

Електронска пошта (е-пошта/имејл/*e-mail*) је један од најстаријих и најкоришћенијих сервиса који нам Интернет омогућава.Његова основна улога је размена како текстуалних тако и мултимедијалних порука између корисника који се налазе на различитим уређајима или локацијама.

Сваки систем е-поште се састоји од три главне компоненте[1]:

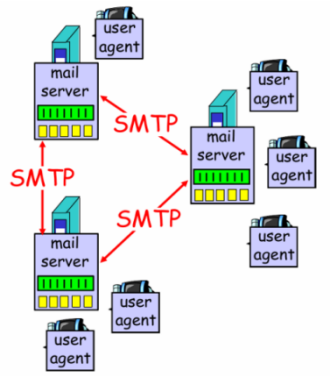
1. ***Useragent***– кориснички програм за е-пошту
2. ***Mail server***
3. **SMTP** (*Simple Mail Transfer Protocol*).

Useragent (имејл клијент) је апликација која омогућава корисницима унос, слање, пријем и приказ е-поште. То је апликација која комуницира са сервером ради приступа долазним и одлазним порукама које су смештене на серверу. Постоје две варијанте: десктоп имејл клијенти и веб-мејл интерфејси. Примери десктоп имејл клијента су *MicrosoftOutlook, MozillaThunderbird, AppleMail.*Неки од најпознатијих примера веб-мејл интерфејса којима се приступа путем веб-читача су *Gmail* и *YahooMail*.

Мејл сервери се састоје из два дела *mailbox* (поштанско сандуче корисника) и реда порука за слање. *Mailbox* садржи долазне (примљене) поруке за одређеног корисника [1]. Ред порука нам, као и што сам назив каже, служи да се у њега смештају поруке за слање (оне поруке које чекају слање, али не могу тренутно да се пошаљу из неких разлога).

SMTP је апликациони протокол који омогућава пренос порука између мејл сервера. За слање користи TCP како би омогућио поуздан пренос порука. На клијентској стани се користи за слање порука, док се на серверској страни користи за пријем порука.

На слици 2.1.1 је приказана шема система за размену е-поште.



Слика 2.1.1 Приказ система за размену е-поште

2.2 Клијент - сервер архитектура

Архитектура е-поште је заснована на клијент – сервер архитектури, где сервер представља централну тачку у којој се складиште и обрађују подаци, док је корисницима омогућен приступ тим подацима са различитих уређаја.

У оваквом моделу, клијент шаље захтев серверу, који одговара са неопходним подацима или операцијама(нпр. слање поруке, преузимање порука, брисање...) [2]. Овакав модел нам омоућава да више клијената у исто време комуницира са сервером и приступа својим налозима без доласка до конфликта.

У пракси нам је овакав приступ омогућио развој бројних email сервиса као што су *Gmail, Outlook, Yahoo! Mail*, и други, где се све поруке чувају централизовано и сви корисници могу да им приступе са различитих уређаја.

2.3 Протоколи електронске поште

Протоколи представљају скуп правила који дефинишу на који начин ће уређаји комуницирати унутар неке мреже.

Када је реч о електронској пошти, неки од најважнијих протокола су:

1. **SMTP** (*Simple Mail Transfer Protocol*)
2. **POP3** (*Post Office Protocol v3*)
3. **IMAP** (*Internet Message Access Protocol*)
4. **HTTP** *(Hypertext Transfer Protocol).*

Главна разлика између наведених протокола јесте што SMTP протокол служи искључиво за слање имејл порука од имејл клијента ка мејл серверу и слање између два мејл сервера. Остала три протокола (POP3, IMAP и HTTP) служе за преузимање имејл порука са сервера до имејл клијента, како је приказано на слици 2.3.1.

SMTP служи за пренос порука између мејл сервера преко TCP везе [1]. TCP нам омогућава безбедан пренос порука преко порта 25. Овај протокол врши директан пренос од клијента ка серверу, ако је сервер тренутно недоступан покушаће касније. Пренос се врши у три фазе: успостава везе (*handshake*), пренос поруке, затварање везе. Првобитно поруке су морале бити искључиво текстуалне у 7-битном *ASCII* коду [1].Накнадно развијен је MIME (*Multipurpose Internet Mail Extensions*) стандард и проширење који омогућавају кодирање произвољног бинарног садржаја у 7-битни *ASCII* код и тако омогућавају слање различитих прилога уз тело имејл поруке.

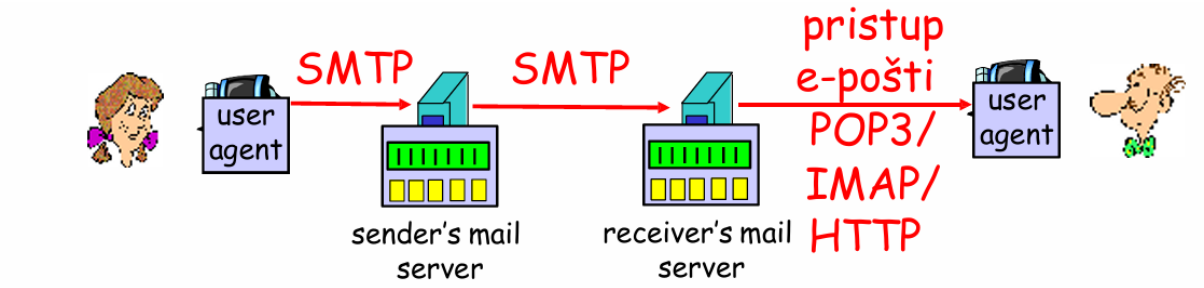
POP3 је један од протокола за преузимање електронске поште. Он је једноставан, након што корисник прочита поруку сервер је више не памти.

Имамо опцију да скинемо поруку на наш уређај, али ако дође до промене промене уређаја или *User agent-a* поруке ће заувек нестати. POP3 ради на порту 110.

IMAP је такође протокол за преузимање поште као и POP3, али за разлику од њега он памти сву пошту. IMAP, развијен 1988. Године, омогућавао је да мејлови остану на серверу пружајући приступ са више уређаја [3]. Омогућава корисницима да управљају порукама. Овај протокол нам дозвољава да управљамо фолдерима, трајно обришемо и ефикасно претражујемо поруке [4]. IMAP сервер користи порт 143, а у случају шифрованих порука говоримо о IMAP-у преко SSL/TLS конекције и сервер тада користи порт 993.

HTTPпротокол првенствено обезбеђује комуникацију између веб-читача и веб-сервера разменом HTML докумената (докумената са*hypertext*-ом).У варијанти клијентског веб-имејл интерфејса, HTTP нам омогућава преузимање имејлова са мејл сервера. На клијентској страни се налази веб читач. Интеракција клијента са мејл сервером је путем стандардних веб-захтева коришћењем HTTP протокола. Клијент шаље е-пошту серверу помоћу HTTP-а. Сервер такође шаље изабране имејл поруке до клијентског веб читача помоћу HTTP-а.

На слици 2.3.1 је приказан основни процес размене електонске поште. Порука се шаље од пошиљаоца ка серверу пошиљаоца коришћењем SMTP протокола, а затим се помоћу SMTP протоколапрослеђује ка серверу примаоца.Прималац приступа порукама које се чувају на његовоммејл серверу помоћу IMAP или POP3 протокола, док је HTTP приступ карактеристичан за веб-мејл сервисе.



Слика 2.3.1 Приказ рада протокола е-поште

2.4 IMAP протокол – структура и функционалност

IMAP (*Internet Message Access Protocol*) је протокол који омогућава клијенту да приступа порукама које се складиште на серверу без потребе да их преузме на локални уређај. Развијен је са намером да буде напреднија и флексибилнија алтернатива POP3протокола, првобитно како би омогућио приступање порукама са више различитих уређаја.

Главна карактеристика овог протокола је та да поруке остају сачуване на серверу и након што их прочитамо. Клијент преузима само метаподатке (као што су

наслов, назив пошиљаоца, датум и време), док тело поруке остаје сачувано на серверу и преузима се само кад клијент затражи да прочита садржај поруке. Овакав приступ нам омогућава да статус поруке (прочитана, непрочитана, обрисана) остане синхронизован на свим уређајима преко којих корисник приступа свом налогу.

2.4.1 Основни принцип рада

Када се корисник улогује на свој профил преко IMAP клијента, клијент успоставља везу са IMAP сервером и захтева листу порука које се налазе у корисничком *mailbox*-у.

Сервер одговара на захтев тако што шаље основне информације о свакој поруци као што су:

1. Идентификатор порука
2. Назив пошиљаоца
3. Назив примаоца
4. Наслов
5. Датум и време пријема поруке.

Цео садржај поруке се преузима са сервера само ако корисник отвори поруку. На овај начин се оптимизује саобраћај у мрежи и избегава непотребно преузимање великог броја података.

2.4.2 Комуникација између клијента и сервера

IMAP се одвија по принципу слања захтева и одговора. Клијент шаље комаде серверу, а сервер одговара резултатом извршавања операције.

Неке од основних IMAP команди су:

1. ***LOGIN***– аутентификација корисника помоћу корисничког имена и лозинке
2. ***SELECT***– избор поштанског сандучета
3. ***FETCH***– преузимање садржаја или метаподатака поруке
4. ***STORE***– промена статуса поруке
5. ***EXPUNGE***– трајно брисање порука које су означене за брисање
6. ***LOGOUT***– одјава са сервера

Овај начин комуникације се у овом раду симулира на једноставан начин: апликација садржи компоненту која симулира IMAP сервер (класа

IMAPSimulator), као и компоненту која симулира клијентску страну (WPF интерфејс) која шаље команде серверу и као одговор добија ажурирану листу порука.

2.4.3 Предности IMAP протокола

Главне предности IMAP протокола у односу на POP3 су:

1. Поруке остају сачуване на серверу и могуће им је приступити са различитих уређаја
2. Синхронизација статуса поруке (прочитана/непрочитана)
3. Организација порука у више различитих фолдера
4. Могућа претраживања порука директно на серверу
5. Смањење складишног простора на клијентском уређају.

2.4.4 Мане IMAP протокола

Иако нам пружа бројне погодности, IMAP такође има и својих недостатака. Пошто су поруке смештене на серверу, потребна нам је интернет конекција како би могли да им приступимо. Због константне синхронизације порука између клијената и сервера, IMAP може захтевати већу искоришћеност ресурса на серверу у односу на POP3. У односу на реалне системе овај проблем се може решити помоћу кеширања или компресије података, или помоћу имплементације безбедносних протокола као што су SSL/TLS.

2.5 Примена IMAP протокола у савременим системима

IMAP је тренутно један од најзаступљенијих протокола у савременим системима електронске поште. Скоро сви познати сервиси е-поште, као што су *Gmail, Yahoo!, Outlook*, подржавају IMAP као стандардни протокол за приступ пошти.

Захваљујући овом протоколу корисници могу да читају и управљају својом поштом са различитих уређаја (телефони, таблети, рачунари) без ризика да изгубе податке или синхронизацију са сервером.

У овом раду IMAP није у потпуности имплементиран, већ је симулиран кроз модел који представља основне функционалности правог IMAP система:

1. Пријем порука

2. Преглед садржаја поруке
3. Означавање поруке као прочитане или непрочитане
4. Брисање поруке са сервера.

На овај начин је приказана суштинска логика рада IMAP протокола без потребе за реалном серверском инфраструктуром.

2.6 Безбедносни аспекти електронске поште

С обзиром на то да је електронска пошта најкоришћенији вид комуникације, посебна пажња је посвећена безбедности преноса и заштити корисничких података. Савремени системи електронске поште примењују неколико механизама како би обезбедили поверљивост, интегритет и аутентичност порука.

2.6.1 Енкрипција електронске поште

Један од основних механизама за заштиту е-поште јесте коришћење **TLS** (*Transport Layer Security*) протокола који омогућава енкрипцију комуникације између корисника и сервера. На овај начин се спречава да неовлашћено треће лице пресретне податке током преноса поруке [5].

Поред тога, постоји и енкрипција од краја до краја (*end-to-end encryption* – E2EE), где се порука шифрује на страни пошиљаоца и може се дешифровати само на страни примаоца. Алати и стандарди као што су PGP (*Pretty Good Privacy*) и S/MIME се углавном користе код овог вида заштите, пружајући висок ниво поверљивости садржаја поруке [6].

2.6.2 Вишефакторска и двофакторска аутентификација

Како би пружили додатну заштиту корисничким налозима, савремени системи е-поште примењују **вишефакторску** (MFA) и **двофакторску аутентификацију** (2FA). **2FA** механизам захтева од корисника да поред корисничког имена и лозинке потврди свој идентитет помоћу неког додатног фактора као што је код послат на мобилни телефон, апликација за аутентификацију или биометријски подаци. На тај начин се значајно смањује ризик од неауторизованог приступа корисничким налозима.**MFA** захтевањем два или више фактора аутентификација значајно побољшава безбедност налога и смањује ризик од неовлашћеног приступа, чак иако је лозинка компликована [7].

2.6.3 Филтери за *spam* и *malware*

Електронска пошта је често мета злоупотребе у виду *spam* порука и злонамерних софтвера. Због тога се користе напредни механизми за филтрирање *spam* порука и *malware*-а, који анализирају садржај поруке, понашање пошиљаоца и сумњиве прилоге. Ови механизми омогућавају да потенцијално опасне поруке аутоматски буду изоловане или блокиране, чиме се значајно повећава безбедност корисника и целокупног система.

3. Опис коришћених технологија и алата

За развој апликације која симулира рад протокола за пријем електронске поште коришћене су савремене технологије и алати који нам омогућавају ефикасан рад, прегледност, и једноставну интеракцију између корисника и система. Избор ових технологија има за циљ да нам омогући јасну и једноставну програмску структуру, једноставан кориснички интерфејс, лако одржавање и скалабилност система.

У наставку су описане све коришћене технологије и алати као и њихова појашњења.

3.1 Коришћене технологије

3.1.1 .NET платформа

.NET платформа, развијена од стране *Microsoft*-а, нам омогућава да креирамо различите врсте апликација погодне за различите уређаје као што су десктоп и мобилне апликације. Она нам омогућава униформно окружење за развој, тестирање и покретање апликација и на тај начин значајно олакшава рад програмерима [8].

Главна компонента **.NET** платформе је CLR (*Common Language Runtime*) задужена за извршење кода, управљање меморијом, контролу *exception*-има и безбедносну контролу.

Поред тога **.NET** подржава различите програмске језике унутар истог пројекта као што су *C#, F#* и *Visual Basics*.

У овом пројекту **.NET** користимо као основу за **WPF** (*Windows Presentation Foundation*) апликацију и на тај начин омогућавамо дизајн визуелно напредног и интерактивног корисничког интерфејса.

3.1.2 C# програмски језик

C# је објекно-оријентисан програмски језик развијен у оквиру .NET платформе [9]. Познат је по својој једноставној синтакси, читљивости и великој продуктивности, због чега је данас један од најпопуларнијих програмских језика за развој десктоп и веб апликација.

Неки од главних концепата објектно-оријентисаног програмирања који су коришћени у развоју ове апликације су:

1. Објекти и класе – коришћене за моделовање порука и корисника
2. Енкапсулација – за заштиту података и скривање интерне логике
4. Наслеђивање и полиморфизам – омогућава флексибилност и проширивост кода

C# је коришћен за имплентацију логике система, симулацију IMAP протокола, управљање порукама, као и управљање комуникацијом између различитих делова система.

3.1.3 WPF

WPF (*Windows Presentation Foundation*) је технологија дизајнирана за развој графичког корисничког интерфејса на *Windows* оперативним системима. Она нам омогућава да раздвојимо логику апликације од визуелног приказа коришћењем **XAML** (*eXtensible Application Markup Language*) датотеке[10].

На овај начин нам поједностављује одржавање програма и омогућава леп и функционалан дизајн корисничког интерфејса.

У овој апликацији WPF је коришћен за креирање свих главних прозора од којих је сачињена апликација:

1. **MainWindow.xaml** – прозор за логовање
2. **Send.xaml** – прозор за слање порука
3. **Receiver.xaml** – прозор за пријем и преглед порука
4. **Message.xaml** – прозор за приказ садржаја поруке

3.2 Коришћени алати

3.2.1 Visual studio 2022

Развој апликације је реализован помоћу *Visual Studio 2022*окружења који нам пружа сигуран и брз приступ и представља развојно окружење са широким спектром проширења и подршком за различите програмске језике подржане у оквиру .NET развојног окружења. Подржава истовремени рад са више пројеката у оквиру једног развојног окружења. Развијен је од стране *Microsoft-a*, па самим тим често добија како безбедносне тако и функционалне надоградње. *Visual*

Studio такође подржава да више програмерских тимова ради на једном пројекту, што нам омогућава дистрибуираност система на високом нивоу [11].

Visual Studio нам омогућава:

1. Аутоматско компајлирање и детекцију грешака
2. Интегрисану подршку за XAML и C#
3. Преглед дизајна у реалном времену
4. Једноставно управљање билиотекама.

У току развоја апликације нису коришћене спољне библиотеке преко **NuGet** пакет менаџера, већ само .NET библиотеке које су интегрисане у оквиру WPF-а.

4.Опис решења проблема

У овом поглављу ћемо дати детаљан опис структуре, функционалности и логике рада апликације која је развијена са циљем да симулира рад протокола за пријем електронске поште.Циљ наше апликације јесте да нашим корисницима омогући да на једноставан начин управљају поштом коју примају. Корисници имају могућност да читају поруке, да их бришу, али и да прочитане поруке означе као непрочитане и тако их пребаце у листу непрочитаних порука.

У наставку можемо видети детаљан опис рада ове апликације као и начин на који је она имплементирана да на успешан начин симулира IMAP протокол и управља порукама у систему.

Апликација представља поједностављен клијент-сервер модел, где WPF интерфејс (клијент) комуницира са симулатором који је имплементиран кроз неколико класа које опонашају рад IMAP сервера.

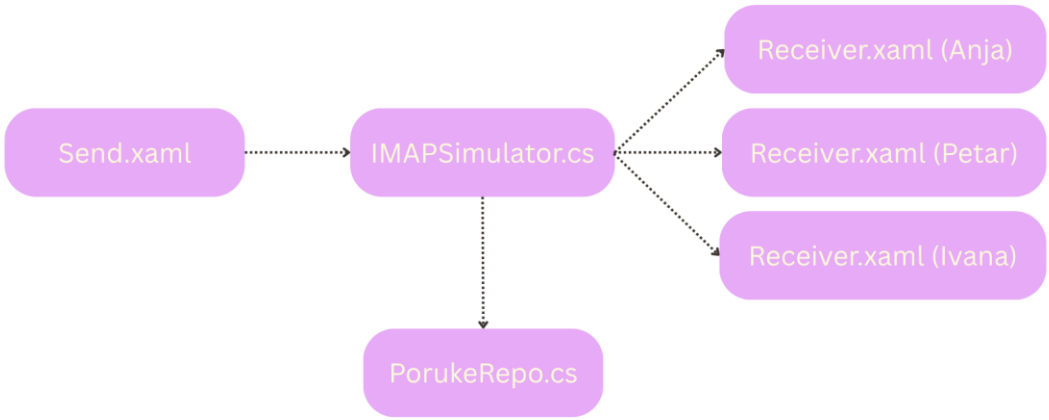
4.1 Структура апликације

Систем се састоји од два главна нивоа:

1. WPF клијент (кориснички интерфејс)- задужен за интеракцију корисника и приказаног садржаја. Састоји се од следећих прозора:
    - **MainWindow** – прозор за пријављивање корисника
    - **Send**–прозор који омогућава слање порука корисницима
    - **Receiver**–прозор за приказ свих примљених порука
    - **Message** – прозор за приказ садржаја одређене поруке
2. Логички ниво (*backend*)- његова улога је да опонаша рад IMAP сервера. Састоји се од следећих класа:
    - **Poruka.cs** – моделује једну поруку
    - **PorukeRepo.cs** – складишти примљене поруке (*mailbox*)
    - **IMAPSimulator.cs** – класа која симулира основне IMAP операције (преузимање порука, брисање, модификација)

Ова структура опонаша рад праве комуникације између клијента и сервера која се користи у системима електронске поште.

Дијаграм структуре рада апликације је приказан на слици 4.1.1.



Слика 4.1.1 Дијаграм структуре рада апликације

4.2 Графички кориснички интерфејс

Графички кориснички интерфејс је реализован помоћу WPF-а (*Windows Presentation Foundation).* Он нам омогућава да креирамо савремена и иновативна окружења за наше кориснике. Сваки прозор у овој апликацији је специјално осмишљен и дизајниран тако да одговара једном кораку у симулацији IMAP протокола, као што су пријава корисника, слање и пријем порука и приказ садржаја селектоване поруке.

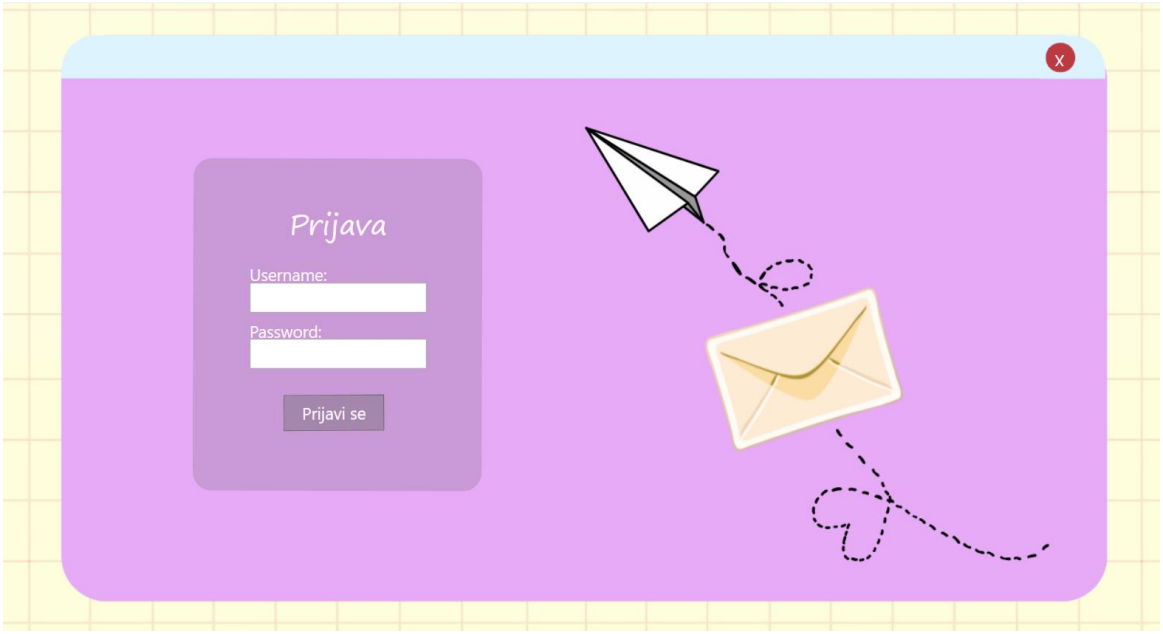
Дизајн интерфејса је фокусиран на једноставно и интуитивно коришћење. Прозори садрже елементарне графичке компоненте као што су:

1. поља за унос текста
2. дугмад
3. листе порука
4. падајући мени.

Они нам омогућавају једноставну и лаку навигацију кроз различите делове и функционалности апликације.

4.2.1 MainWindow – прозор за логовање

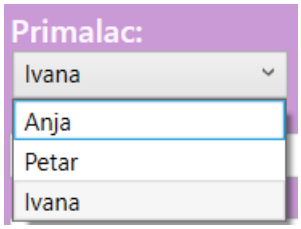
Почетна тачка апликације јесте прозор за пријаву корисника (слика 4.2.1.1). Његова сврха је да аутентификује кориснике пре него што им дозволи приступ систему. На основу пријављеног корисника, систем их даље усмерава или на прозор *Send* (у случају да је пријављени корисник ,,Пошиљалац") или на прозор *Receiver* (у случају да је пријављен неко од прималаца нпр. ,,Ања", ,,Петар" или ,,Ивана").



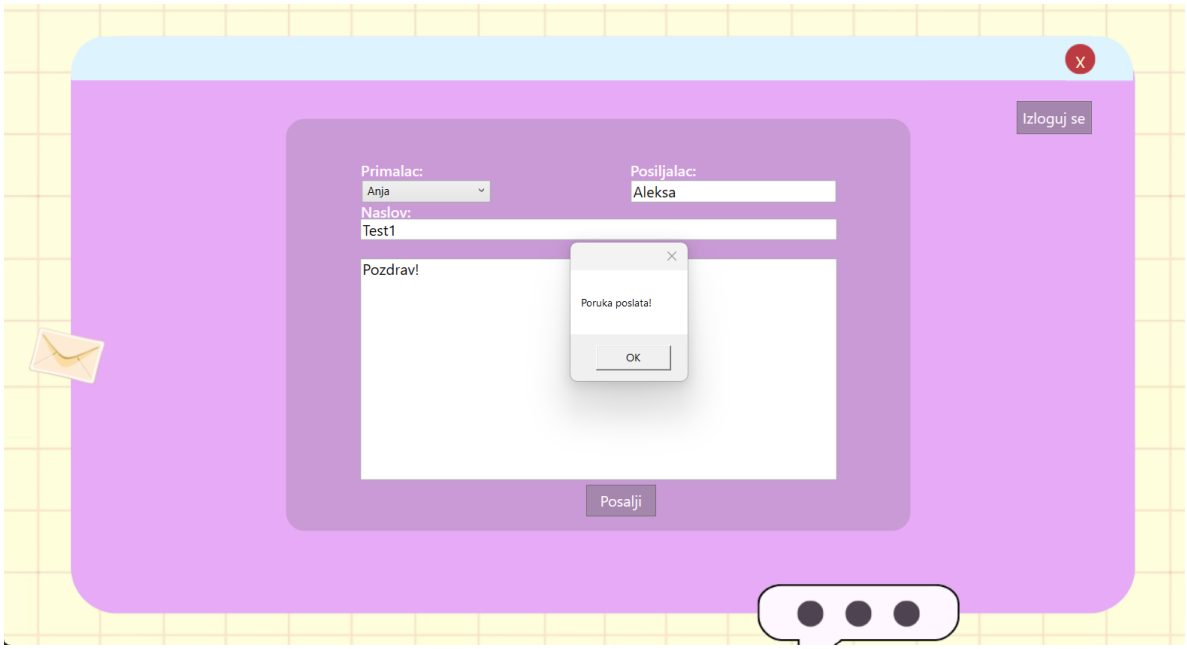
Слика 4.2.1.1 Приказ прозора за пријаву корисника (MainWindow)

4.2.2 Send Window

Овај прозор омогућава пошиљаоцу да саставља нове поруке и шаље их некоме од понуђених примаоца. Састоји се из падајуће листе на којој се налазе сви примаоци (слика 4.2.2.1), поља упис имена пошиљаоца, поља за наслов и поља за текст поруке. Када корисник кликне на дугме Sendдобија се обавештење да је порука успешно послата. А затим се формира нова порука и смешта на централни репозиторијум (*PorukeRepo*). Поруке тада постају доступне свим ауторизованим примаоцима. Приказ прозора за слање порука можете видети на слици 4.2.2.2.



Слика 4.2.2.1 Приказ падајуће листе прималаца



Слика 4.2.2.2 Приказ прозора за слање порука (Send)

4.2.3 Receiver Window

Прозор за пријем поште представља један од кључних елемената апликације, јер баш он симулира понашање IMAP пријемног сандучета.

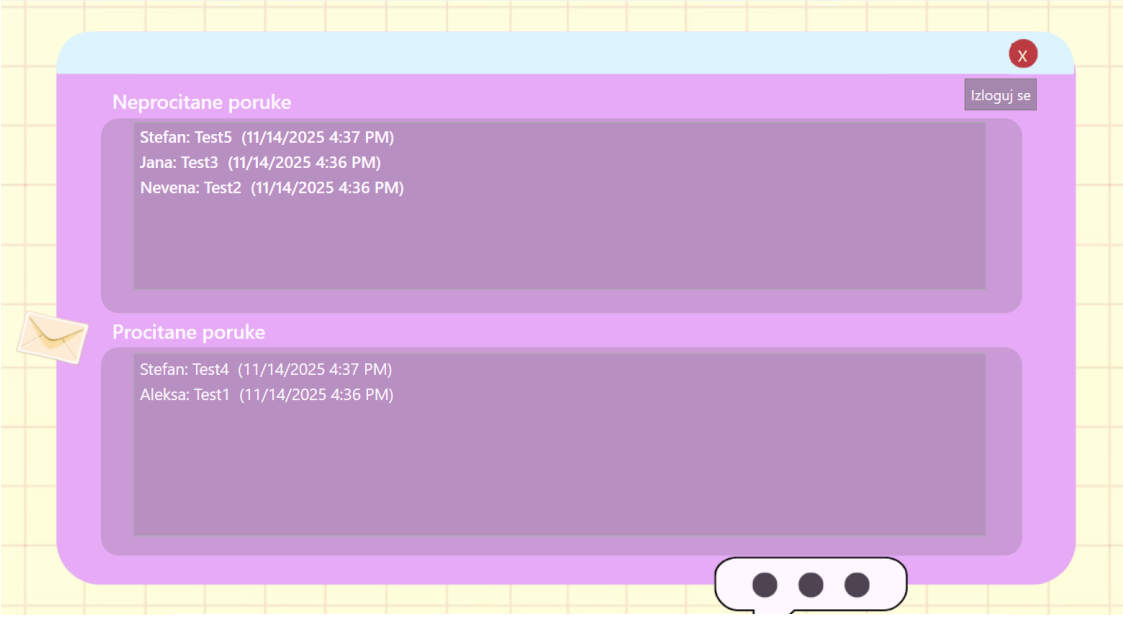
Функционалности:

1. Приказ листе непрочитаних порука
2. Приказ листе прочитаних порука
3. Отварање порука које се налазе у једној од листа.

Поруке се преузимају помоћу функције **IMAPSimulator.GetInbox()**, која враћа све поруке за тренутно улогованог корисника, сортиране по времену и датуму слања у опадајућем редоследу.

Овај прозор симулира понашање IMAP клијента, код ког се пристигла пошта (*inbox*) динамички синхронизује након сваке промене у систему (читање, брисање, означавање прочитане поруке као непрочитане).

Пример прозора за пријем поште је приказан на слици 4.2.3.1.

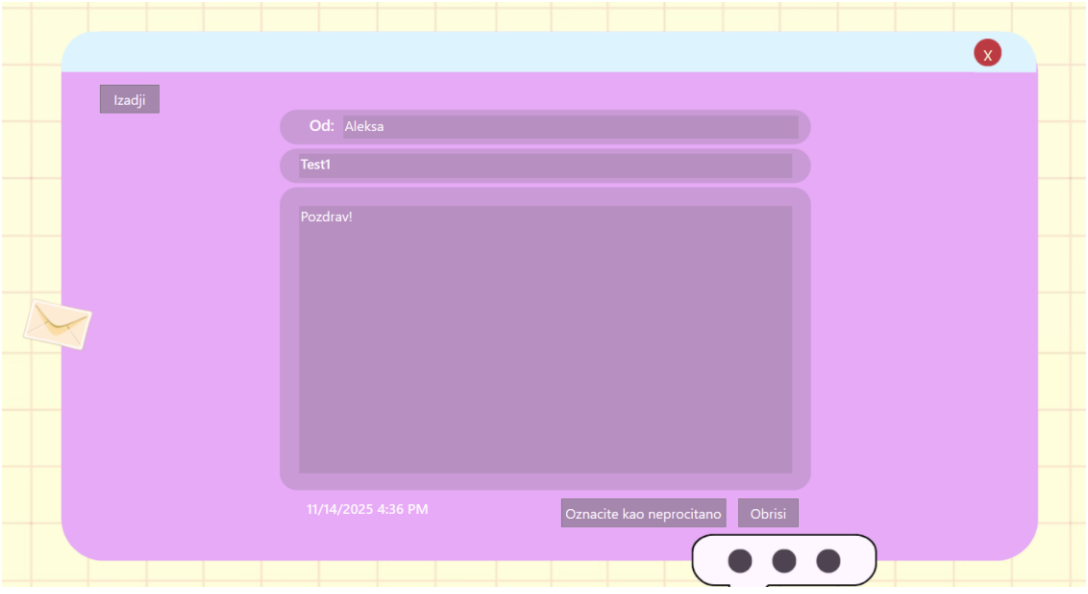


Слика 4.2.3.1 Приказ прозора за пријем поште (Receiver)

4.2.4 Message Window

Овај прозор нам омогућава да видимо комплетан садржај жељене поруке.Приказује нам име пошиљаоца, наслов поруке, садржај поруке, време пријема поруке, али нам поред тога даје могућност да извршимо неке акције над поруком као што су брисање поруке и премештање у листу непрочитаних порука. Након што изађемо из прозора за приказ поруке прозор *Receiver* ће аутоматски синхронизовати своје листе за прочитане и непрочитане поруке.

Све ове операције су извршене позивањем метода из класе *IMAPSimulator*која ажурира централни репозиторијум у ком су смештене поруке.



Слика 4.2.4.1 Приказ прозора за приказ садржаја поруке (Message)

4.3 Структура кода и симулација рада IMAP протокола

У овом делу нам је приказан детаљан опис свих класа које су кључне за симулацију IMAP протокола. Садржи најбитније делове кода и објашњава шта се дешава у позадини иза корисничког интерфејса. Ово је кључни део пројекта који садржи логику која симулира комуникацију између IMAP клијента и IMAP сервера.

4.3.1 Модел података – класа *Poruka*

Класа *Poruka* креира једну електронску поруку тако што моделује све кључне атрибуте које прави IMAP сервер памти за све поруке, као што је приказано на слици 4.3.1.

```
public class Poruka
{
    3 references
    public string From { get; set; }
    2 references
    public string To { get; set; }
    3 references
    public string Naslov { get; set; }
    2 references
    public string Sadrzaj { get; set; }
    5 references
    public bool Procitana { get; set; } = false;
```

Слика 4.3.1Имплементација класе Poruka

Објашњење:

1. Променљива ***Procitana***симулира IMAP команде *STORE +FLAGS*
2. ***DatumSlanja*** памти тренутак у ком је сервер примио поруку
3. Објекти ове класе се чувају на серверској страни репозиторијума ***PorukeRepo***

Овај начин је идентичан начину на који IMAP сервер чува поруке у својим *mailstore* системима.

4.3.2 Репозиторијум за складиштење порука – класа *PorukeRepo*

Ова класа симулира део сервера задужен за складиштење порука. Уместо коришћења праве базе података (нпр. *MySQL*) подаци се чувају помоћу статичке листе (слика 4.3.2.).

```
public static class PorukeRepo
{
    3 references
    public static ObservableCollection<Poruka> SvePoruke { get; set; } = new ObservableCollection<Poruka>();
}
```

Слика 4.3.2Имплементација класе PorukeRepo

Зашто баш *ObservableCollection*?

Зато што нам *ObservableCollection* омогућава да:

1. Аутоматски обавестимо UI о променама које су се догодиле
2. WPF ажурира листе у реалном времену
3. Симулирамо IMAP *push-update* механизам.

У IMAP-у сервер обавештава клијента о свим променама (нова порука, обрисана порука, промена статуса). Овде нам све то омогућава *ObservableCollection*.

4.3.3 IMAPSimulator – симулација рада IMAP сервера

IMAPSimulator представља централну класу овог пројекта.

Она симулира следеће IMAP команде:

IMAP команда	Функција	Симулација у пројекту
LOGIN	аутентификација	Врши се у *MainWindow*
FETCH	преузимање порука	*GetInbox()*
STORE +FLAGS	означи као прочитано	*MarkAsRead()*
STORE –FLAGS	означи као непрочитано	*MarkAsUnread()*
EXPUNGE	обриши поруку	*Delete()*

Табела 4.3.3Табела IMAP команди

Имплементација ове класе приказана је на слици 4.3.3.

```
public static class IMAPSimulator
{

    1 reference
    public static List<Poruka> GetInbox(string korisnik)
    {
        return PorukeRepo.SvePoruke
            .Where(p => p.To == korisnik)
            .OrderByDescending(p => p.DatumSlanja)
            .ToList();
    }


    1 reference
    public static void MarkAsRead(Poruka p)
    {
        p.Procitana = true;
    }


    1 reference
    public static void MarkAsUnread(Poruka p)
    {
        p.Procitana = false;
    }


    1 reference
    public static void Delete(Poruka p)
    {
        PorukeRepo.SvePoruke.Remove(p);
    }
}
```

Слика 4.3.3 Имплементација класе IMAPSimulator

Објашњење:

Метода *GetInbox()* враћа листу порука за примљеног корисника. Она те поруке преузима из репозиторијума у ком су смештене све поруке, проверава назив примаоца , а затим их сортира по редоследу пријема поруке.

Метода *MarkAsRead()* нам омогућава да поруку означимо као прочитану тако што мења вредност променљиве *Procitana* у *true*.

Метода *MarkAsUnread()* нам омогућава да поруку означимо као непрочитану тако што мења вредност променљиве *Procitana* у *false*.

Метода *Delete()* брише означену поруку из репозиторијума.

4.3.4 Логика прозора за слање порука Send.xaml.cs

Део апликације приказан на слици 4.3.4 симулира слање порука помоћу SMTP протокола, након чега се поруке одмах смештају на импровизовани IMAP сервер.

```
private void SendBtn_Click(object sender, RoutedEventArgs e)
{
    Poruka nova = new Poruka
    {
        From = FromBox.Text,
        To = (PrimalacCombo.SelectedItem as ComboBoxItem).Content.ToString(),
        Naslov = NaslovBox.Text,
        Sadrzaj = PorukaBox.Text,
        Procitana = false
    };

    PorukeRepo.SvePoruke.Add(nova);
    MessageBox.Show("Poruka poslata!");
}
```

Слика 4.3.3 Имплементација дела класе Send.xaml.cs

Објашњење:

Кликом на дугме *,,Posalji''* креира се нови објекат *Poruka* који се смешта на репозиторијум *PorukeRepo*. Корисник који прима поруку може одмах да је види у свом сандучету у листи непрочитаних порука.

4.3.5 Логика прозора за примање порука Receiver.xaml.cs

Receiver.xaml.cs симулира *IMAP FETCH* команду сваки пут кад се прозор учита или рефрешује.

```
public void UpdateLists()
{
    var inbox = IMAPSimulator.GetInbox(Korisnik);

    NeprocitaneLst.ItemsSource = inbox.Where(p => !p.Procitana).ToList();
    ProcitaneLst.ItemsSource = inbox.Where(p => p.Procitana).ToList();
}
```

Слика 4.3.5.1 Имплементација дела класе Receiver.xaml.cs

Објашњење:

Позивањем методе *UpdateLists()* креирамо пријемно сандуче одређеног корисника. То реализујемо тако што позовемо класу IMAPSimulator и њену методу *GetInbox()* којој проследимо одређеног корисника. Након што креирамо пријемно сандуче поруке разврставамо у две листе (непрочитане и прочитане поруке) тако што проверамо вредност променљиве *Procitana*. Уколико променљива *Procitana* враћа вредност *false* поруку смештамо у листу непрочитаних порука, а уколико враћа вредност *true* поруку смештамо у листу прочитаних порука.

Код за приказ садржаја поруке приказан је на слици 4.3.5.2.

```
private void NeprocitaneLst_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    if (NeprocitaneLst.SelectedItem is Poruka p)
    {
        this.Hide();
        Message msgWindow = new Message(p, this);
        msgWindow.ShowDialog();
        this.Show();
        UpdateLists();
    }
}
```

Слика 4.3.5.2 Имплементација дела кода за отварање поруке класе Receiver.xaml.cs

Објашњење:

Двокликом на неку од порука која се налази у листи непрочитаних порука отварамо прозор који нам приказује цео садржај поруке. Након сваког отварања поруке позивамо методу *UpdateLists()* како би ажурирали листе, јер након отварање неке поруке може доћи до промене њеног статуса. Исти код важи и за листу прочитаних порука.

4.3.6 Логика прозора за приказ садржаја поруке Message.xaml.cs

Овај прозор симулира следеће IMAP команде:

1. **STORE** – означи као прочитано
2. **STORE –FLAGS** – означи као непрочитано
3. **EXPUNGE** – брисање поруке

Означавање поруке прочитаном реализовано је у оквиру методе приказане на слици 4.3.6.1.

```
private void NazadBtn_Click(object sender, RoutedEventArgs e)
{
    IMAPSimulator.MarkAsRead(trenutnaPoruka);
    receiverWindow.UpdateLists();
    this.Close();
}
```

Слика 4.3.6.1 Имплементација дела кода за повратак на прозорReceiver.xaml.cs

Објашњење:

Кликом на дугме *,,Izadji''* позивамо методу *MarkAsRead()* која се налази у класи *IMAPSimulator* и помоћу ње мењамо статус поруке у прочитану. Наконтога је потребно да позовемо методу *UpdateLists()* како би нам се порука преместила из листе непрочитаних у листу прочитаних порука.

Означавање поруке као непрочитано реализовано је у оквиру методе приказане на слици 4.3.6.2.

```
private void UnreadBtn_Click(object sender, RoutedEventArgs e)
{
    IMAPSimulator.MarkAsUnread(trenutnaPoruka);
    receiverWindow.UpdateLists();
    this.Close();
}
```

Слика 4.3.6.2 Имплементација дела кода за означавање поруке непрочитаном Message.xaml.cs

Објашњење:

Кликом на дугме *,,Oznacite kao neprocitano''* позивамо методу *MarkAsUnread()* која се налази у класи *IMAPSimulator* и помоћу ње мењамо статус поруке у непрочитану. Након тога је потребно да позовемо методу *UpdateLists()* како би нам се порука преместила из листе прочитаних у листу непрочитаних порука.

Брисање поруке реализовано је у оквиру методе приказане на слици 4.3.6.3.

```
private void ObrisiBtn_Click(object sender, RoutedEventArgs e)
{
    IMAPSimulator.Delete(trenutnaPoruka);
    receiverWindow.UpdateLists();
    this.Close();
}
```

Слика 4.3.6.3 Имплементација дела кода брисање поруке Message.xaml.cs

Објашњење:

Кликом на дугме *,,Obrisi''* позивамо методу *Delete()* која се налази у класи *IMAPSimulator*и помоћу ње бришемо поруку из листе примљених порука. Након тога позивамо методу *UpdateLists()* како би ажурирали листе које се налазе на прозору за пријем порука и аутоматски се враћамо на тај прозор.

4.4 Ток рада апликације

Ток рада апликације прати логику рада правог система електронске поште који користи IMAP протокол. У наставку је описан корак по корак процес који корисник пролази од тренутка пријаве до управљања порукама.

1. Пријава корисника *MainWindow.xaml.cs*
    - Корисник уноси корисничко име и лозинку
    - На основу улоге (пошиљалац или прималац), систем упућује корисника на одговарајући прозор
2. Слање поруке *Send.xaml.cs*
    - Пошиљалац уноси све потребне информације (адресу примаоца, наслов, текст поруке)
    - Кликом на дугме ,,*Posalji*'' креира се нови објекат класе *Poruka* и чува се на репозиторијуму
3. Складиштење порука *PorukeRepo.cs*
    - Све поруке се додају у *ObservableCollection* који представља *mailbox* IMAP сервера
    - Свака порука постаје доступна кориснику након њеног додавања
4. Преузимање порука *Receiver.xaml.cs*
    - Када се корисник улогује прозор позива *IMAPSimulator.GetInbox(),* који учитава све поруке тог корисника
    - Поруке се разврставају у две листе прочитане и непрочитане
5. Преглед садржаја поруке *Message.xaml.cs*
    - Дуплим кликом на жељену поруку отвара се прозор који приказује читав садржај поруке (информације о пошиљаоцу, наслов поруке, текст поруке, датум и време пријема поруке)

6. Промена статуса поруке *IMAPSimulator.cs*
    - Корисник може да прочита поруку, да је означи као непрочитану или обрише
    - *IMAPSimulator* користи методе *MarkAsRead(),MarkAsUnread(), Delete()*
    - Све промене се одмах примењују на прозор за пријем порука, што нампредставља IMAP концепт синхронизације у реалном времену.

4.5 Резултати и могућа проширења

Развојем ове апликације смо успешно демонстрирали основне принципе рада IMAP протокола и омогућили јасан увид у рад савремених система за пријем електронске поште

4.5.1 Постигнути резултати

Пројекат успешно имплементира:

1. Симулацију приступа порукама путем IMAP протокола – поруке се складиште на серверу (*PorukeRepo*), што нам омогућава читање порука, ажурирање и брисање.
2. Синхронизација у реалном времену – свака промена се одмах приказује примаоцу као и код правог IMAP протокола
3. Подржава више различитих корисничких улога – систем разликује примаоца и пошиљаоца , свако има свој интерфејс и различите функционалности
4. Једноставан савремен кориснички интерфејс (WPF)
5. Приказ метаподатака поруке – увођењем датума и времена смо добили могућност сортирања порука.

4.5.2 Могућа проширења

У даљем развоју апликације могуће је имплементирати нека побољшања:

1. Повезивање са стварним IMAP сервером – ово би нам омогућило преузимање порука са удаљених сервера
2. Повезивање на базу података – *SQLite*, *MySQL*би нам омогућилотрајно чување података

3. Имплементација SSL/TSL безбедности – шифровање канала за комуникацију би симулирало реалне безбедносне стандарде
4. Додавање прилога уз поруке – корисници би били у могућности да шаљу слике, документе и остале фајлове
5. Више фолдера (Пријем, Слање, Обрисане поруке,..) – проширење структуре пријемног сандучета на ниво стварног IMAP клијента
6. Додавање напредних IMAP команди – на пример **SEARCH, MOVE, IDLE**…

5. Закључак

Развој и функционалност система електронске поште представља једанод основних компоненти савремених информационих технологија. Данас, када је комуникација путем интернета постала саставни део свакодневног живота, разумевање рада ових система је од кључног значаја, како у смислу техничког знања тако и у примени у разним софтверским системима. У оквиру овог рада представљена је апликација која симулира основне операције IMAP протокола, једног од најпознатијих протокола за преузимање и управљањем електронском поштом.

Главна сврха овог пројекта јесте да представи како се поруке чувају, преузимају, означавају и бришу на серверском нивоу, као и да опонаша интеракцију између клијента и сервера у систему електронске поште. Решење проблеме које смо имплементирали успешно приказује како IMAP ради у пракси и како омогућава синхронизацију на више различитих уређаја. Кроз ову симулацију корисник стиче знање о разлици између локалне обраде порука и централизованог приступа, који се данас највише користи у савременим системима као што су *Gmail, Outlook*, и др.

Апликација је развијена коришћењем савремних технологија као што су C#, .Net платформа и WPF коришћен за израду корисничког интерфејса. Посебан акценат је стављен на развој *Poruka, PorukaRepo* и *IMAPSimulator* класа, које формирају логику система. Ове класе имплементирају симулацију свих IMAP операција као што су *FETCH*, *STORE*, *EXPUNGE* и тако нам омогућава јасније разумевање како протокол управља порукама на серверу.

Поред техничких аспеката, овај пројекат нам пружа значајну улогу у образовању. Симулација омогућава почетницима у програмирању да на најједноставнији начин схвате комплексне протоколе за комуникацију. Уместо да уче о IMAP протоколу само кроз теорију, сада могу да кроз ову апликацију прате податке кроз мрежу и виде како се то све одвија у позадини.

Иако тренутна верзија ове апликације испуњава све основне циљеве симулације понашања протоколза преузимање електронске поште, она нам оставља доста простора за разне надоградње и проширења. Главни циљ у даљем развоју јесте повезивање симулације са стварним IMAP сервером. На тај начин би нам било омогућено преузимање порука са интернета и на тај начин би апликација била проширена на ниво правог клијент-сервер решења. Такође, имплементација базе података би омогућила трајно складиштење порука и корисничких налога. Функционалност апликације би била знатно побољшана додавањем додатних фолдера и имплементацијом напредних IMAP команди.

Укратко, ако узмемо у разматрање све што смо претходно навели, пројекат је успешно испунио своју сврху – пружање практичне симулације рада протокола

за преузимање електронске поште, као и развој функционалне WPF апликације која опонаша рад механизма система електронске поште. Овај рад нам је поред практичне симулације пружио и едукативни садржај као и искуство у раду са .Net платформом, развојем графичког интерфејса, моделовањем података и разумевање комуникационих протокола.

Литература

[1] М. Бојанић (2022). Апликативни слој [PowerPoint slides]. Доступно на: https://www.eepsi.ftn.uns.ac.rs/group/primena-racunarskih-mreza-u-infrastrukturnim-sis/custom (приступљено: 10.11.2025.)

[2] GeeksforGeeks (2025). Client-Server Architecture- System Design [Online]. Доступно на: https://www.geeksforgeeks.org/system-design/client-server-architecture-system-design/ (приступљено: 10.11.2025.)

[3] Hostragons (2025). Šta su IMAP i POP3? Koje su razlike? [Online]. Доступно на:

https://www.hostragons.com/bs/blog/koje-su-razlike-izmedu-imap-a-i-pop3-protokola/# (приступљено: 11.11.2025.)

[4] SiteGround (2025). What are Email Protocols (POP3, SMTP and IMAP) and their default ports [Online]. Доступно на: https://www.siteground.com/tutorials/email/protocols-pop3-smtp-imap/ (приступљено: 11.11.2025.)

[5] CloudFlare (2025). What is TLS (Transport Layer Security)? [Online]. Доступно на: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls (приступљено: 14.12.2025.)

[6] Proton (2021). What is the best encryption for emails? [Online]. Доступно на:

https://proton.me/blog/best-encryption-for-emails (приступљено: 14.12.2025.)

[7] NordLayer (2024) . MFA vs 2FA: what's the difference? [Online]. Доступно на:https://nordlayer.com/blog/mfa-vs-2fa-whats-the-difference (приступљено: 14.12.2025.)

[8] Microsoft.com (2025). .Net documentation [Online]. Доступно на: https://learn.microsoft.com/en-us/dotnet/ (приступљено: 13.11.2025.)

[9] Microsoft.com (2025). C# documentation [Online]. Доступно на: https://learn.microsoft.com/en-us/dotnet/csharp/ (приступљено: 13.11.2025.)

[10] Microsoft.com (2025). Windows Presentation Foundation documentation [Online]. Доступно на: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-8.0 (приступљено: 13.11.2025.)

[11] Microsoft.com (2025). Visual Studio documentation [Online]. Доступно на: https://learn.microsoft.com/en-us/visualstudio/windows/?view=visualstudio (приступљено: 13.11.2025.)

Списак коришћених табела и слика