Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Чумак О.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Чумаку Олексію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Чумак О.В. Дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Метою розробки є дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Об'єктом дослідження є процес комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Предметом дослідження є методи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Методи дослідження базуються на методах теорії телетрафіку, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: комп'ютерні науки, миттєвий обмін повідомленнями

ABSTRACT

Chumak O.V. Research and software implementation of a user switching system based on a centralized instant messaging service. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software is developed, which is intended for a user switching system based on a centralized instant messaging service.

The purpose of the development is the research and software implementation of a user switching system based on a centralized instant messaging service.

The object of research is the process of user switching based on a centralized instant messaging service.

The subject of research is user switching methods based on a centralized instant messaging service.

Research methods are based on teletraffic theory methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of a user switching system based on a centralized instant messaging service.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: computer science, instant messaging

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ОС	–	операційна система
BMP	–	графічний формат
JPEG	–	графічний формат
GIF	–	графічний формат
IP	–	інтернет протокол
OSCAR	–	протокол обміну інформацією
P2P	–	peer-to-peer – безпосереднє інтернет-з'єднання двох комп'ютерів, минаючи сервер
SQL	–	мова управління базами даних
UIN	–	Universal Identification Number – унікальний для кожного облікового запису номер

ВСТУП

Актуальність теми. Мессенджери, або програмні засоби комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями одержали поширення з появою першої програми такого роду ICQ, що залишається понині самою популярною. У назві за рахунок подібного звучання зашифрована фраза «I seek you» (я шукаю тебе). Творці програми по сумісництву є й розроблювачами протоколу Oscar, що став основним стандартом для обміну миттєвими повідомленнями. Але й у програми й у протоколу швидко з'явилися безліч конкурентів. А самі програми згодом стали обростати додатковими функціями. Серед основних зараз пропонуються: робота з поштою, інтеграція із соціальними мережами, голосове спілкування, передача файлів, гри й т.д. Якщо не бажаєте встановлювати програму-клієнт на комп'ютер, можна знайти онлайнові клієнти, що працюють через веб-інтерфейс.

У 2012 році ICQ та інші месенджери втратили аудиторію через соцмережі, а в 2021-му WhatsApp став популярнішим за Facebook.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

– Дослідження системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

– Програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Об'єктом дослідження є процес комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Предметом дослідження є методи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Методи дослідження базуються на методах теорії телетрафіку, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

– Розроблено вітчизняний продукт комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Месенджер (від англ. Messenger – кур'єр, посланник) – це програма для миттєвого обміну текстовими повідомленнями, аудіозаписами, фотографіями та іншими мультимедіа. Програми встановлюються на комп'ютер, смартфон, планшет та працюють через інтернет.

Першим месенджером у світі став EMISARI (Emergency Management Information Systems and Reference Index – Інформаційні системи управління надзвичайними ситуаціями та довідковий індекс). Його створив фізик та математик Мюррей Турофф для уряду США у 1971 році. За допомогою EMISARI державні службовці могли швидко зв'язуватися. Вони підключалися до мережі міжміськими телефонними лініями через телетайпи – електромеханічні друкарські машини. Так влада США могла швидко реагувати на кризові ситуації в країні, але звичайні громадяни EMISARI не використали.

Першим месенджером, доступним для цивільних осіб, став Internet Relay Chat. Його розробив фінський програміст Яркко Ойкарінен у 1988 році. Програма була популярна в Європі та Північній Америці – у 2009 році на серверах спілкувалося понад 500 тис. користувачів.

У 1996 році ізраїльська компанія Mirabilis розробила месенджер ICQ. Через нього спілкувалися користувачі з Росії та інших країн – на початок 2010 року кількість активних облікових записів становила 47,9 млн. доларів. З розвитком соціальних мереж популярність ICQ знизилася – у 2012 році аудиторія скоротилася на 30,9%.

Великі компанії почали розробляти свої месенджери. У 1997 році з'явився AIM, який був популярний у США. 1998-го – Yahoo! Messenger, 2002-го – iChat від Apple, Talk від Google.

1.2 Область застосування

Соціальні мережі, як і месенджери, призначені для спілкування. Однак у перших ширший набір функцій: вони поєднують у собі чат, сервіс знайомств, стрічку новин і особистий блог в одному додатку. У соцмережах можна спілкуватися через відкриті спільноти, публікувати записи, викладати фото та відео, слухати музику.

Деякі соціальні мережі випускають окремі програми-месенджери, пов'язані з основним сайтом. Наприклад, так вчинив Facebook.

Месенджери призначені для приватного спілкування між кількома людьми. Ці програми не перевантажені іншими функціями, тому їх часто використовують для роботи, щоб не відволікатися на стрічку новин і розважальний контент. Однак у популярних месенджерах також є спільнота. У Telegram та WhatsApp будь-який користувач може створити канал і публікувати в ньому записи, які прочитають інші.

У 2020 році послуги для обміну повідомленнями стали на 20% популярнішими, ніж соцмережі. За даними дослідження креативного агентства ZAK, користувачі віддають перевагу месенджерам, оскільки це більш закритий простір. 43% опитаних у віці від 16 до 30 років вважають, що в Instagram та Facebook «занадто багато людей»: будь-який користувач може зайти на твою сторінку та побачити особисті фотографії, записи. А ось у месенджерах можна вибирати, якою інформацією ділитися і з ким.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Для огляду я вибрав останні версії найвідоміших і затребуваних служб обміну повідомленнями.

ICQ

Почнемо із засновника. Остання версія позначається вже 8-м номером і може похвастатися інтеграцією різних соціальних мереж і сервісів: Facebook, Twitter, YouTube, Flickr і ще парочки не настільки відомих.

Рисунок 2.1 – Інтерфейс користувача ICQ

У Європі програма вже давно втратила провідне місце. Наприклад, в Америці ICQ із упевненістю обганяють мессенджери від Microsoft, Yahoo! і Google.

Також програма стала швидше працювати й менше займати оперативної пам'яті, а от недолік, залишився – рекламні баннери прямо в клієнті.

ICQ була випущена ізраїльською компанією Mirabilis, однак уже через 2 роки була продана американської компанії AOL за $287 млн. Незважаючи на те, що власники ICQ інвестували в проект $125 млн (майже половину вартості ICQ), так і не вдалося заробити пристойну суму й заволодіти серйозною часткою на американському IT-ринку.

А от справжні лідери ринку. Miranda і QIP – як Firefox і Opera серед браузерів: один дозволяє все налаштувати, другий споконвічно укомплектований по повній.

Miranda

Ядро Міранди, властиво програма-клієнт, представляє із себе самий невимогливий до ресурсів і малофункціональний мессенджер. Але в такому виді її ніхто не використовує.

На комп'ютерах користувачів Миранда існує у вигляді збірок – навішених на ядро численних плагінів, які дозволяють налаштувати все від зовнішнього вигляду спливаючих повідомлень до відображення музики, що прослуховується. Таких плагінів величезна кількість, зацікавлені люди роблять із них збірка – завдання непроста, окремі деталі можуть конфліктувати один з одним, адже створюються різними розроблювачами – і викладають на спеціальних сайтах. От найвідоміший з них на просторах Рунета – Miranda-Planet. Кожна збірка – по суті самостійний мессенджер.

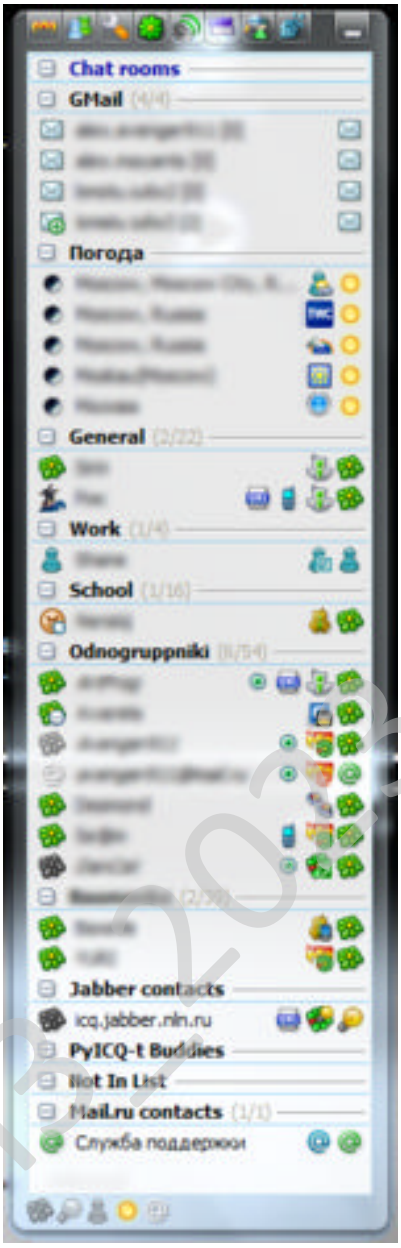

Рисунок 2.2 – Інтерфейс користувача Miranda

Але в цій конструктивності криється й головна проблема. У середньому стабільність збірок уступає стабільності монолітних мессенджерів. Якщо ви розберетеся в роботі із плагінами, то зможете забрати всі баги самостійно й створити для себе ідеальний мессенджер

QIP

Перший клієнт мессенджера QIP був випущений в 2005 році, що так і називається QIP 2005. Зараз найбільш популярний QIP InfiumP.

Рисунок 2.3 – Інтерфейс користувача QIP

Що мене у свій час віджахнуло від програми, це спрощений дизайн, що майже не налаштовується. Зміна тим є, але вони досить одноманітні – по суті міняють тільки фонові картинки, так шрифти.

QIP Online

Цей додаток є дітищем тієї ж компанії, що й попередній мессенджер. Виконує функції комунікації в соціальних мережах. Також вона повідомляє про нові листи на пошті й служить RSS-рідером. Незважаючи на назву є встановлюваним додатком.

Meebo

Цей інтернет-сервіс призначений для спілкування в різних мережах без установки програмного забезпечення. До веб-інтерфейсу можливо одержати доступ з будь-якого комп'ютера, маючи логін і пароль. Сервіс підтримує ICQ, GTalk, Yahoo, Jabber, Facebook, MySpace і деякі інші.

Відразу можна одержати собі на сайт панель інструментів (bar). Інструменти дозволяють вести чат у різних мережах, працювати з поштою, і публікувати інтернет-сторінку в різних соціальних службах. На жаль, ця частина Meebo на росіянин не переведений. Та й панель інструментів Wibiya перекриває Meebo функціонально в кілька разів.

ICQ2GO

Продовжуючи тему онлайнових замін мессенджерам, варто звернути увагу на такий сервіс від самої ICQ. Перейти до нього можна з офіційного сайту ICQ. Відкривається веб-версія в окремому вікні, імітуючи стандартний інтерфейс мессенджеру, і виглядає дуже приємно. Працює теж. З додаткових функцій тільки відправлення смс. І так, реклама представлена баннером, що легко блокується баннерорізкою. Я про нього спочатку взагалі не знав.

На жаль, і отут є свої мінуси. ICQ2GO не завжди вірно відображає кирилицю. При відправленні повідомлень із деяких мессенджерів приходить абракадабра.

Skype

Найвідоміший і не має конкурентів мессенджер, але у своїй області. Звичайно, обмін повідомленнями отут підтримується, але не настільки зручний, як в інших сервісах із цього огляду. Головний козир і функція Skype – IP-телефонія. Безкоштовні іншим власникам Skype або дуже дешеві дзвінки в будь-яку точку планети, от за що Skype відомий по усьому світі. Крім спілкування між двома абонентами можна влаштовувати конференції з необмеженим числом учасників, і навіть – відеоконференції. Зручний і гарний інтерфейс додається.

Google Talk

Як і все від Google радує приємним лаконічним дизайном. Функціональність обмежується обміном повідомлень, інтеграцією з поштою GMail і дзвінками іншим власникам Google Talk. При цьому мессенджер завоював високу популярність, десь обійшовши навіть ICQ. Справа в поширеності пошти Google, ну й у тому, що ті деякі функції, що були реалізовані, зроблені якісно.

До речі, Google Talk працює із протоколом Jabber – система обміну миттєвими повідомленнями й інформацією про присутність на основі відкритого протоколу XMPP.

Yahoo messenger

Непопулярний у нас, але вартий уваги мессенжер. Користується заслуженою популярністю в Європі. Остання 10 версія наголошує на IP-телефонії, стаючи конкурентом Skype. Користувачі відзначають всю зростаючу якість аудіо й відео переговорів. Крім того Yahoo Messenger інтегрується із численними сервісами Yahoo, наприклад, дозволяє слухати радіо або читати новини.

Live messenger

Мессенджер від Microsoft, що замінив в 2005 році MSN messenger. Свої плюси є, але за мінусами їх можна не розглянути.

WhatsApp

Рік створення: 2009.

Щомісячна аудиторія: 2,5 млрд. користувачів.

Кількість завантажень: 5 млрд в App Store та Google Play за даними на 2023 рік.

Згідно зі статистикою від липня 2023 року, WhatsApp – найпопулярніший месенджер у світі. Його створили колишні співробітники американської компанії Yahoo – Брайан Ектон та Ян Кум. У лютому 2014 року компанія Facebook викупила сервіс за $19 млрд.

У WhatsApp можна надсилати текстові та голосові повідомлення, редагувати фото та відео перед відправкою, створювати бесіди до 256 користувачів. Є режим повідомлень, що зникають.

Для власників малого бізнесу доступний сервіс WhatsApp Business. З ним можна створити в профілі каталог товарів і послуг, відстежувати замовлення, налаштувати автоматичні відповіді на запитання клієнтів, що часто ставляться.

Facebook Messenger

Рік створення: 2011.

Щомісячна аудиторія: 1,3 млрд. користувачів.

Кількість завантажень: 5 млрд в App Store та Google Play за даними на 2023 рік.

Messenger – це окремий сервіс Facebook для обміну повідомленнями. Додаток і соціальна мережа пов'язані: у яких відображаються одні й самі діалоги. Можна надсилати текстові та голосові повідомлення, створювати розмови до 250 користувачів, спілкуватись у секретних чатах.

У 2019 році засновник Facebook Марк Цукерберг заявив, що планує об'єднати Instagram, WhatsApp та Messenger. У 2020-му компанія почала робити перші кроки у цьому напрямі: деякі функції Messenger (наприклад, можливість вибрати оформлення чату) були додані до Instagram.

WeChat

Рік створення: 2011.

Щомісячна аудиторія: 1,24 млрд користувачів.

Кількість скачувань: точних даних немає, але в 2023 році додаток посідає перше місце за популярністю в Китаї.

Месенджер WeChat розроблений китайською компанією Tencent. Він не такий відомий у Росії, але популярний на батьківщині. У 2020 році WeChat був заблокований у США через те, що сервіс порушує конфіденційність даних: дії користувачів у додатку відстежуються. 2021-го заборону зняли з умовою, що на китайських серверах не буде персональної інформації американців.

У програмі можна надсилати текстові та голосові повідомлення, обробляти фото, перекладати тексти іншими мовами та оплачувати покупки через внутрішню платіжну систему.

Viber

Рік створення: 2010.

Щомісячна аудиторія: 823 млн. користувачів.

Кількість скачувань: 1,2 млрд в App Store і Google Play за даними на 2023 рік.

Месенджер створили ізраїльські розробники Талмон Марко та Ігор Магазинник. У березні 2014 року японська корпорація Rakuten Group викупила програму за $900 млн.

У Viber можна надсилати текстові та голосові повідомлення, створювати розмови до 250 користувачів. Є тариф Viber Out, який дозволяє здійснювати міжнародні дзвінки дешевше, ніж телефоном.

Телеграм

Рік створення: 2013.

Щомісячна аудиторія: 500 млн. користувачів.

Кількість завантажень: 540 млн в App Store та Google Play за даними на 2023 рік.

У Telegram можна надсилати текстові та голосові повідомлення, інші прослуховувати повільно або прискорено. Максимальна кількість учасників у розмові – 200 тис. Є режим секретного чату, доступного лише через пристрій, на якому розпочали листування.

Користувачі можуть створювати ботів – віртуальних помічників, які реагують на команди. Їх часто використовують у бізнесі, щоб автоматизувати виконання простих рутинних завдань.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Основні завдання для месенджера:

– текстовий чат користувач – користувач, зазвичай це лікар – лікар або лікар – медсестра;

– різні повідомлення користувачів, наприклад, лікаря про появу результатів аналізів або прибуття пацієнта до приймального відділення;

– текстовий чат у контексті пацієнта або випадку лікування, коли лікар може поставити якісь уточнюючі питання іншим лікарям з прив'язкою до лікування або пацієнту;

– передача файлів у повідомленнях;

– робота з історією повідомлень (пошук, перегляд, позначки про прочитання).

В перспективі:

– різні сценарії повідомлення, наприклад, повідомлення лікаря при зміні статусу талона на запис до поліклініки, розсилки по посадах, структурних підрозділах;

– «пейджер» – push-повідомлення лікаря на його особистий мобільний пристрій, якщо лікар не підключений до внутрішньої мережі;

– аудіо-відео конференції (консиліум);

– телемедичні консультації, у тому числі за участю пацієнта, авторизованого через Держпослуги.

Основні користувачі системи повідомлень на цьому етапі – лікарі та середній медперсонал. Пацієнти тут не задіяні, за винятком у перспективі їхньої участі в телемедичних консультаціях, авторизувавшись через Держпослуги.

Коли йдеться про якісь системи повідомлень, «олдфаги» згадують IRC та ICQ. Якщо треба «модно та молодіжно» – мова заходить про Discord та Slack. Шанувальники приватності беруть Matrix. Решта використовують Telegram. Начебто бери та користуйся. Але для наших цілей це все абсолютно не застосовується: медична інформація, з якою працюють лікарі, – це такий забористий коктейль із персональних даних та медичної таємниці, що потребує особливих підходів, зокрема:

– практично вся робота йде у захищеному контурі, куди немає доступу стороннім сервісам;

– персональні дані та інша чутлива інформація повинна зберігатися та оброблятися до;

– небажано використовувати якихось іноземних постачальників.

Загалом усе це різко обмежує набір можливих рішень.

А чого, власне, хочеться від месенджера:

– відкритість – Відсутність vendor lock in, відкритий код;

– контроль – можливість розгорнути self-hosted інсталяцію;

– наявність реалізацій (клієнтів) під основні мови (платформи), які у нас: Java, PHP, Node.js, Python;

– стабільність – технологія має пройти фазу «хайпу»;

– розвиток – технологія має бути «мертвою»;

– шифрування – добре, але не в першу чергу, вся робота йде у захищеному контурі;

– інтеграція – можливість вбудовування у існуючі системи, зокрема, авторизація та список користувачів;

– розширюваність – певний підхід до створення розширень у протоколі/ПЗ без «глобальних милиць».

Десь тут ми почали розуміти, що Телеграм, Дискорд та інший Слак нас не врятують. Потрібно переходити до області Open Source. Завдання ускладнювало те, що потрібно було інтегруватися з нашими існуючими системами та сервісами,

а також була потрібна передача різної специфіки з нагоди лікування (лікар, терміновість, тип діагнозу тощо). Можна було б придумати свій формат повідомлень – обмінюватися json-ами та передавати всю необхідну специфіку в полях, але хотілося не втрачати можливості роботи з якимись стандартними клієнтами без наших доробок для полегшення інтеграції сторонніх модулів.

Безумовно, одним із можливих шляхів було залишити все як є та продовжувати розвивати власне рішення. Тим більше, що на той момент вже окрім чатів була в якомусь вигляді реалізована підтримка аудіоконференцій. Альтернативним напрямом пошуку став перехід від готових рішень до області протоколів.

Побіжний пошук показав активний розвиток різних децентралізованих протоколів, наприклад, Matrix, Signal. Але за низкою параметрів, зокрема, можливість роботи з історією це нам не підходило. Щось стало відвертою екзотикою (OSCAR). Або більше належало до категорії «месенджер» ніж протокол (Mattermost). І тут хтось згадав про XMPP. Насправді у нас вже був досвід використання XMPP, але як корпоративний месенджер. Якраз у той період, коли ICQ вже перестала бути популярною, а щось просунуте ще не набуло потрібної популярності. Згодом, вже в «наш час» ми вдруге намагалися його використати, але незважаючи на цікаві фішки, які там з'явилися, наші технічні фахівці не змогли (або не захотіли) до ладу все налаштувати і XMPP програв гонку якомусь платному рішенню.

Короткий вступ у XMPP

XMPP

eXtensible Messaging and Presence Protocol – «розширюваний протокол обміну повідомленнями та інформацією про присутність», раніше відомий як джаббер. Відкритий, заснований на XML, вільний для використання протоколу для миттєвого обміну повідомленнями та інформацією про присутність у режимі, близькому до реального часу. Спочатку спроектований протокол, що легко розширюється, крім передачі текстових повідомлень, підтримує передачу голосу,

відео і файлів по мережі.

JID

Jabber Identifier будується за тим же принципом, що адреса електропошти: ім'я@домен. Може бути записаний у короткій формі ім'я@домен (bare JID) або у повній (full JID) ім'я@домен/ресурс. Ресурс служить для того, щоб можна було розрізнити кількох клієнтів, підключених до одного облікового запису. У кожного клієнта ресурс має бути унікальним. Тоді ми можемо вибирати надіслати повідомлення лише одному клієнту або всім одразу. JID може бути не тільки у користувача, а й у чат-кімнати, передплати і т.д. Для ресурсу вводиться поняття «пріоритету» – якщо повідомлення буде відправлено на короткий JID, то воно буде доставлене тому клієнту, пріоритет якого вищий (або всім, якщо пріоритет у всіх однаковий).

Станза (строфа)

Закінчений елемент XML-потоку, який містить певну керуючу інформацію:

– інформація про присутність (Presence) – інформаційні пакети спеціального виду, які містять інформацію про те, чи підключений в даний момент певний JID до мережі Jabber, а також передає його статус, статусне повідомлення та пріоритет;

– IQ (Info/Query) – особливий вид стансів, що реалізує механізм типу "запит-відповідь". Інтерпретація IQ-станс дозволяє «сутності» зробити запит та отримати відповідь від іншої «сутності». Тип даних, що передаються у запиті чи відповіді визначає простір імен (namespace) дочірнього елемента щодо IQ;

– Message – використовується для обміну повідомленнями між користувачами.

Ростер (список контактів)

Розбитий на групи список Jabber-адрес ваших співрозмовників (контактів). Зберігається на сервері та передається клієнту на запит. Сервер також обробляє запити на додавання, видалення контакту зі списку та зміни групи для

конкретного контакту.

XEP (розширення)

XMPP Extension Protocol – розширення протоколу XMPP. Наприклад, XEP-0045 – розрахований на багато користувачів чат, XEP-0084 – підтримка аватарок користувачів, XEP-0107 – статус користувача (user mood). XEP описують як якісь базові речі (XMPP Core), і безліч просунутого і дуже цікавого функціоналу.

Взагалі, розширення – одна з найцікавіших особливостей XMPP, коли, використовуючи цеглини, описані вище (різні станси), ми описуємо необхідний нам функціонал. За бажання можна зробити і власне розширення. На даний момент налічується близько двох сотень діючих XEP.

Використання XMPP

Для всіх цих випадків можна виділити одну картину (особливо характерну для великих порталів): швидкий старт сервісів, використовуючи XMPP, а потім, коли вступає в гру комерційна складова і завдання прив'язати користувача до порталу, вже народжуються якісь власні рішення, можливо, що залишаються у своїй масі, заснованими на XMPP.

Список компаній та рішень вселяв, завдання «заробляти з користувача» перед нами не стояло, і ми вже були готові бігти і робити все на XMPP. Але тут з'ясувалась одна особливість: для XMPP необхідно розглядати не тільки протокол, але переважно сервер і клієнта, що його реалізують. А все тому, що набір реалізованих розширень (тих XEP) від сервера до сервера можуть відрізнятися.

Вибираємо сервер

Найчастіше згадуваними серверами XMPP є (у дужках – мова реалізації):

– Ejabberd (Erlang).

– Prosody (Lua).

– Openfire (Java).

Коли ви читаєте про десятки та сотні тисяч користувачів, яких тримає

один XMPP-сервер, швидше за все йдеться про Ejabberd. Але ми одразу розуміли, що можливі доопрацювання, а фахівців з Erlang серед нас не було. Тому вибір ліг на Openfire від компанії Igniterealtime, до речі, автора одного з найпопулярніших XMPP-клієнтів для Android – Smack.

Деталі нашої реалізації

Для оптимізації роботи з нашим веб-додатком на PHP реалізували відправку повідомлень через плагін з REST API – інакше щоразу авторизуватися виходить накладно за часом та ресурсами. Додаткова фішка плагіна – підтримується надсилання повідомлень до json, включаючи наші додаткові поля:

Крім того, повідомлення в цьому плагіні складаються в чергу – додатковий плюс для масштабування.

Повідомлення ми зробили через кімнати (груповий чат) – бот відправляє повідомлення у потрібну кімнату і всі, хто до неї входить, отримують повідомлення. Зазвичай кімнати створюються за принципом "одна кімната – одне відділення". Це виявився найшвидший і найпростіший спосіб для реалізації.

Загальні враження щодо XMPP та Openfire

XML-природа протоколу нехай надмірна, але сувора і зручна.

XEP описують багато «смачних» речей, але треба уважно дивитися, що реалізовано для конкретних клієнтів та серверів. Список Openfire: http://download.igniterealtime.org/openfire/docs/latest/documentation/protocol-support.html. Для нас загалом цей список виявився достатнім.

Поняття «ресурсу» («пристрою») – може бути застосовано дуже широко, наприклад, у нас як «пристрій» може виступати бічна панель повідомлень для веб-програми, основне вікно чату в тому ж веб-додатку, мобільний пристрій, програма – "пейджер" і т.д.

До переваг Openfire можна віднести:

– активну розробку;

– багато готових плагінів;

– хороші можливості для кастомізації: за допомогою плагінів та

розширень можна налаштувати авторизацію, обробку пакетів, маршрутизацію та багато іншого.

З недоліків:

– не дуже успішно реалізований механізм плагінів, що реалізують своє REST API. Очевидно, автори спочатку не особливо розраховували на таке застосування, тому вийшло те, що вийшло;

– відсутнє автоматичне чищення історії в кімнатах – видаляємо скриптом із БД;

– при старті Openfire підвантажується ВСЯ історія по ВСІМ кімнатах – призводить до різкого зростання споживання пам'яті, вирішилося обмеженням глибини історії;

– за замовчуванням кімнати, що не використовуються, видаляються. Довго шукали в чому причина, поки не знайшли, що це регулюється опцією «Disable MUC room unloading for this service» у властивостях служби групового чату. Тут же можна налаштувати після кількох днів кімната, що не використовується, буде видалена, а також завантажувати чи ні всі кімнати при старті.

Окрім цього, для нас певною проблемою стало розгортання сервісу на регіонах – початкові варіанти та особливості інфраструктури вимагали застосування ручних налаштувань, і, на жаль, людський фактор дався взнаки. Після налаштування і контейнери допрацювали, стало набагато простіше.

Якщо потрібно швидко підняти корпоративний централізований месенджер або інтегрувати його до існуючого продукту – XMPP та Openfire чудовий варіант для старту. Чат, груповий чат – все працює "з коробки".

Потрібно уважно дивитися які XEP реалізує сервер і клієнт.

Загалом XMPP – це ближче до фреймворку, коли сервер (і клієнт) реалізують багато всяких цікавих штук, але потрібно докласти певного зусилля для того, щоб це стало закінченим рішенням.

Перспективи (плани на наступний етап):

– перехід на використання PubSub замість групового чату для повідомлень;

– «Пейджер» та push-повідомлення для лікарів – з відправкою знеособлених даних незахищеними мережами;

– авторизація через Держпослуги;

– інтеграція Openfire з Jitsi для аудіо-відео конференцій;

– інтеграція Openfire з Minio/IPFS для зберігання великих файлів, зокрема записів конференцій.

3.2 Розробка структурної схеми

Структурна схема розробленого програмного забезпечення системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями зображена на рисунку 3.1.

Розроблене програмне забезпечення складається з наступних блоків:

Клієнтська частина:

– Блок виводу GUI користувача програми.

– Блок авторизації.

– Блок обміну мовною інформацією.

– Блок обміну текстовими повідомленнями.

– Блок обміну відеоінформацією.

– Блок історії повідомлень по контактам.

– Блок пошуку.

– Блок відображення контактів.

– Блок налаштовувань.

– Блок встановлення статусів.

Серверна частина:

– База даних користувачів.

– База даних інформації про користувачів.

– База даних оффлайн повідомлень.

У основі програмного забезпечення лежить протокол OSCAR та Jabber.

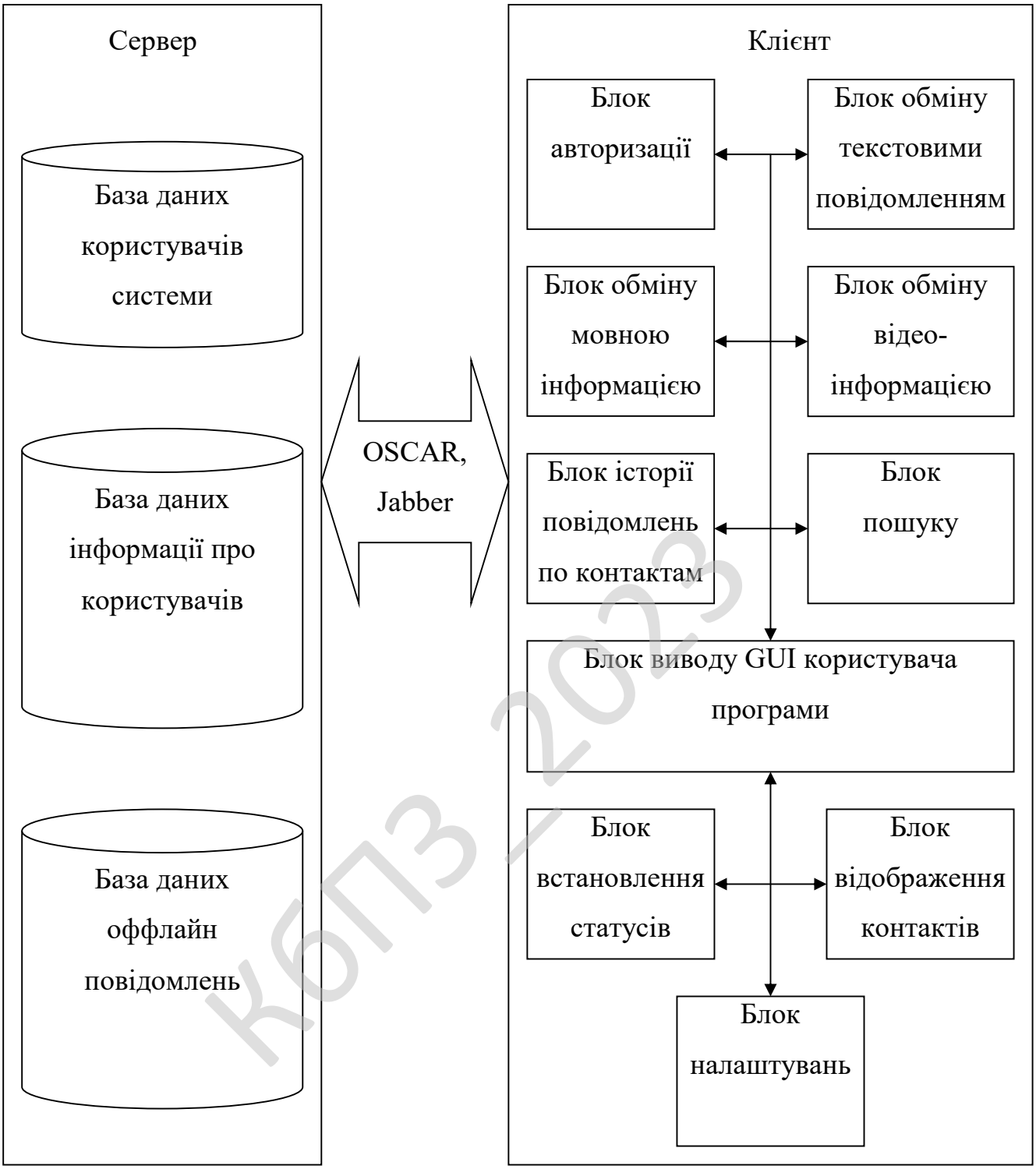

Рисунок 3.1 – Структурна схема системи

Згідно зі статистикою за 2023 рік, з появою месенджерів люди стали рідше дзвонити: 41% опитаних вважають за краще писати повідомлення, а не говорити по телефону. За даними OpenMarket, причина в тому, що під час дзвінка можна

упустити інформацію, а в чаті текст збережеться. Крім того, телефонні розмови забирають більше часу: на них необхідно концентруватися, а в листуванні можна відповідати паралельно займаючись іншими справами.

Месенджери змінили наше спілкування, це є факт. З одного боку ми завжди на зв'язку. Раніше ми могли забути про існування деяких знайомих та далеких родичів. Зараз WhatsApp і Viber регулярно нагадують про те, що та чи інша людина з'явилася в мережі.

Зворотний бік медалі у тому, що ми позбавляємо наші контакти природної енергії людського спілкування. Ми не докладаємо зусиль, щоб виявити любов та увагу до близьких. Завантажуючи картинку з поздоровленням, ми робимо це автоматично, без емоцій. Згадайте, як ви вітали вчителя з професійним святом, колегу – з Новим роком, троюрідного брата – з днем народження. Чи не траплялося так, що ви копіювали вітання з інтернету, або завантажували стандартну картинку з побажаннями?»

Viber вивчив психологічні моделі поведінки людей у месенджерах, опитавши 2,8 тис. респондентів. Переважна більшість (88%) відзначили, що спілкуватися в листуванні легше, ніж телефоном чи реальному житті, оскільки співрозмовники не бачать друга, над відповіддю можна подумати, а висловлювання емоцій є стікери і эмоди.

Також 45% опитаних вважають зручними спільні чати, стверджуючи, що так простіше залишатися на зв'язку з усіма близькими людьми, а 33% вважають за краще спілкуватися із співрозмовником у приватному діалозі.

Використання месенджерів – не добре і не погано. Вони заощаджують час, дозволяючи бути на зв'язку з тими, з ким ми раніше не могли розмовляти так часто. З іншого боку, вони позбавляють спілкування трепету, передчуття, інтимності. Під час самоізоляції месенджери дозволяли допомагати старшому поколінню, піклуватися, інформувати близьких про нововведення. Але ми ставали самотніми, годинами дивлячись у телефон.

З погляду психологічного здоров'я я наголошую на важливості живого спілкування, обіймів, прояву тілесності. Месенджери, швидше, повинні бути нам помічниками. Коли немає іншої можливості, ми робимо дзвінок у WhatsApp або Viber. Зідзвонюємося з тими, хто з тих чи інших причин далеко – колегами, закордонними родичами та друзями. Але якщо є можливість зустрітися, обійняти, торкнутися, поговорити до душі, краще зробити це».

Месенджери стали новим інструментом у бізнесі. За допомогою них зручно організовувати комунікацію з покупцями, які можуть поставити запитання про товар або послугу, та миттєво отримати відповідь від менеджера або чат-бота. Згідно з дослідженням, 62,5% компаній виставляють клієнтам рахунки через послуги обміну повідомленнями.

Месенджер-маркетинг допоможе бізнесу збільшити продажі. Наприклад, сервіс доставки їжі Elementaree з месенджерів отримує мільйонний виторг за пару годин.

Згідно зі статистикою завантаження програм для смартфонів за 2023 рік, месенджери – найпопулярніші програми. Щодня ними користуються мільйони, і в майбутньому це число зросте.

У бізнесі через месенджер покупець зможе купувати товари, послуги та керувати замовленнями – для цього не потрібно буде переходити на окремий сайт. Зараз ця функція є в китайському WeChat, через який можна забронювати авіаквиток, оформити доставку їжі та не тільки.

З режимом доповненої реальності можна буде приміряти одяг в онлайн-магазинах і одразу радитися з близькими з приводу покупки. Facebook Messenger тестував цей режим.

Режим доповненої реальності в месенджерах схожий на маски в Instagram Stories і Snapchat

Чат-боти стануть важливим інструментом у спілкуванні між бізнесом та клієнтом. Згідно з дослідженням Oracle, більше 50% покупців очікують, що компанії будуть доступні для здійснення угод по буднях і вихідним, у будь-який

час дня та ночі. Це можливо за допомогою ботів, які цілодобово відповідатимуть на запити клієнтів. Таке рішення допоможе заощадити до $174 млрд. у сферах страхування, фінансів, продажів.

Ще в 2019 році засновник Facebook Марк Цукерберг заявив, що скоро інтернет-спілкування буде переважно приватним. Прямий обмін повідомленнями між користувачами в месенджерах затьмарить публічні дискусії у соціальних мережах.

3.3 Розробка функціональної схеми

Нижче наведені функції додатка.

Обмін текстовими повідомленнями

За допомогою розробленого програмного забезпечення можна швидко, надійно й легко відправляти текстові повідомлення контактам. Розроблене програмне забезпечення також забезпечує підтримку офлайнових повідомлень. Якщо відправити текстове повідомлення користувачеві, що не перебуває в мережі, він одержить повідомлення, як тільки ввійде в розроблене програмне забезпечення.

Розроблене програмне забезпечення надає легкі у використанні, повнофункціональні, безпечні й надійні послуги миттєвого обміну повідомленнями. На перший погляд, це традиційна програма для миттєвого обміну повідомленнями, але відкривши список функцій, ви переконаєтеся, що це не те, із чим ви зіштовхувалися раніше. Ми надаємо професійний набір функцій і переваг, призначених головним чином для бізнесу й корпоративних клієнтів, але індивідуальні користувачі також знайдуть для себе багато корисного в розробленому програмному забезпеченні.

Функція обміну текстовими повідомленнями забезпечує безпечний і надійний зв'язок, до того ж вона дуже проста у використанні.

Нижче наведені деякі ради, щоб полегшити початок роботи із програмою:

– Початок сеансу чату. Виберіть ім'я в списку контактів. Клацніть значок "Текстовий чат" із правої сторони:  . Уведіть своє повідомлення в нижньому текстовому полі й натисніть кнопку "Відправити" (або клавішу Уведення).

– Більше легкий спосіб почати чат. Двічі клацніть ім'я в списку контактів. Уведіть своє повідомлення в нижньому текстовому полі й натисніть кнопку "Відправити" (або клавішу Уведення).

– Чат-конференція Можна запросити інших користувачів зі списку контактів у поточний сеанс чату й почати конференцію. Просто натисніть кнопку "Запросити користувачів" угорі, виберіть користувачів зі списку й клацніть "Запросити".

– Журнал повідомлень. Всі сеанси чату зберігаються локально на комп'ютері. За допомогою журналу чатів можна легко переглянути повідомлення, якими ви обмінювалися із друзями (колегами).

– Офлайнові повідомлення. Час від часу може виникати необхідність у відправленні миттєвих повідомлень користувачеві зі списку контактів, що у цей момент не перебуває в мережі. Розроблене програмне забезпечення гарантує, що ці повідомлення будуть доставлені відразу після того, як користувач знову ввійде в систему. Офлайнові повідомлення зберігаються до 10 днів.

– Вид тексту. Можна легко змінити колір шрифту, а також його стиль і оформлення тексту, після чого зберегти поточні параметри як параметри за замовчуванням, або можна використовувати поточний зовнішній вигляд тексту тільки для одного сеансу чату.

– Значки настрою (смайлики)  . Якщо необхідно поділитися своїми емоціями зі співрозмовником, можна використовувати значки настрою, передбачені в розробленому програмному забезпеченні. Виберіть відповідну емоцію в меню й вставте неї в текстове повідомлення.

– Виклик  . Використовуйте цю кнопку для залучення уваги. Якщо співрозмовник довгий час не відповідає, можна відправити сигнал для залучення його уваги.

– Відправити файл, відкрити спільний перегляд або білу дошку, почати відеочат. Оскільки вікно чату є однією з найбільш використовуваних функцій, у верхньому меню цього вікна відображаються інші функції. Завдяки цьому забезпечується швидкий доступ до кожної функції.

Обмін офлайновими текстовими повідомленнями

Розроблене програмне забезпечення забезпечує підтримку офлайнових повідомлень. Якщо відправити текстове повідомлення користувачеві, що не перебуває в цей момент у мережі, він одержить повідомлення, як тільки ввійде в розробленому програмному забезпеченні.

Якщо відправити текстові повідомлення користувачам, що не перебувають у цей момент у мережі, вони одержать повідомлення, як тільки ввійдуть у розробленому програмному забезпеченні.

Офлайнові повідомлення мають такий же рівень безпеки, як і звичайні онлайновы повідомлення. Вони шифруються, і тільки одержувач повідомлення може прочитати їх.

Журнал чатів

Всі текстові чати записуються локально на комп'ютер. Можна переглядати попередні сеанси чату в будь-який час.

Браузер історії чатів простий, легкий у використанні й інтуїтивно-зрозумілий. Можна легко переглядати попередні розмови або здійснювати пошук по ключових словах за допомогою засобу пошуку.

Відкриття журналу чатів. Браузер журналу чатів можна відкрити або у вікні чату, або в головному вікні програми для обміну повідомленнями. Під час використання вікна чату натисніть кнопку журналу чатів: . У головному вікні програми для обміну повідомленнями виберіть ім'я в списку контактів, а потім клацніть значок журналу чатів.

Пошук у базі даних журналу. Щоб переглянути стару розмову, його можна знайти по даті або ключовому слові за допомогою пошукової системи в нижній частині вікна

Голосовий чат

Розмова є більше легким способом спілкування. Ми розробили нову технологію, що забезпечує кращу якість без затримки сигналу під час розмови, незалежно від місця знаходження співрозмовника.

Спілкуватися усно значно легше, ніж відправляти миттєві повідомлення й повідомлення електронної пошти. Обмін текстовими повідомленнями має свої переваги, але в деяких випадках словесне спілкування більш ефективно. Здійснити стандартний виклик зовсім не складно – усі вміють користуватися телефоном. Із програмою процедура ідентична.

Що якщо клієнт, колега або співробітник перебувають в іншій країні або на іншому континенті? Тоді вам доведеться оплачувати величезні рахунки за дорогі міжнародні виклики.

У такому випадку найбільш ефективним рішенням є додаток голосового зв'язку за IP-протоколом.

Безпека й надійність. Розроблене програмне забезпечення забезпечує стійке шифрування кожного виклику, що робить розмову безпечним. Кожний виклик обробляється за допомогою розробленої нами спеціальної технології, що виключає ймовірність поганої якості зв'язку.

Безкоштовно. Так, це безкоштовно, ніяких додаткових схованих витрат. Незалежно від того, скільки триває розмова, вона завжди безкоштовна.

Легкість у використанні. Наша мета полягає в тому, щоб зробити програмне забезпечення зручним і легким у використанні. Ми прагнемо до того, щоб наші клієнти почували себе комфортно в процесі повсякденної роботи. Зробити голосовий виклик за допомогою програми дуже легко. Необхідно всього лише вибрати ім'я контакту, що перебуває в мережі, і клацнути значок "Голосова й відеозв'язок" із правої сторони.

Сумісність із брандмауером. Багато додатків голосового зв'язку за IP-протоколом не працюють із брандмауерами або трансляторами мережних адрес (NAT). Голосовий зв'язок за допомогою розробленого програмного забезпечення

за IP-протоколом сумісний із брандмауером. Це значить, що можна використовувати неї в обхід брандмауера або транслятора мережних адрес, аналогічно прямому підключенню до Інтернету.

Відеочат

За допомогою функції відеочату ми надаємо високоякісний і надійний відеозв'язок між користувачами. Сеанс відеочату можна проводити одночасно із сеансом текстового чату.

Наступною корисною функцією програми є відеочат. Крім того, що ви будете чути сміх, ви ще зможете бачити посмішки.

За допомогою функції відеочату можна організувати особисту зустріч. Для цього потрібно мати всього лише веб-камеру.

Можна відкрити кілька сеансів відеозв'язку одночасно з різними людьми. Єдиним обмеженням у цьому випадку є пропускна здатність підключення до Інтернету.

Почати відеочат так само легко, як відправити текстове повідомлення. Виберіть ім'я в списку контактів. Клацніть значок "Голосова й відеозв'язок" із правої сторони. Натисніть кнопку "Почати відео" під час голосового виклику.

Сеанс відеочату можна також почати у вікні текстового чату.

Передача файлів

Дозволяє передавати файл або групу файлів своїм контактам. Цей процес проходить через стійке шифрування, тому відсутні погрози для конфіденційності даних.

Швидка й надійна технологія передачі файлів між рівноправними вузлами!

Функція передачі файлів розробленому програмному забезпеченні автоматично стискає всі файли в процесі передачі. Це робить обмін файлами надзвичайно швидким і зменшує необхідну пропускну здатність.

Розроблене програмне забезпечення відкриває прямі канали між користувачами, що обмінюються файлами. До всіх переданих файлів застосовується стійке шифрування даних.

Це робить обмін файлами швидким і безпечним. Можливо бути впевнені в тому, що під час пересилання конфіденційні файли не будуть перехоплені й використані сторонніми особами.

Відправлення файлів. Розроблене програмне забезпечення надає кілька способів відправлення файлів, що робить програму надзвичайно зручною. Перетаскуйте обрані файли безпосередньо в список контактів у головному вікні розробленого програмного забезпечення. Перетаскуйте обрані файли безпосередньо у вікно текстового чату. Виберіть ім'я зі списку контактів і натисніть кнопку "Передача файлів" праворуч. Натисніть кнопку "Передача файлів" у вікні текстового чату під час сеансу чату з одним з контактів. Перетаскуйте обрані файли у вікно активного голосового або відеовиклику.

Після початку передачі файлу відкриється діалогове вікно відправлення файлу. У ньому можна побачити стан передачі, скасувати передачу або вибрати параметр "Закрити після завершення завантаження" (щоб автоматично закрити вікно).

Одержання файлів. Коли хтось відправляє файл, у розробленому програмному забезпеченні негайно відображається повідомлення. Одержання файлу також є дуже простою процедурою – додержуйтеся підказок.

Спільний перегляд

Ви й ваш співрозмовник можете переглядати веб-сайти разом, спільно використовуючи те саме зображення на екрані. Більше немає необхідності в обміні посиланнями або URL-адресами.

Покажіть іншому користувачеві те, що дивитеся самі!

Функція спільного перегляду є однією з багатьох корисних функцій розробленого програмного забезпечення.

Ви коли-небудь намагалися пояснити друзям, колегам або клієнтам, який веб-сайт переглядаєте або яке посилання клацнути, щоб вони могли бачити те, на що дивитеся ви?

За допомогою функції спільного перегляду (веб-браузер для спільної роботи) можливо зробити це значно легше.

Початок сеансу спільного перегляду. Виберіть ім'я в списку контактів. Клацніть значок "Спільний перегляд" із правої сторони:  . Уведіть URL-адресу в поле адреси. Після початку сеансу те ж вікно з'явиться на екрані співрозмовника, і ви зможете переглядати веб-сторінки разом.

Біла дошка

Виражайте свої ідеї легше ніж раніше за допомогою малювання. Розроблене програмне забезпечення оснащене широким набором різних засобів малювання, палітр кольорів, текстових заголовків і покажчиків, які допомагають візуалізувати ідеї.

Біла дошка – це відмінний засіб, що є частиною розробленого програмного забезпечення. Воно дозволяє обмінюватися малюнками й графічними даними в реальному часі у вікні додатка, схожого на MS Paint.

Користувачі, які спільно використовують функцію "Біла дошка", можуть перебувати на відстані в тисячу кілометрів друг від друга, але при цьому мати можливість разом малювати фігури, схеми й малюнки.

Функція "Біла дошка" призначена для того, щоб допомогти користувачеві виразити свою ідею за допомогою малювання. Вона надає можливість візуалізації й спільної роботи з іншим користувачем у реальному часі. Незалежно від того, чи розробляєте ви блок-схему для своєї організації або наполягаєте на важливості майбутніх цілей, біла дошка може бути неймовірно корисної.

При використанні в сполученні з іншими засобами біла дошка є дуже ефективним методом для проведення презентацій.

Початок сеансу "Біла дошка". Виберіть ім'я в списку контактів. Клацніть значок "Біла дошка" із правої сторони:  .

Нижче наведені доступні інструменти для малювання.

– Фарба.

– Малювання прямокутника.

– Малювання кола.

– Малювання лінії.

– Вставка малюнка.

– Заповнення області.

– Покажчик.

– Текстовий заголовок.

– Стирання.

Знімок екрана

Відправте знімок екрана свого комп'ютера іншому користувачеві всього одним клацанням.

Іноді в процесі роботи виникає необхідність у показі екрана деяким колегам. Відправлення знімка екрана є дуже зручним способом обміну інформацією. Це швидше, ніж набирати текст або давати посилання. Розроблене програмне забезпечення забезпечує найлегший спосіб одержання зображення екрана і його відправлення іншому користувачеві за кілька секунд.

Існують деякі застарілі й повільні способи, які усе ще використовуються. Наприклад, можна:

– Нажати клавішу "Print Screen" на клавіатурі.

– Вставити отримане зображення в програму Paint (або іншу програму для редагування зображень).

– Зберегти зображення.

– Знайти зображення на жорсткому диску.

– Відправити зображення другові або колезі (необхідно використовувати програму для обміну повідомленнями або електронною поштою).

Завдяки розробленому програмному забезпеченню можна відправити знімок екрана, виконавши всього два кроки:

– Виберіть ім'я контакту, що перебуває в мережі, або відкрийте вікно чату.

– Натисніть кнопку "Знімок екрана" у правій частині списку контактів  .

Якщо необхідно включити всі вікна у відправляється снимок, що, екрана, утримуйте натиснутої клавішу CTRL при натисканні кнопки "Знімок екрана".

Показ екрана

Працюйте віддалено на комп'ютері іншого користувача.

Ви коли-небудь намагалися пояснити клієнтові або колезі, як налаштувати комп'ютер або користуватися додатком? Іноді легше показати, чим пояснити. Іноді навіть краще зробити це самому на його або її комп'ютері за допомогою свого.

Функція показу екрана дозволяє без особливих складностей показати екран колезі або другові, а також попросити його або їй показати свій екран.

Існує чотири типи сеансів показу екрана за допомогою розробленого програмного забезпечення:

– відправлення екрана контакту;

– відправлення екрана контакту й надання йому дозволу використовувати вашу мишу й клавіатуру;

– перегляд екрана контакту;

– перегляд екрана контакту й використання його миші й клавіатури.

Такий інструмент необхідний у безлічі випадків. Можливо попросити IT-відділ допомогти з вашим комп'ютером. Співробітники IT-відділу можуть перебувати в тисячі кілометрів від вас, але вони мають можливість допомогти вам негайно. Можливо, вам знадобиться показати колезі нове програмне забезпечення, яким користуєтеся, разом переглядати веб-сайти та інше.

Нижче зазначені дві головних переваги функції показу екрана за допомогою розробленого програмного забезпечення.

Вона абсолютно безкоштовна – немає ніяких обмежень по частоті або тривалості використання. Її можна використовувати раз у день протягом 20 мінут або залишити включеної на весь день.

Вона надзвичайно легка у використанні – кілька клацань, і сеанс почнеться.

Для запуску сеансу показу екрана необхідно виконати всього три кроки.

– Виберіть ім'я в списку контактів.

– Клацніть значок "Обмін робочими столами" із правої сторони:  .

– Виберіть тип сеансу.

Автоматичне відновлення

Розроблене програмне забезпечення буде обновлятися автоматично при кожному випуску нової версії. Не потрібно відвідувати веб-сайт і перевіряти наявність нових версій.

Статус користувача

Можна вибрати один з декількох поточних статусів, щоб показати контактам, зайняті ви чи ні, або ж відсутні: "У мережі", "Немає на місці", "Зайнятий".

Розроблене програмне забезпечення автоматично перевіряє й установлює статус "Немає на місці", якщо на комп'ютері не виконуються ніякі дії протягом певного періоду часу. Таким чином, ваші колеги знають, що вас немає за комп'ютером у цей момент і ви не можете відповісти на їхні питання.

Установите статус "Зайнятий", щоб показати друзям і колегам, що прямо зараз ви зайняті й не хочете, щоб вас відволікали.

Можна також установити інші статуси, наприклад "Немає на місці: обідаю" або "Зайнятий: важлива нарада". Ці статуси, які налаштовуються відбивають особливі випадки, наприклад, коли ви перебуваєте на важливій нараді. Можливо призначити їхні функції "Автоматична відповідь" текстового чату, і програма автоматично відповість колезі й повідомить про те, що ви зайняті прямо зараз і відповісте пізніше.

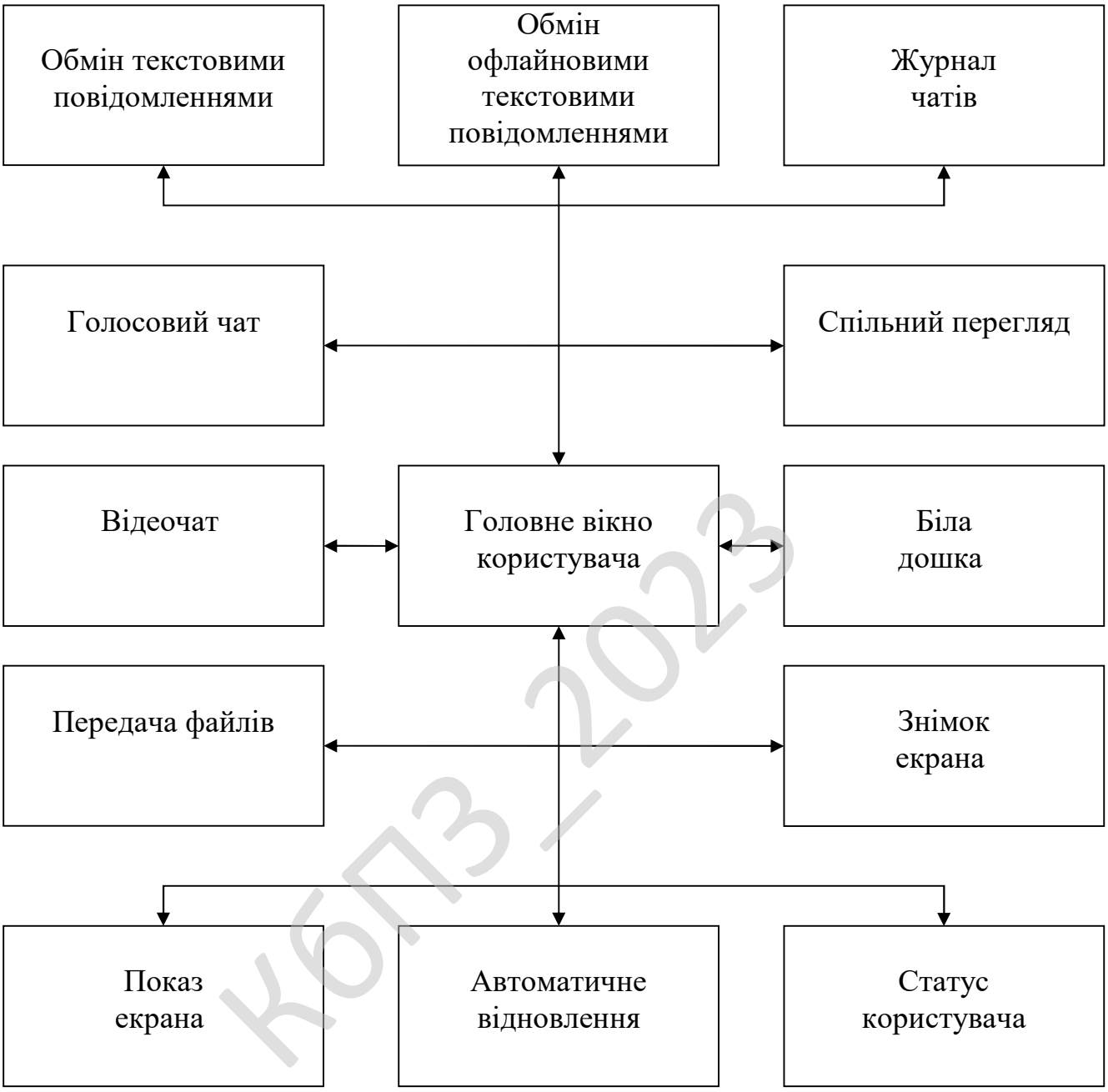

Рисунок 3.2 – Функціональна схема системи

Таким чином функціональна схема системи, яка зображена на рисунку 3.2, складається з наступних блоків:

– Головне вікно користувача.

– Обмін текстовими повідомленнями.

– Обмін офлайновими текстовими повідомленнями.

– Журнал чатів.

– Голосовий чат.

– Відеочат.

– Передача файлів.

– Спільний перегляд.

– Біла дошка.

– Знімок екрана.

– Показ екрана.

– Автоматичне відновлення.

– Статус користувача

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3.

Після початку роботи програми проходе виведення головного вікна ПЗ, далі може бути проведені дії налаштування ПЗ, через підключення бібліотеки додаткових функцій проведено редагування профілю, робота к контактами, перегляд логів, автоматична фільтрація. Крім цього через головне вікно проводиться підключення до серверу з подальшим обміном текстових та мультимедіа даних.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

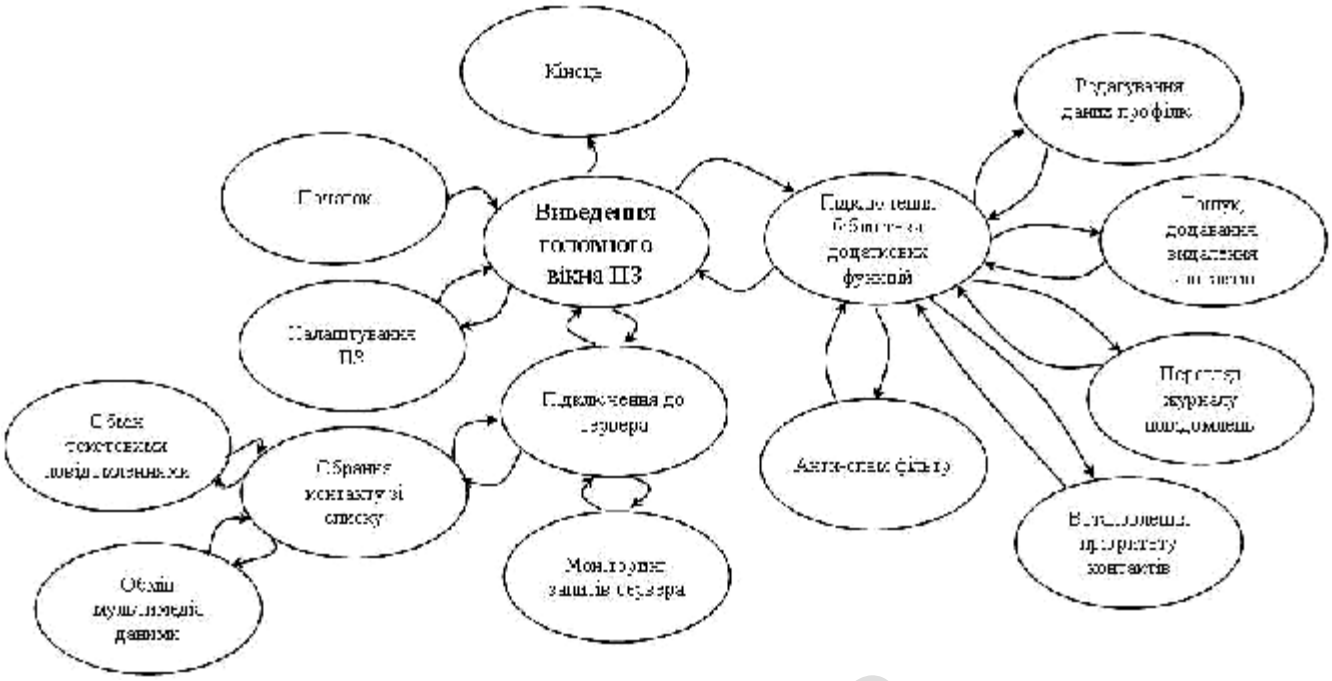

Рисунок 3.3 – Діаграма взаємодії процесів

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних основних кроків.

Спершу відбувається виведення вікна ідентифікації користувача. Якщо користувача не зареєстровано на сервері проводиться введення його даних та його послідуюча реєстрація на сервері.

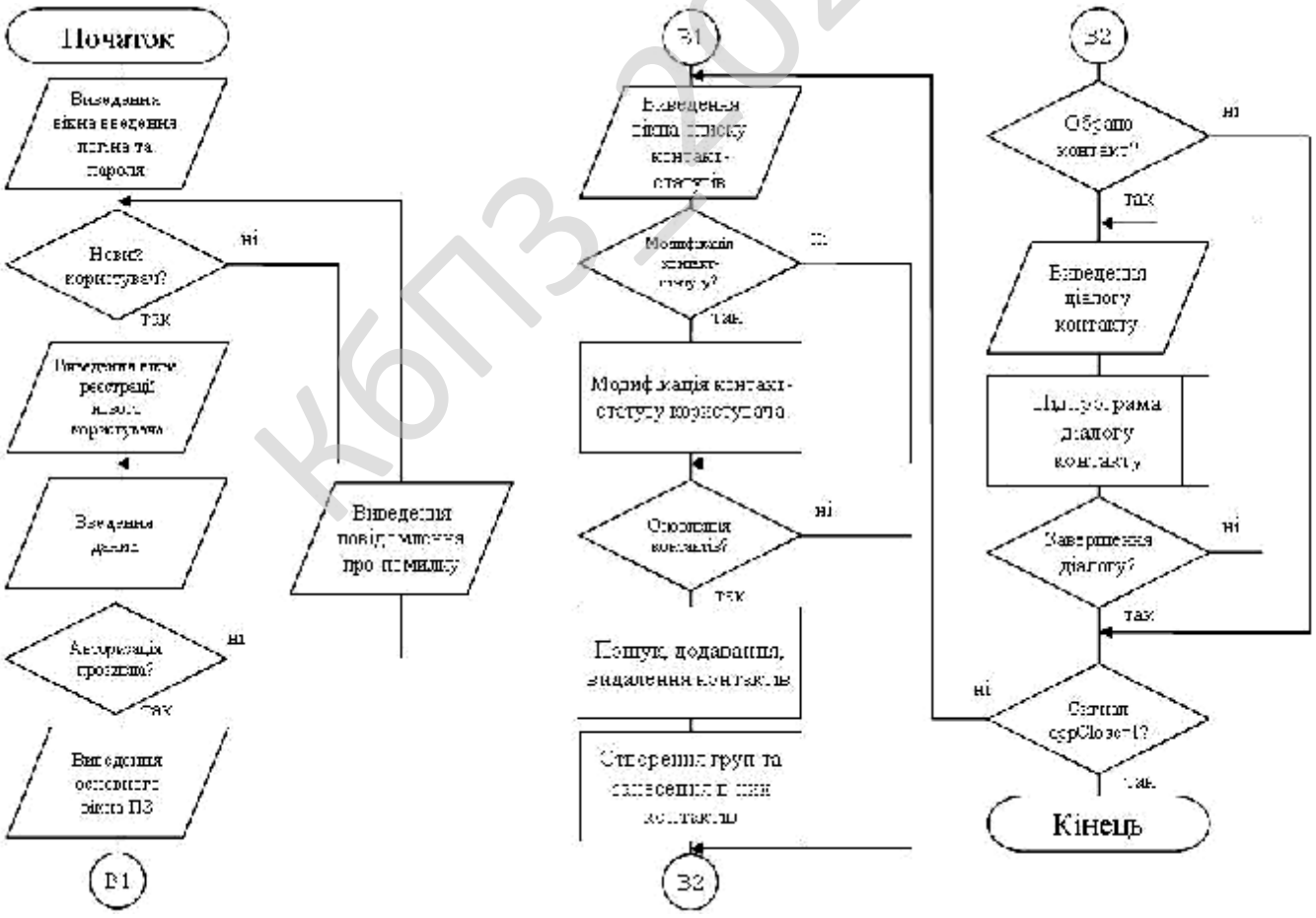

Рисунок 4.1 – Блок-схема основної програми

Після цього проходе виведення головного вікна програми де проводиться виведення списку контактів користувача.

Далі йде перевірка запита користувача на зміну контакт статусу, оновлення контактів та діалог з обраним контактом.

На рисунку 4.2 зображено підпрограму діалогу контакту, де спочатку проходе виведення поточного статусу діалогу, якщо отримано пакет повідомлення проводиться його обробка та виведення тексту на екран.

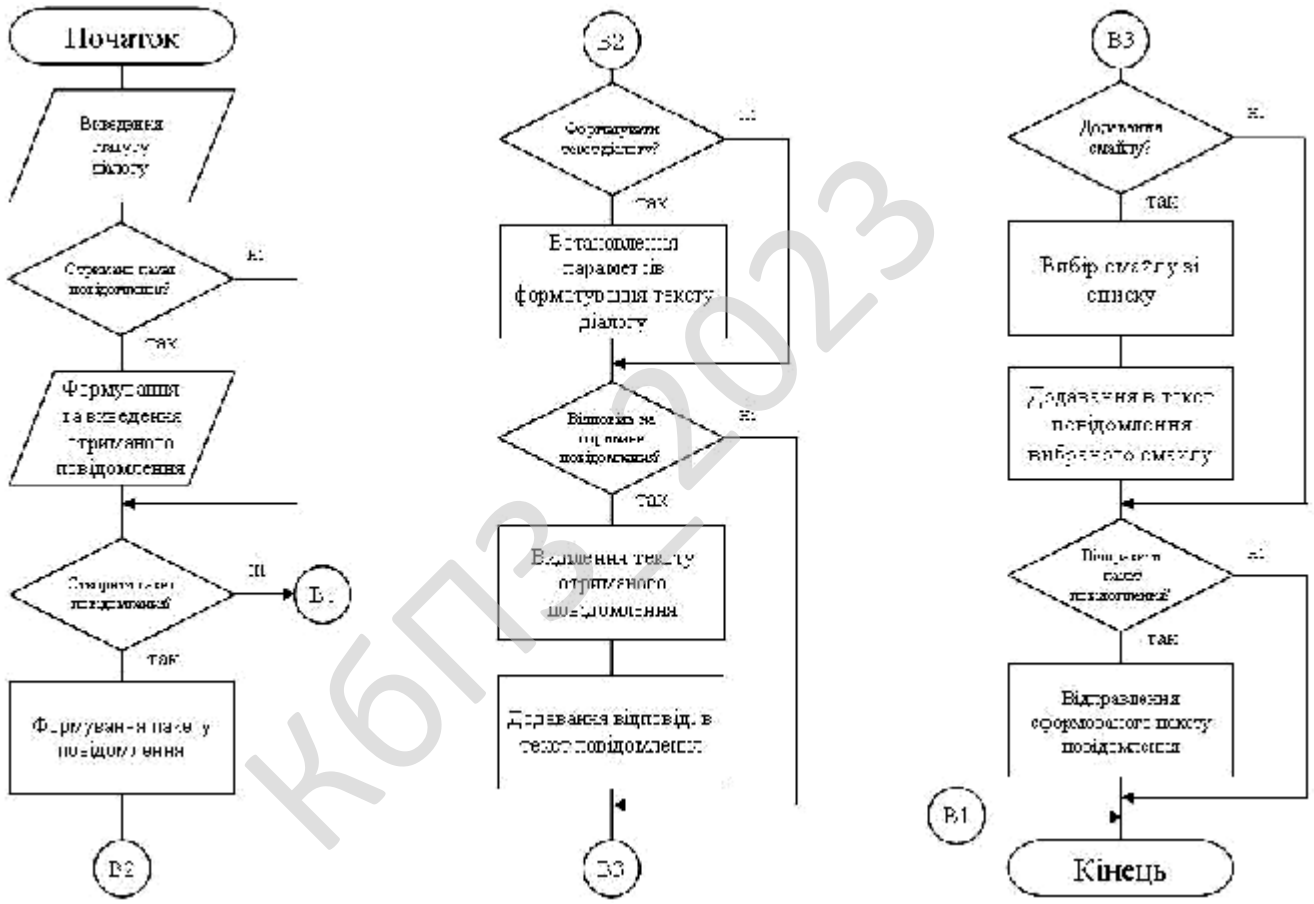

Рисунок 4.2 – Блок-схема підпрограми діалогу контакту

Якщо користувач хоче відправити повідомлення контакту проводиться формування пакету повідомлення з всебічними можливостями форматуванням повідомлення. Далі проходе відправлення повідомлення.

Реалізація алгоритму взаємодії з сервером

Сервер, заснований на сокетному протоколі, дозволяє обслуговувати відразу декількох клієнтів. Причому, обмеження на їхню кількість задає програма. Для кожного підключеного клієнта сервер відкриває окремий сокет, по якому проводиться обмін даними із клієнтом. Також створює для кожного підключення окремий процес (Thread).

Розберемо схему докладніше:

1. Визначення властивостей Port і Servertype – щоб до сервера могли нормально підключатися клієнти, потрібно, щоб порт, використовуваний сервером, точно збігався з портом, використовуваним клієнтом (і навпаки). Властивість Servertype визначає тип підключення;

2. Відкриття сокета – відкриття сокет й зазначеного порту. Тут виконується автоматичний початок очікування приєднання клієнтів (Listen);

3. Підключення клієнта й обмін даними з ним – тут підключається клієнт і проходить обмін даними з ним;

4. Відключення клієнта – тут клієнт відключається й закривається його сокетне з'єднання із сервером;

5. Закриття сервера й сокета – по команді адміністратора сервер завершує свою роботу, закриваючи всі відкриті сокетні канали й припиняючи очікування підключень клієнтів.

Пункти 3-4 повторюються багаторазово, тобто ці пункти виконуються для кожного нового підключення клієнта.

Опис використовуваного компонента TServerSocket

Розглянемо основні властивості, методи й події компонента Tserversocket які використовувалися при написанні програми.

Socket – клас TserverWinSocket, через який я одержував доступ до відкритих сокетних каналів.

Servertype – тип сервера. Може приймати одне із двох значень: stnonblocking – синхронна робота із клієнтськими сокетами.

При такому типі сервера можна працювати із клієнтами через події Onclientread і Onclientwrite. stthreadblocking – асинхронний тип. Для кожного клієнтського сокетного каналу створюється окремий процес (Thread). Тип: TServerType;

Threadcachesize – кількість клієнтських процесів (Thread), які будуть керуватися сервером. Тут необхідно підбирати середнє значення залежно від завантаженості сервера. Керування відбувається для того, щоб не створювати щораз окремий процес і не знищувати закритий сокет, а залишити їх для подальшого використання. Тип: Integer;

Active – показник того, активний у даних момент сервер, чи ні, фактично, значення True указує на те, що сервер працює й готів до приймання клієнтів, а False – сервер виключений. Щоб запустити сервер, потрібно просто привласнити цій властивості значення True. Тип: Boolean;

Port – номер порту для встановлення з'єднань із клієнтами. Порт у сервера й у клієнтів повинні бути однаковими. Застосовувалися значення від 1025 до 65535, тому що від 1 до 1024 – можуть бути зайняті системою. Тип: Integer;

Service – рядок, що визначає службу (ftp, http, pop, і т.д.), порт якої буде використаний. Це своєрідний довідник відповідності номерів портів різним стандартним протоколам. Тип: string.

Методи що використовувались:

– Open. Запускає сервер. По суті, ця команда ідентична присвоєнню значення True властивості Active;

– Close. Зупиняє сервер. По суті, ця команда ідентична присвоєнню значення False властивості Active.

Події що використовувались:

– OnClientConnect виникає, коли клієнт установив сокетне з'єднання й чекає відповіді сервера (Onaccept);

– OnClientDisconnect виникає, коли клієнт від'єднався від сокетного каналу;

– OnClientError виникає коли поточна операція завершилася невдало, тобто відбулася помилка;

– Onclientread виникає, коли клієнт передав серверу які-небудь дані. Доступ до цих даних можна одержати через переданий параметр Socket: TCustomWinSocket;

– Onclientwrite виникає, коли сервер може відправляти дані клієнтові по сокету;

– OnGetSocket в оброблювачі цієї події можна відредагувати параметр Clientsocket;

– OnGetThread в оброблювачі цієї події можна визначити унікальний процес (Thread) для кожного окремого клієнтського каналу, привласнивши параметру Socketthread потрібне підзавдання Tserverclientthread;

– Onthreadstart, Onthreadend виникає, коли підзавдання (процес, Thread) запускається або зупиняється, відповідно;

– Onaccept виникає, коли сервер ухвалює клієнта або відмовляє йому в з'єднанні;

– Onlisten виникає, коли сервер переходить у режим очікування приєднання клієнтів.

Tserversocket.Socket (Tserverwinsocket). Спілкуватися із клієнтом можна через параметр Clientsocket (Tcustomwinsocket):

– Activeconnections (Integer) кількість підключених клієнтів;

– Activethreads (Integer) кількість працюючих процесів;

– Connections (array) масив, що складається з окремих класів Tclientwinsocket для кожного підключеного клієнта. Наприклад, така команда: Serversocket1.Socket.Connections[0].Sendtext('Hello!'); Відсилає першому підключеному клієнтові повідомлення 'Hello!'. Команди для роботи з елементами цього масиву – також (Send/Receive) (Text,Buffer, Stream);

– Idlethreads (Integer) кількість вільних процесів. Такі процеси керуються сервером;

– Local address, Localhost, Localport відповідно – локальний Ip-адреса, хост, порт;

– Remot eaddress, Remote host, Remote port відповідно віддалений Ip-адрес, хост– ім'я, порт;

– Методи Lock і Unlock – відповідно, блокування й розблокування сокета.

Розглянемо приклад роботи з Tserversocket який реалізований у магістерській програмі, демонструється протоколювання всіх важливих подій і можливість приймати й відсилати текстові повідомлення.

Протоколювання й вивчення роботи сервера, посилка/приймання повідомлень через сокети:

```
{ заголовок файлу  Form1 }
procedure Tform1.Button1Click(Sender: Tobject);
begin
    {Визначаємо порт і запускаємо сервер}
      Serversocket1.Port := 1025;
    {Метод Insert вставляє рядок у масив у зазначену позицію}
      Memo2.Lines.Insert(0,'Завантаження серверу');
      Serversocket1.Open;
end;
procedure Tform1.Button2Click(Sender: Tobject);
begin
      {Зупиняємо сервер}
      Serversocket1.Active := False;
      Memo2.Lines.Insert(0,'Зупинка серверу');
end;
procedure Tform1.Serversocket1Listen(Sender: Tobject;
          Socket: Tcustomwinsocket);
begin
{Тут сервер "прослуховує" сокет на наявність клієнтів}
Memo2.Lines.Insert(0,'Listening on port '+Inttostr(
                    Serversocket1.Port));
end;
procedure Tform1.Serversocket1Accept(Sender: Tobject;
      Socket: Tcustomwinsocket);
begin
      {Тут сервер приймає клієнта}
      Memo2.Lines.Insert(0,'Client connection accepted');
end;
```

```
procedure Tform1.Serversocket1Clientconnect(Sender: Tobject;
      Socket: Tcustomwinsocket);
begin
      {Тут клієнт приєднується}
      Memo2.Lines.Insert(0,'Client connected');
end;
procedure Tform1.Serversocket1Clientdisconnect( Sender:
Tobject;
      Socket: Tcustomwinsocket);
    begin
      {Тут клієнт від'єднується}
      Memo2.Lines.Insert(0,'Client disconnected');
end;
procedure Tform1.Serversocket1Clienterror(Sender: Tobject;
      Socket: Tcustomwinsocket; Errorevent: Terrorevent;
      var Errorcode: Integer);
begin
{Відбулася помилка – виводимо її код}
Memo2.Lines.Insert(0,'Client error. Code = '+Inttostr(
Errorcode));
end;
procedure Tform1.Serversocket1Clientread(Sender: Tobject;
      Socket: Tcustomwinsocket);
begin
  {Від клієнта отримане повідомлення – виводимо його в Memo1}
      Memo2.Lines.Insert(0,'Message received from client');
      Memo1.Lines.Insert(0,'> '+Socket.Receivetext);
end;
procedure Tform1.Serversocket1Clientwrite(Sender: Tobject;
      Socket: Tcustomwinsocket);
begin
      {Тепер можна відіслати дані в сокет}
      Memo2.Lines.Insert(0,'Now can write to socket');
end;

procedure Tform1.Serversocket1Getsocket(Sender: Tobject;
Socket: Integer;
      var Clientsocket: Tserverclientwinsocket);
    begin
      Memo2.Lines.Insert(0,'Get socket');
    end;
procedure Tform1.Serversocket1Getthread(Sender: Tobject;
```

```
      Clientsocket: Tserverclientwinsocket;
      var Socketthread: Tserverclientthread);
begin
      Memo2.Lines.Insert(0,'Get Thread');
end;
procedure Tform1.Serversocket1Threadend(Sender: Tobject;
      Thread: Tserverclientthread);
begin
      Memo2.Lines.Insert(0,'Потік зупинено та знищено ');
end;
procedure Tform1.Serversocket1Threadstart(Sender: Tobject;
      Thread: Tserverclientthread);
begin
      Memo2.Lines.Insert(0,'Потік завантажено');
end;
procedure Tform1.Button3Click(Sender: Tobject);
     var i: Integer;
begin
      {Посилаємо всім клієнтам повідомлення з Edit1}
for i := 0 to Serversocket1.Socket.Activeconnections-1 do
begin
   Serversocket1.Socket.Connections[i].Sendtext(Edit1.Text);
end;
   Memo1.Lines.Insert(0,'< '+Edit1.Text);
end;
```

При роботі програми необхідно зберігати унікальні дані для кожного клієнта. Потрібно зберігати інформацію для кожного клієнта (ім'я, і ін.), причому із прив'язкою цієї інформації до сокету даного клієнта.

У деяких випадках робити все це вручну (прив'язка до handle сокета, масиви клієнтів, і т.д.) не дуже зручно.

Тому для кожного сокета існує спеціальна властивість – Data. Насправді, Data – це всього-на-всього покажчик.

Тому, записуючи дані клієнта в цю властивість, я додержувався правил роботи з покажчиками (виділення пам'яті, визначення типу, і т.д.).

Розглянемо реалізацію посилки файлів через сокет. Відкривається файл як файловий потік (Tfilestream) і відбувається відправлення його через сокет (Sendstream).

```
{Посилка файлу через сокет}
  procedure Sendfilebysocket(filename: string);
   var srcfile: Tfilestream;
  begin
    {Відкриваємо файл filename}
    srcfile := Tfilestream.Create(filename,fmopenread);
    {Посилаємо його першому підключеному клієнтові}
    Serversocket1.Socket.Connections[0].Sendstream(srcfile);
    {Закриваємо файл}
    srcfile.Free;
end;
```

Метод Sendstream використовується не тільки сервером, але й клієнтом (Clientsocket1.Socket.Sendstream(srcfile)), кілька блоків при передачі можуть поєднуватися в один.

Сокет це звичайний потік, він передає дані повільніше, саме тому дві команди:

```
Serversocket1.Socket.Connections[0].Sendtext('Hello,');
Serversocket1.Socket.Connections[0].Sendtext('world!');
```

Передача даних при використанні TCP

При програмуванні TCP використовуються ті ж функції, що й при програмуванні UDP, але їхнє поводження при цьому інше.

Для передачі даних за допомогою TCP необхідно спочатку встановити з'єднання, і після цього можливий обмін даними тільки з тією адресою, з яким це з'єднання встановлене.

Функція SendTo може використовуватися для TCP-сокетів, але її параметри, що задають адресу одержувача, ігноруються, а дані відправляються на ту адресу, з яким з'єднаний сокет.

Тому при відправленні даних через TCP звичайно використовують функцію Send, що дає той же результат. По цим причинам звичайно використовується Recv, а не RecvFrom.

В TCP існує поділ ролей взаємодіючих сторін на клієнт і сервер. Розлянео вивчення передачі даних в TCP з вивчення дій клієнта.

Для початку взаємодії клієнт повинен з'єднається із сервером за допомогою функції Connect.

У цьому випадку вона встановлює реальне з'єднання, тому її дії починаються з перевірки того, чи існує по зазначеній адресі серверний сокет, що перебуває в режимі очікування підключення.

Функція Connect завершується успішно тільки в тому випадку, якщо з'єднання встановлене, і серверна сторона виконала всі необхідні для цієї дії. При використанні Connect в TCP попередній явний виклик функції Bind також не обов'язковий.

На відміну від UDP, сокет в TCP не можна від’єднати або з'єднати з іншою адресою, якщо він уже з'єднаний. Для нового з'єднання необхідно використовувати новий сокет.

TCP є надійним протоколом, тобто в тому випадку, якщо пакет не доставлений, що відправляє сторона повідомляється про це. Проте, успішне завершення Send, як і у випадку UDP, не є гарантією того, що пакет був відісланий і дійшов до одержувача, а говорить тільки про те, що дані скопійовані у вихідний буфер сокету, і на момент копіювання сокет був з'єднаний.

Якщо надалі бібліотека сокетів не зможе відправити ці дані або не одержить підтвердження про їхню доставку, з'єднання буде закрито, і наступна операція із цим сокетом завершиться з помилкою.

Якщо вихідний буфер сокету дорівнює нулю, дані відразу копіюються в мережу, але успішне завершення функції й у цьому випадку не гарантує успішну доставку.

Використовувати нульовий вихідний буфер для TCP-сокетів не рекомендується, тому що це знижує продуктивність при послідовному відправленні даних невеликими порціями.

При буферизації ці порції накопичуються в буфері, а потім відправляються одним більшим пакетом, що вимагає одного підтвердження від клієнта.

Якщо ж буферизація не використовується, буде відправлено кілька дрібних пакетів, кожний зі своїм заголовком і своїм підтвердженням від клієнта, що приведе до зниження продуктивності.

Функція Recv копіює дані, що прийшли, із вхідного буфера сокету в буфер, заданий параметром Buf, але не більше BufLen байт. Скопійовані дані віддаляються з буфера сокету. При цьому всі отримані дані зливаються в один потік, тому одержувач може самостійно вибирати, який обсяг даних зчитувати за один раз.

Якщо за один раз була скопійована тільки частина пакета, що прийшов, що залишилася частина не пропадає, а буде скопійована при наступному виклику Recv. Функція Recv повертає кількість байт, скопійованих у буфер. Якщо на момент її виклику вхідний буфер сокету порожній, вона чекає, коли там щось з'явиться, потім копіює отримані дані й лише після цього повертає керування її програмі, що викликав.

Якщо Recv повертає 0, це значить, що віддалений сокет коректно завершив з'єднання. Якщо з'єднання завершене некоректно (наприклад, через обрив кабелю або збоїти віддаленого комп'ютера), функція завершується з помилкою (тобто повертає Socket_Error).

Тепер розглянемо, які дії повинен виконати сервер при використанні TCP. Сервер повинен перевести сокет у режим очікування з'єднання.

Це робиться за допомогою функції Listen, що має наступний прототип:

```
function Listen(S:TSocket;BackLog:Integer):Integer;
```

Параметр S задає сокет, що переводиться в режим очікування підключення. Цей сокет повинен бути прив'язаний до адреси, тобто функція Bind повинна бути викликана для нього явно.

Для сокету, що перебуває в режимі очікування, створюється черга підключень. Розмір цієї черги визначається параметром BackLog. Якщо цей параметр дорівнює SoMaxConn, черга буде мати максимально можливий розмір.

В MSDN'і відзначається, що довідатися максимально припустимий розмір черги стандартними засобами не можна. Функція повертає нуль у випадку успішного завершення й Socket_Error у випадку помилки.

Коли клієнт викликає функцію Connect, і по зазначеному в ній адресі є сокет, що перебуває в режимі очікування підключення, то інформація про клієнта міститься в чергу підключень цього сокету.

Успішне завершення Connect говорить про те, що на стороні сервера підключення додане в чергу. Однак для того, щоб з'єднання було дійсно встановлене, сервер повинен виконати ще деякі дії, а саме: витягти із черги з'єднань інформацію про з'єднання й створити сокет для його обслуговування. Ці дії виконуються за допомогою функції Accept, що має наступний прототип:

```
function Accept(S:TSocket;Addr:PSockAddr;AddrLen:PInteger):TSocket;
```

Параметр S задає сокет, що перебуває в режимі очікування з'єднання й із черги якого витягає інформація про з'єднання. Вихідний параметр Addr дозволяє одержати адресу клієнта, що встановив з'єднання.

Тут повинен бути переданий покажчик на буфер, у який ця адреса буде поміщений. Параметр AddrLen містить покажчик на змінну, у якій зберігається довжина цього буфера: до виклику функції ця змінна повинна містити фактичну довжину буфера, що задається параметром Addr, після виклику – кількість байт буфера, що реально знадобилися для зберігання адреси клієнта. Очевидно, що при використанні TCP і вхідне, і вихідне значення цієї змінної повинне бути дорівнює SizeOf(TSockAddr).

Ці параметри передаються як покажчики, а не як параметри-змінні, що було б більш природно для Delphi, тому що бібліотека сокетів допускає для цих покажчиків нульові значення, якщо сервер не цікавить адреса клієнта. У цьому випадку розроблювачі модуля WinSock зберегли повну функціональність, надавану даною бібліотекою.

У випадку помилки функція Accept повертає значення Invalid_Socket. У випадку успішного завершення вертається дескриптор сокету, створеного бібліотекою сокетів і призначеного для обслуговування даного з'єднання. Цей

сокет уже прив'язаний до адреси й з'єднаний із сокетом клієнта, що встановив з'єднання, і його можна використовувати у функціях Recv і Send без попереднього виклику яких-небудь інших функцій. Знищується цей сокет звичайним образом, за допомогою CloseSocket.

Вихідний сокет, обумовлений параметром S, залишається в режимі прослуховування. Якщо сервер підтримує одночасне з'єднання з декількома клієнтами, функція Accept може бути викликана багаторазово. Щораз при цьому буде створюватися новий сокет, що обслуговує одне конкретне з'єднання: протокол TCP і бібліотека сокетів гарантують, що дані, послані клієнтами, потраплять у буфери відповідних сокетів і не будуть перемішані.

Для одержання цілісної картини коротко повторимо вищесказане. Для встановлення з'єднання сервер повинен, по-перше, створити сокет за допомогою функції Socket, а по-друге, прив'язати його до адреси за допомогою функції Bind. Далі сокет повинен бути переведений у режим очікування за допомогою функції Listen, а потім за допомогою функції Accept створюється новий сокет, що обслуговує з'єднання, установлене клієнтом. Після цього сервер може обмінюватися даними із клієнтом.

Клієнт же повинен створити сокет, при необхідності прив'язки до конкретного порту викликати Bind, і потім викликати Connect для встановлення з'єднання. Після успішного завершення цієї функції клієнт може обмінюватися даними із сервером. Це ілюструється наведеними нижче прикладами.

Код сервера:

```
var S,AcceptedSock:TSocket;
    Addr:TSockAddr;
    Data:TWSAData;
    Len:Integer;
 begin
  WSAStartup($101,Data);
  S:=Socket(AF_Inet,Sock_Stream,0);
  Addr.sin_family:=PF_Inet;
  Addr.sin_port:=HToNS(3030);
  Addr.sin_addr.S_addr:=InAddr_Any;
  FillChar(Addr.Sin_Zero,SizeOf(Addr.Sin_Zero),0);
```

```
  Bind(S,Addr,SizeOf(TSockAddr));
  Listen(S,SoMaxConn);
  Len:=SizeOf(TSockAddr);
  AcceptedSock:=Accept(S,@Addr,@Len);
  { Тепер Addr містить адреса клієнта, з яким установлено
    з'єднання, а AcceptedSock – дескриптор, що обслуговує це
    з'єднання. Припустимо наступні дії:
     Send(AcceptedSock,…)–відправити дані клієнтові
     Recv(AcceptedSock,…)–одержати дані від клієнта
     Accept(…)–установити з'єднання з новим клієнтом }
```

Тут сокет сервера прив'язується до порту з номером 3030. У загальному випадку розроблювач сервера сам повинен вибрати порт із діапазону 1024-65535.

Код клієнта:

```
var S:TSocket;
    Addr:TSockAddr;
    Data:TWSAData;
 begin
  WSAStartup($101,Data);
  S:=Socket(AF_Inet,Sock_Stream,0);
  Addr.sin_family:=AF_Inet;
  Addr.sin_port:=HToNS(3030);
  Addr.sin_addr.S_addr:=Inet_Addr(...);
  FillChar(Addr.Sin_Zero,SizeOf(Addr.Sin_Zero),0);
  Connect(S,Addr,SizeOf(TSockAddr));
  { Тепер з'єднання встановлене. Припустимо наступні дії:
     Send(S,…)–відправити дані серверу
     Recv(S,…)–одержати дані від сервера }
```


У наведеному вище коді для стислості опущені перевірки результатів функцій з метою виявлення помилок. При написанні серйозних програм цим зневажати не можна. Якщо на момент виклику функції Accept черга з'єднань порожня, то нитка, що викликала її, блокується доти, поки який-небудь клієнт не підключиться до сервера.

З одного боку, це зручно: сервер може не викликати функцію Accept у циклі доти, поки вона не завершиться успіхом, а викликати її один раз і чекати, коли підключиться клієнт. З іншого боку, це створює проблеми тим серверам, які повинні взаємодіяти з декількома клієнтами. Дійсно, нехай функція Accept успішно завершилася й у розпорядженні програми виявилися два сокети: що

перебуває в режимі очікування нових підключень і створений для обслуговування вже існуючого підключення.

Якщо викликати Accept, то програма не зможе продовжувати роботу доти, поки не підключиться ще один клієнт, а це може відбутися через дуже тривалий проміжок часу або взагалі ніколи не відбутися. Через цього програма не зможе обробляти виклики клієнта, що вже підключився. З іншого боку, якщо функцію Accept не викликати, сервер не зможе виявити підключення нових клієнтів. Для рішення цієї проблеми бібліотека сокетів пропонує засобу, які ми розглянемо нижче (а бібліотека Windows Sockets пропонує для цього свої засоби, які будуть розглянуті в наступній статті).

Тут же я хочу запропонувати досить популярний спосіб її рішення, що використовує засобу не бібліотеки сокетів, а операційної системи. Він полягає у використанні окремої нитки для обслуговування кожного із клієнтів. Щораз, коли клієнт підключається, функція Accept передає керування програмі, повертаючи новий сокет.

Тут сервер може породити нову нитку, що призначена винятково для обміну даними з новим клієнтом.

Стара нитка після цього знову викликає Accept для старого сокету, а нова – функції Recv і Send для нового сокету. Такий метод вирішує заодно й проблеми, пов'язані з тим, що функції Send і Recv також можуть блокувати роботу програми й перешкодити обміну даними з іншими клієнтами.

У цьому випадку буде блокована тільки одна нитка, що обмінюється даними з одним із клієнтів, а інші нитки продовжать свою роботу.

Дані передаються з використанням сокетів. Сокетом називається спеціальний об'єкт, створюваний для відправлення й одержання даних через мережу. Відзначимо, що під терміном "об'єкт" у цьому випадку мається на увазі не об'єкт у термінах об'єктно-орієнтованого програмування, а деяка сутність, внутрішня структура якої схована від нас, тому із цією сутністю ми можемо оперувати тільки як з єдиним і неподільним (атомарним) об'єктом.

Цей об'єкт створюється усередині бібліотеки сокетів, а програміст, що використовує цю бібліотеку, одержує унікальний номер (дескриптор) цього сокету. Конкретне значення цього дескриптора не несе для програміста ніякої корисної інформації й може бути використано тільки для того, щоб при виклику функції з бібліотеки сокетів указати, з яким сокетом потрібно виконати операцію.

Щоб дві програми могли спілкуватися один з одним через мережу, кожна з них повинна створити сокет. Кожний сокет володіє двома основними характеристиками: протоколом і адресою, до яких він прив'язаний. Протокол задається при створенні сокету й не може бути змінений згодом. Адреса сокету задається пізніше, але обов'язково до того, як через сокет підуть дані. У деяких випадках прив'язка сокету до адреси може бути неявною.

Формат адреси сокету визначається конкретним протоколом. Зокрема, для протоколів TCP і UDP адреса складається з IP-адреси мережного інтерфейсу й номера порту.

Кожний сокет має два буфери: для вхідних і для вихідних даних. При відправленні даних вони спочатку кладуть у буфер вихідних, і лише потім відправляються у фоновому режимі.

Програма в цей час продовжує свою роботу. При одержанні даних сокет кладе їх у буфер для вхідних, звідки вони потім можуть витягати програмою.

Мережа може зв'язувати різні апаратні платформи, тому потрібне узгодження форматів переданих даних, зокрема – форматів цілих чисел. Двобайтніе цілі числа зберігаються в пам'яті у двох послідовно розташованих байтах.

При цьому можливі два варіанти: у першому байті зберігається молодший байт числа, а в другому – старший, і навпаки. Спосіб зберігання визначається апаратною частиною платформи. Процесори Intel використовують перший варіант, тобто першим зберігається молодший байт, а інші процесори (наприклад, Motorola) – другий варіант. Те ж стосується й чотирьохбайтних чисел: процесори Intel зберігають їх, починаючи з молодшого байта, а деякі інші процесори –

починаючи зі старшого. Мережний формат подання таких чисел збігається з форматом процесора Motorola, тобто на платформах із процесором Intel необхідно переставляти байти при конвертації чисел у мережний формат.

Протокол UDP

Функції, які розглядаються в цьому розділі, можуть бути використані й з іншими протоколами, і від цього їхнє поводження може мінятися. Ми тут описуємо тільки їхнє поводження при використанні UDP.

Для передачі даних віддаленому сокету використовується функція SendTo, описана в такий спосіб:

```
function SendTo(S:TSocket;var Buf;Len,Flags:Integer;
                var AddrTo:TSockAddr;ToLen:Integer):Integer;
```

Перший параметр даної функції задає сокет, що використовується для передачі даних. Тут потрібно вказати значення, отримане раніше від функції Socket.

Параметр Buf задає буфер, у якому зберігаються дані для відправлення, а параметр len – розмір цих даних у байтах. Параметр Flags дозволяє вказати деякі додаткові опції, яких ми тут стосуватися не будемо, тому що в більшості випадків вони не потрібні.

Поки варто запам'ятати, що параметр Flags у функції SendTo, а також в інших функціях, де він зустрічається, повинен бути дорівнює нулю. Параметр AddrTo задає адресау(що складається з IP-адреси й порту) віддаленого сокету, що повинен одержати ці дані. Значення параметра AddrTo повинне формуватися по тим же правилам, що й значення аналогічного параметра функції Bind, за винятком того, що IP-адреса й порт повинні бути задані явно (тобто не допускається використання значення InAddr_Any і нульового номера порту).

Параметр ToLen задає довжину буфера, відведеного для адреси, і повинен бути дорівнює SizeOf(TSockAddr).

Один виклик функції SendTo приводить до відправлення однієї дейтаграми. Дані, передані в SendTo, ніколи не розбиваються на трохи дейтаграм,

і дані, передані послідовними викликами SendTo, ніколи не поєднуються в одну дейтаграму.

Функцію SendTo можна використовувати із сокетами, не прив'язаними до адреси. У цьому випадку усередині бібліотеки сокетів буде неявно викликана функція Bind для прив'язки сокету до адреси InAddr_Any і нульовому порту (тобто адреса й порт будуть обрані системою).

Якщо вихідний буфер сокету має ненульовий розмір, SendTo кладе дані в цей буфер і відразу повертає керування програмі, а властиво відправлення даних здійснюється бібліотекою сокетів у фоновому режимі.

Тому успішне завершення SendTo гарантує тільки те, що дані скопійовані в буфер і що на момент їхнього копіювання не виявлено ніяких проблем, які робили б неможливим їхнє відправлення. Але такі проблеми можуть виникнути пізніше, тому навіть у випадку успішного завершення SendTo відправник не одержує гарантії, що дані послані.

Якщо у вихідному буфері сокету не вистачає місця для нової порції даних, SendTo не повертає керування програмі доти, поки в буфері за рахунок фонового відправлення не з'являться досить місця або не буде виявлена помилка.

Якщо розмір вихідного буфера сокету дорівнює нулю, функція SendTo копіює дані відразу в мережу, без проміжної буферизації.

Коли функція поверне керування програмі, програма може бути впевнена, що інформація вже успішно передана в мережу. Однак навіть у цьому випадку успішне завершення SendTo не гарантує того, що інформація доставлена: дейтаграма може втратитися по дорозі.

У випадку успішного завершення функція SendTo повертає кількість байт, скопійованих у буфер (або переданих прямо в мережу, якщо буфера немає). Для протоколу UDP це значення може бути дорівнює тільки значенню параметра Len, хоча для деяких інших протоколів (наприклад, TCP), можливі ситуації, коли в буфер сокету копіюється тільки частина даних, переданих програмою, і тоді SendTo повертає значення в діапазоні від 1 до Len. Якщо при виконанні SendTo

виникає помилка, вона повертає значення Socket_Error (ця константа має негативне значення).

Для одержання даних, присланих сокету, використовується функція RecvFrom, що має наступний прототип:

```
function RecvFrom(S:TSocket;var Buf;Len,Flags:Integer;
                  var From:TSockAddr;var FromLen:Integer):Integer;
```

Параметр S задає сокет, із вхідного буфера якого будуть витягати дані, параметр Buf – буфер, у який ці дані будуть копіюватися, а параметр Len – розмір цього буфера. Параметр Flags задає додаткові опції й у більшості випадків повинен бути дорівнює нулю. Параметр From є вихідним параметром: у нього міститься адреса, з якого була послана дейтаграма. Параметр FromLen задає розмір у байтах буфера для адреси відправника.

При виклику функції значення змінної, що підставляється як фактичний параметр, повинне бути дорівнює SizeOf(TSockAddr). Функція міняє це значення на ту довжину, що реально треба було для зберігання адреси відправника (у випадку UDP це значення також буде дорівнює SizeOf(TSockAddr)).

В оригіналі параметри From і FromLen передаються як покажчики, і програма може використовувати замість них нульові покажчики, якщо її не цікавить адреса відправника.

Розроблювачі модуля WinSock замінили покажчики параметрами-змінними, що в більшості випадків зручніше. Однак можливість передавати нульові покажчики при цьому виявилася загубленою.

Функція RecvFrom завжди читає тільки одну дейтаграму, навіть якщо розмір переданого їй буфера достатній для читання декількох дейтаграм. Якщо на момент виклику RecvFrom дейтаграми у вхідному буфері сокету відсутні, функція буде чекати, поки вони там з'являться, і до цього моменту не поверне керування її програмі, що викликав. Якщо в буфері перебуває декілька дейтаграм, то вони читаються в порядку черговості надходження в буфер.

Дейтаграми можуть надходити в буфер не в тім порядку, у якому вони були відправлені. Крім того, буфер може містити значення, що повертається

функцією RecvFrom, дорівнює довжині прочитаної дейтаграми. Це значення може бути дорівнює нулю, тому що UDP дозволяє відправляти дейтаграми нульової довжини (для цього при виклику SendTo треба задати параметр Len рівним нулю). Якщо виявлено якусь помилку, вертається значення Socket_Error.

Якщо розмір буфера, обумовленого параметром Buf, менше, ніж перша лежача у вхідному буфері сокету дейтаграма, то копіюється тільки частина дейтаграми, що міститься в буфері, а RecvFrom завершується з помилкою (WSAGetLastError при цьому поверне помилку WSAEMsgSize). Частина дейтаграми, що залишилася, при цьому безповоротно губиться, при наступному виклику RecvFrom буде прочитана наступна дейтаграма.

Цієї проблеми легко уникнути, тому що довжина дейтаграми в UDP не може перевищувати 65507 байт. Досить підготувати буфер відповідної довжини, і в нього гарантовано поміститься будь-яка дейтаграма.

Інший спосіб уникнути подібної проблеми – використовувати прапор Msg_Peek. У цьому випадку дейтаграма не віддаляється із вхідного буфера сокету, а значення, що повертається функцією RecvFrom, дорівнює довжині дейтаграми. При цьому в буфер, заданий параметром Buf, копіюється та частина дейтаграми, що у ньому міститься. Програма може діяти в такий спосіб: викликати RecvFrom із прапором Msg_Peek, виділити пам'ять, необхідну для зберігання дейтаграми, викликати RecvFrom без прапора Msg_Peek, щоб видалити прочитати дейтаграму цілком і видалити її із вхідного буфера сокету. Цей метод складніше, а 65507 байт – не дуже більша по нинішніх мірках пам'ять, тому легше все-таки використовувати буфер фіксованої довжини.

Функцію RecvFrom не можна використовувати з тими сокетами, які ще не прив'язані до адреси, тому перед викликом цієї функції повинна бути викликана або функція Bind, або функція, що здійснює неявну прив'язку сокету до адреси (наприклад, SendTo).

Протокол UDP не підтримує з'єднання в тому розумінні, у якому їх підтримує TCP, але бібліотека сокетів дозволяє частково імітувати таке з'єднання. Для цього служить функція Connect, що має наступний прототип:

```
function Connect(S:TSocket;var Name:TSockAddr; NameLen:Integer):Integer;
```

Параметр S задає сокет, що повинен бути "з'єднаний" з віддаленою адресою. Адреса задається параметром Name аналогічно тому, як він задається в параметрі Addr функції SendTo. Параметр NameLen містить довжину структури, що описує адреса, і повинен бути дорівнює SizeOf(NameLen). Функція повертає нуль у випадку успішного завершення й Socket_Error у випадку помилки.

Виклик функції Connect у випадку використання UDP установлює фільтр для вхідних дейтаграм. Дейтаграми, адреса відправника яких не збігається з адресою, заданою у функції Connect, ігноруються: нові дейтаграми не містяться у вхідний буфер сокету, а ті, які лежали там на момент виклику Connect, віддаляються з нього. Connect не перевіряє, чи існує адреса, з яким сокет "з'єднується", і може успішно завершитися, навіть якщо вузла з таким IP-адресою не існує.

Програма може викликати Connect необмежене число раз із різними адресами. Якщо параметр Name задає IP-адресу InAddr_Any і нульовий порт, то сокет "від'єднується", тобто всі фільтри для нього знімаються, і він поводиться так само, як сокет, для якого не була викликана функція Connect. Для сокетів, не прив'язаних до адреси, Connect неявно викликає Bind.

Після виклику Connect для відправлення даних можна використовувати функцію Send з наступним прототипом:

```
function Send(S:TSocket;var Buf;Len,Flags:Integer):Integer;
```

Від функції SendTo вона відрізняється відсутністю параметрів AddrTo і ToLen. При використанні Send дейтаграма відправляється за адресою, заданому при виклику Connect. В іншому цій функції поводяться однаково.

Функція SendTo при використанні з "з'єднаним" сокетом поводиться так само, як з несполученим, тобто відправляє дейтаграму за адресою, обумовленому параметром AddrLen, а не за адресою, заданому при виклику Connect.

Для одержання даних через "з'єднані" сокети можна використовувати функцію Recv, що має наступний прототип:

```
function Recv(S:TSocket;var Buf;Len,Flags:Integer):Integer;
```

Від свого аналога RecvFrom вона відрізняється тільки відсутністю параметрів From і FromLen, через які передається адреса відправника дейтаграми. Строго говорячи, функцію Recv можна використовувати й для несполучених сокетів, але при цьому програмі залишається невідомим адреса відправника.

У випадку ж "з'єднаних" сокетів адреса відправника заздалегідь відомий – це адреса, задана у функції Connect, а дейтаграми всіх інших відправників будуть відкидатися. Функцію RecvFrom також можна використовувати для "з'єднаних" сокетів, але адреса отруйника, що вона повертає, у цьому випадку може бути тільки той, котрий визначений у функції Connect.

Таким чином, функція Connect при використанні протоколу UDP дозволяє, по-перше, виконати фільтрацію вхідних дейтаграм за адресою засобами самої бібліотеки сокетів, а по-друге, використовувати більше лаконічні альтернативи RecvFrom і SendTo – Recv і Send.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм PRESENT – окремий випадок SP-мережі й складається з 31 раунду. Довжина блоку становить 64 біта, а ключі підтримуються в 2 варіантах, 80- і 128-бітні. Такого рівня захисту повинно цілком вистачати для низькозахищених додатків, звичайно використовуваних для розгортання на основі тегів, а крім того, що важливіше, PRESENT багато в чому збігається своїми конструктивними особливостями з потоковими шифрами проекту estream, заточеними на ефективну реалізацію в залозі, що дозволяє нам адекватно порівнювати їх.

Вимоги з безпеки й експлуатаційні властивості 128-бітних версій надані в додатку до оригінальної статті.

Кожний з 31 раундів складається з операції XOR, щоб увести ключ K_i для $1 \leq i \leq 32$, де K_{32} використовується для «відбілювання» ключа, лінійної побітової перестановки й нелінійного шару заміщення (або, попросту говорячи, збільшення стійкості шифрування . Нелінійний шар використовує роздільні 4-бітні S-блоки, які застосовуються паралельно 16 раз на кожному раунді. Шифр, описаний псевдо-кодом представлено на рисунку 4.3.

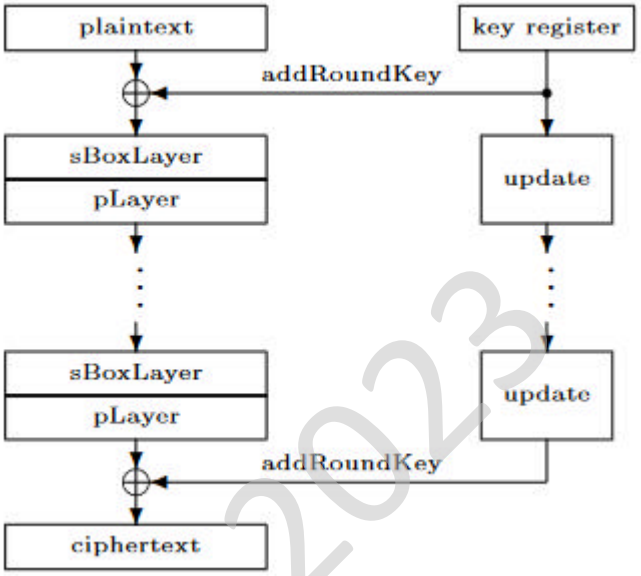

Рисунок 4.3 – Шифр PRESENT

Тепер кожна стадія визначається по черзі. Обґрунтування конструкції наведені нижче, а біти всюди нумеруються з нуля, починаючи із правого в блоці або слові.

Додавання раундового ключа (addRoundKey)

Заданий раундовий ключ $K_i = k^i_{63} \ldots k^i_0$, де $1 \leq i \leq 32$, а так само поточний стан $b_{63} \ldots b_0$. Додавання раундового ключа до поточного стану відбувається за модулем 2 ($b_j = b_j \oplus k^i_j$, де $0 \leq j \leq 63$).

Шар S-блоків (sBoxlayer)

Використовувані в PRESENT S-блоки відображають 4-бітні блоки в 4-бітні блоки. Дія цього блоку в шістнадцятковій системі числення наведене в наступній таблиці.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Рисунок 4.4 – Дія блоку в шістнадцятковій системі числення

Для шару S-блоків поточний стан $b_{63}\ldots b_0$ представляє із себе 16 4-бітних слів $w_{15}\ldots w_0$, де $w_i = b_{4*i+3} \,||\, b_{4*i+2} \,||\, b_{4*i+1} \,||\, b_{4*i}$ для $0 \le i \le 15$. Вихід рамки $S[w_i]$ видає оновлені значення станів очевидним образом.

Шар перестановки (player)

Побітова перестановка, використовувана в PRESENT задається наступною таблицею (біт і стану зміщається на позицію P(i).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Рисунок 4.5 – Побітова перестановка, використовувана в PRESENT

Перетворення ключа (The key schedule)

PRESENT може використовувати 80- і 128-бітні ключі, однак зосередимося на 80-бітній версії. Надаваний користувачем ключ зберігається в регістрі ключів K, представлений у вигляді $k_{79}k_{78}\ldots k_0$. На i-ом раунді 64-бітний раундовий ключ $Ki = k_{63}k_{62}\ldots k_0$, що полягає з 64 лівих битов поточного вмісту регістру K. Таким чином, на i-ом раунді маємо:

$K_i = k_{63}k_{62}\ldots k_0 = k_{79}k_{78}\ldots k_{16}$.

Після розпакування раундового ключа K_i регістр ключа $K = k_{79}k_{78}\ldots k_0$ обновляється в такий спосіб:

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$

2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$

3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus$ round_counter

Отже, регістр ключа зрушується на 61 позицію вліво, 4 крайніх лівих біта, що пройшли через S-блок і round_counter значення і складається за модулем 2 з бітами $k_{19}k_{18}k_{17}k_{16}k_{15}$ з K з найменшим значущим бітом з round_counter праворуч.

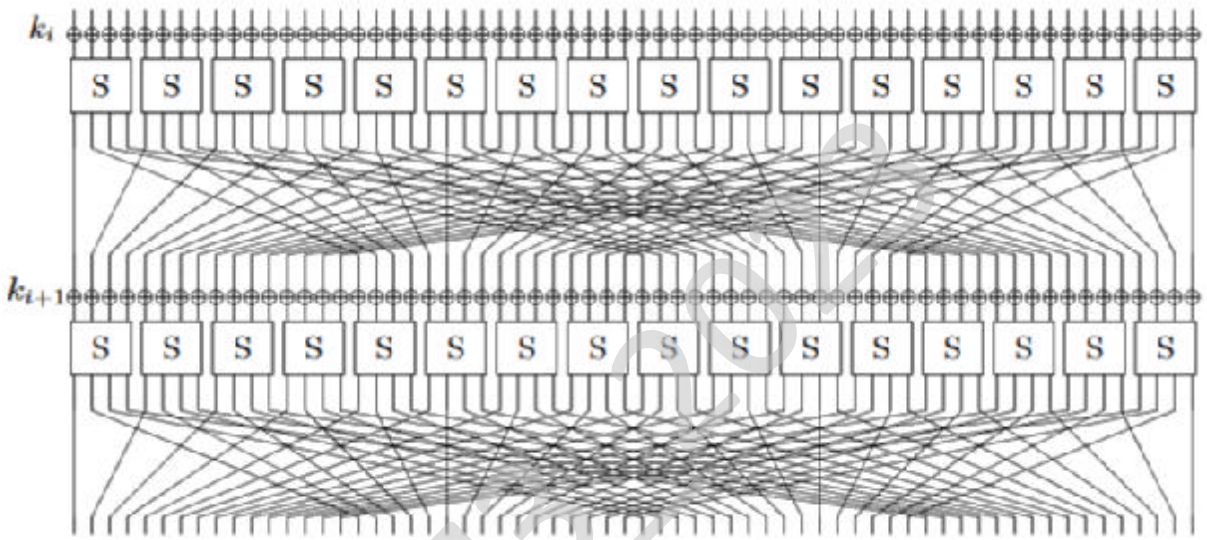

Рисунок 4.6 – Перетворення ключа (The key schedule)

Перетворення ключа для 128-бітного алгоритму можна знайти в додатку до оригінальної статті.

Конструктивні особливості PRESENT

Крім безпеки й ефективної реалізації, основне досягнення PRESENT – його простота. по цьому не дивно, що схожі проекти були прийняті в інших обставинах, і навіть були використані як навчальний посібник для студентів. У даній секції ми обґрунтуємо рішення, прийняті нами при проектуванні PRESENT. Однак, у першу чергу, опишемо очікувані прикладні вимоги.

Цілі й середовище застосування

При проектуванні блокового шифру, застосовного в жорстко обмежених оточеннях, важливо зрозуміти, що ми не створюємо блоковий шифр, який, неодмінно, буде застосовний у багатьох ситуаціях – для цього існує AES. Навпаки, ми націлені на досить специфічне застосування, для якого AES не підходить. Вищесказане визначає нам наступні характеристики:

– Шифр буде реалізований «у залізі»

– Додатки будуть вимагатися лише для регулювання рівня безпеки. Отже, 80-бітний ключ буде здоровим розв'язком. Відзначимо, що такої ж позиції дотримуються розроблювачі потокових шифрів проекту eSTREAM.

– Додатки не припускають шифровки великої кількості даних. Таким чином, реалізація може бути оптимізована для продуктивності або простору без внесення занадто великих змін.

– У деяких застосуваннях можлива ситуація, що ключ буде зафіксований при виробництві. У такому випадку, не треба буде змінювати ключ пристрою (що може вилитися в атаки з маніпуляцією ключем .

– Фізичний обсяг пристрою буде першим пріоритетом, після безпеки, що спричинить обмеження на пікові й середні споживання енергії, і, отже, зрушить швидкодія в область низькопріоритетних параметрів.

– У пристроях, що вимагають найбільш ефективного використання фізичного простору, блоковий шифр найчастіше зможе лише шифрувати дані (encryption-only mode). Таким чином, він зможе бути використаний у запит-відповідь (challenge-response) протоколах авторизації, і, при дотриманні контролю стану, може бути використаний для шифровки й дешифрування переговорів із пристроєм, використовуючи режим лічильника.

Виходячи з таких міркувань, розв'язали створити PRESENT як 64-бітний блоковий шифр із 80-бітним ключем. Шифровка й дешифрування, у цьому випадку, мають приблизно схожі фізичні вимоги. Маючи можливість підтримувати як шифрацію, так і дешифрацію, PRESENT буде компактніше, чим

підтримуючий лише шифрацію AES. А у випадку encryption-only виконання, наш шифр виявиться й зовсім понад-легко. Суб-ключі що шифрують будуть обчислюватися на ходу.

У літературі є безліч прикладів атак компромісу між часом, датою й пам'яттю, або атак з використанням парадокса днів народження при шифровці великих обсягів даних. Однак, дані атаки залежать тільки від параметрів шифру й не використовують внутрішню структуру. Наша мета полягає в тому, щоб ці атаки були кращим, що можуть застосувати проти нас. Атаки стороннього каналу й атаки з безпосереднім зломом чипа загрожують PRESENT тією самою мірою, як і іншим криптографічним примітивам. Однак для ймовірних застосувань, помірні вимоги безпеки роблять вигоду, одержувану зловмисником на практиці, досить обмеженої. В оцінці ризиків, подібні погрози не сприймаються як істотний фактор.

Перестановочний шар

При виборі шару змішування ключа, наша увага до апаратної ефективності вимагає наявності лінійного шару, який може бути реалізований з мінімальною кількістю керуючих елементів (наприклад, транзисторів. це приводить до побітової перестановки. Приділяючи увагу простоті, ми вибрали регулярну бітову перестановку, що допомагає провести прозорий аналіз безпеки.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

1. Меню програми:

– Контакти.

– Налаштування.

– Довідка.

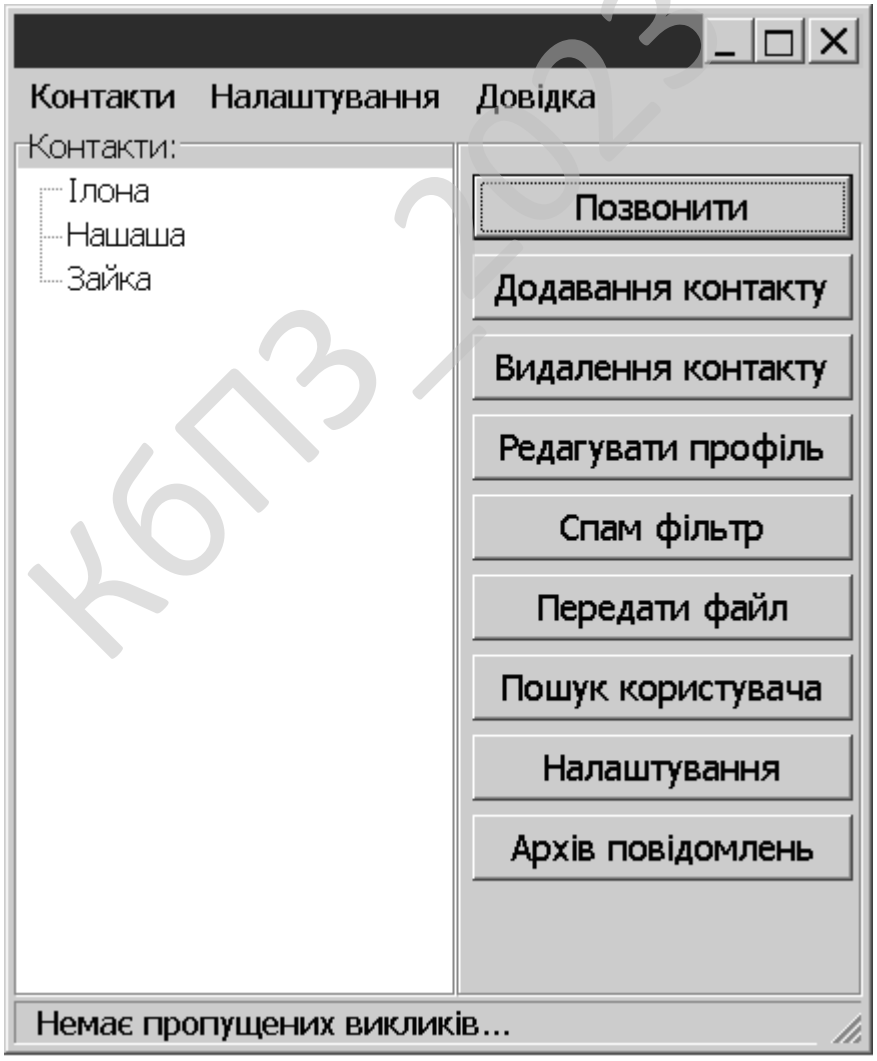

Рисунок 5.1 – Головне вікно програми

2. Блок контактів – список контактів користувача.

3. Функціональні можливості програми:

– Подзвонити.

– Додавання контакту.

– Видалення контакту.

– Редагувати профілю.

– Спам фільтр.

– Передати файл.

– Пошук користувача.

– Налаштування.

– Архів повідомлень.

На рисунку 5.2 зображено форму авторського права, з якої можливо отримати наступну інформацію:

– Автор проекту.

– Версія проекту.

– Місце розробки проекту.

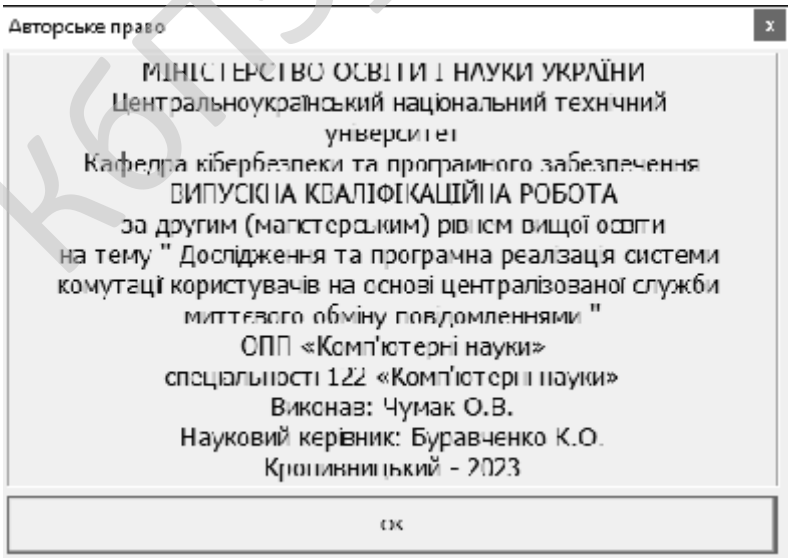

Рисунок 5.2 – Довідка автора

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Метою розробки є дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Об'єктом дослідження є процес комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Предметом дослідження є методи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Методи дослідження базуються на методах теорії телетрафіку, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

– Розроблено вітчизняний продукт комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір та системні потреби;

б) незалежність від встановлених на комп'ютері баз даних;

в) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	60
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B \qquad (7.1)$$

де А – коефіцієнт Боема, А=2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПV_j, \tag{7.3}$$

де $ПV$j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \tag{7.4}$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot146=245 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	245	Ф 7.1-7.4
Впровадження	13	Д13
Всього	286	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів,

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{347 \cdot 1}{48\text{-}5} = 8 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	12	1080	18
Монітор	60	14	840	14
Клавіатура	30	12	360	6
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	350	875	14,58
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	65,24

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{mic}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{65,24 \cdot 2}{1,2} = 109 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=109/(48·8)=0,28 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів–електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	18330	36660
Продакт-менеджер	0,5	14000	14000
Інженер-програміст	8	16000	256000
Інженер-електронщик	0,28	14000	7840
Інженер-системотехнік	0,2	14000	5600
Адміністратор мережі	0,2	14000	5600
Системний програміст	0,2	14000	5600
Дизайнер WEB	0,2	15000	6000
Інженер-верстальник	0,2	14000	5600
Бухгалтер-економіст	0,2	15000	6000
Всього за період розробки	$R_{сп}$=10,98	-	$Ф_{роб}$=348900

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{348900}{10,98 \cdot 48} = 662 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць.

$S_{у}$ – питома площа на одне робоче місце, $м^2$,

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 17.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	Intel Xeon E5-1620 v4 (4 (8) ядер по 3.5 – 3.8 GHz), 10 MB Cache	-
Системна плата	HP Z440, 2x PS/2, 2x USB 2.0, 8x USB 3.0, 1x LAN (RJ-45), 4x Audio, 1x DVI, 2x DisplayPort	-
Жорсткий диск	240 GB SSD	-
Оперативна пам’ять	8 GB DDR4	-
Відеоадаптер	nVidia Quadro K2200, 4 GB GDDR5, 128-bit	
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	HP Workstation Z440 Tower	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де $N_е$ – Кількість екземплярів програм, шт.

$$З_о = 662 \cdot 286/60 = 3158 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних т суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 3158 \cdot 10 \cdot 0{,}01 = 316 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основно та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(3158+316) = 764 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N. \quad (7.16)$$

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д., \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 33 грн./шт., DVD-R box – 49 грн./шт.

$$З_{М2} = 49 \cdot 10 = 490 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають ус друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (80 + 490 + 1702)/60 = 38 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційни систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де $Н_п$ – норматив витрат на освоєння нових мов програмування, %

$$О_п = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програ ($N_е$ = 60 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 174246 \cdot 2/(60 \cdot 12) = 484 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтям калькуляції

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 3158+316+764+474+38+474+484 = 5708 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмно продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де $Р_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 5708 = 2854 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна цін програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмног забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	3158
2. Додаткова зарплата виконавців	$З_д$	316
3. Відрахування на соціальні потреби	$С_{оц}$	764
4. Загальногосподарські витрати	$Г_{осп}$	474
5. Витрати на матеріали	$З_М$	38
6. Освоєння нових операційних систем, мов програмування	$О_п$	474
7. Амортизація основних фондів	$А_м$	484
8. Повна собівартість програмного забезпечення	$С_п$	5708
9. Плановий прибуток	$П_р$	2854
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	8562
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1712,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	10274,4

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаєм на основі балансової вартості основних фондів, яка враховує ціну, транспортно заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт а також витрати на випробування у виробничих умовах. Результати розрахункі зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживач програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	10274
Всього капітальних витрат	–	10274

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо пр умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмно продукції

Найменування статей витрат	Позна-чення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на оплату праці	$З_р$	87230	31537
2. Витрати на електроенергію	$З_{ел}$	621	108
3. Витрати на амортизацію	$З_{ам}$	0	5137
Всього витрат за рік	$І$	87851	36782

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де $Т_р$ – кількість годин роботи з системою за рік, *год.*,

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення час на виконання завдання зменшилася з 650 годин на рік до 235 годин на рік, тому витрати зменшилися з

$$З_{р\ баз} = 650 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 87230\ грн.$$

до

$$З_{р\ нов} = 235 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 31537\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел}=П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз=0,2·1350·2,3=621 грн

Зел нов=0,2·235·2,3=108 грн

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	10274	–	5137
Всього відрахувань	-	–	10274	–	5137

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (8562 - 5708) \cdot 60 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 142199 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{361406}{(8562 - 5708) \cdot 60 \cdot 12 / 2} = 0,4 \; років$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (87851\text{-}36782)\text{-}0,5 \cdot 10274 = 45932 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{10274}{87851 - 36782} = 0,2 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	60
2. Повна собівартість розробленої програми	Грн.	5708
3. Ціна розробленої програми	Грн.	8562
4. Плановий прибуток від реалізації розробленої програми	Грн.	2854
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	171240
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	142199
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,4
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	10274
11.Величина економічного ефекту у користувача програмної продукції	Грн.	45923
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона здоров'я працівників, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму повинна складати одну з головних завдань роботодавця.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту.

В зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером.

Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

8.2 Аналіз умов праці програміста

Умови праці в приміщенні, в якому знаходиться робоче місце програміста є сприятливими. Приміщення обладнане автономною системою газового опалення, основною перевагою якого є програмування режиму роботи в залежності від

погодних умов, оскільки клімат є нестійким. Використовується система природної та штучної вентиляції, що забезпечує ефективну циркуляцію повітря. В кабінеті знаходяться 1 кондиціонер (CARRIER 42QCR018/38QCR018).

Засоби копіювальної техніки знаходяться на достатньо далекій відстані від робочих місць, оскільки приміщення складає 20 м^2, а у відділі налічується два працівники, тобто концентрація озону та оксиду азоту в повітрі є невисокою. Таким чином на кожного програміста приходиться 10,0 м^2 що відповідає нормам Державним санітарним правилам і нормам ДСанПіН 3.3.2.007-98 [1]. Висота стелі приміщення складає 3 метри, що також не порушує нормативні вимоги.. Прибиральники підтримують порядок в службових приміщеннях, дотримуються санітарно-гігієнічних норм по прибиранню приміщень, витирають пил, підмітають підлогу наприкінці кожного робочого дня.

В цілому потрібно відмітати застаріліть офісної техніки та відсутність клавіатур з ергономічною розкладкою та рідкокристалічних моніторів, які здійснюють менш негативний вплив на стан здоров'я працівників відділу.

Оформлення інтер'єру приміщення є відповідає вимогам з ергономіки та стимулює працівників до підвищення працездатності та зниження втоми. Стеля білого кольору створює оптичний ефект збільшення висоти приміщення, підлога пофарбована коричневим кольором, а стіни – у жовтий. Перевагами даного кольору є створення відчуття теплоти, здатність привертати увагу без додаткової втоми.

Висота столу складає 72,5 см, до того ж його можна регулювати відповідно до власних потреб. Стіл має достатній внутрішній об'єм, завдяки ширині у 73 см та висоті простору під столом – у 64 см, є достатньо важким для забезпечення стійкості. Крісла забезпечують фізіологічно раціональну позу, мають підлокітники, здатні обертатися та регулятор висоти, кута нахилу спинки й відстані спинки від краю сидіння.

В кабінеті створено оптимальні умови праці відносно температури, вологості приміщення та вентиляції.

Наприкінці аналізу небезпечних факторів праці побудуємо підсумкову таблицю 8.1.

Таблиця 8.1 – Підсумкова таблиця значень параметрів небезпечних факторів праці

Найменування параметра	Значення параметра		Нормативний документ
	Фактичне	Нормоване	
1	2	3	4
Освітленість штучна, лк	300	300	ДБН.В 2.5-28:2018 [2]
Значення КПО,%	1,0	1,1	ДБН.В 2.5-28:2018 [2]
Повітрообмін,м /год			
взимку	76	80	ДСН 3.3.6.042-99[3]
влітку	36	80	ДСН 3.3.6.042-99 [3]
Температура повітря. °С			
взимку	22	21-25	ДСанПіН 3.3.2-007-98
влітку	24	27-28	ДСанПіН 3.3.2-007-98
Відносна вологість,%			
взимку	60	<75	ДСанПіН 3.3.2-007-98
влітку	55	<60	ДСанПіН 3.3.2-007-98
Швидкість переміщення повітря, м/с			
взимку	0,16	<0,2	ДСанПіН 3.3.2-007-98
влітку	0,10	<0,2	ДСанПіН 3.3.2-007-98

Щодо вимог електробезпеки, то приміщення за небезпекою ураження електричним струмом можна віднести до 1 класу, тобто це приміщення без підвищеної небезпеки (сухе, без пилу, з нормальною температурою повітря, ізольованими підлогами і малим числом заземлених приладів).

Для запобігання поразки електричним струмом в приміщенні відділу використовується ряд організаційно-технічних заходів: розташування проводів

живлення поза зоною пересування людей; допуск до роботи електроприладів тільки тих робітників, що знайомі із технікою безпеки; використання мережних продовжувачів з вбудованими запобіжниками на 0,1 А; при ремонті обладнання персонал попереджується.

Устаткування, що працює в приміщенні живиться від мережі 220В та частотою 50Гц. Споживачами цієї напруги є також джерела штучного освітлення. Вони розташовуються на висоті 3 м, що задовольняє нормі, відповідно до якого джерела освітлення повинні розташовуватися на висоті 2,5 м від підлоги.

Проводка схована. У якості розеток для підключення устаткування застосовуються розетки з заземленим кожухом, захищеного від випадкового доторку до струмоведучих частин. Електроустаткування, що знаходиться в приміщенні відділу відноситься до установок напругою до 1000В.

На робочому місці програміста з всього устаткування металевим є лише корпус системного блоку комп'ютера, але тут використовуються системні блоки, що відповідають стандартові фірми IBM, у яких крім робочої ізоляції передбачений елемент для заземлення і провід з жилою, що заземлює, для приєднання до джерела живлення.

Основні причини ураження людини електричним струмом на робочому місці:

- дотик до металевих неструмоведучих частин (корпусу, периферії комп'ютера), що можуть виявитися під напругою в результаті ушкодження ізоляції:
- нерегламентоване використання електричних приладів:
- відсутність інструктажу співробітників з правил електробезпеки.

На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кВ/м, тобто в 10 разів перевищує норму 20 кВ м.

Отже за результатами проведеного аналізу можна зробити висновки, що всі показники знаходяться у межах запропонованих значень

8.3 Розробка заходів пожежної безпеки

Ступінь вогнестійкості будинків приймається в залежності від їхнього призначення, категорії по вибухопожежній і пожежній небезпеці, по поверховості, площі поверху в межах пожежного відсіку згідно ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків, установок за вибухопожежною та пожежною небезпекою [4]

За критеріями по пожежній безпеці будівля відноситься до категорії Д. За особливостями функціонування установи вибухонебезпечних парів та концентратів не помічено. Будівля побудована мармуроподібного вапняка межа стійкості складає 0,5-2,5 години.

Для профілактики пожежі надзвичайно важлива правильна оцінка пожежонебезпеки будинку, визначення небезпечних факторів і обґрунтування способів і засобів пожежопопередження і захисту.

Одне з умов забезпечення пожежобезпеки – ліквідація можливих джерел запалення.

У приміщенні програмістів джерелами запалення можуть бути:

- несправне електроустаткування, несправності в електропроводці, електричних розетках і вимикачах. Для виключення виникнення пожежі з цих причин необхідно вчасно виявляти й усувати несправності, проводити плановий огляд і вчасно усувати всі несправності;
- несправні електроприлади. Необхідні міри для виключення пожежі містять у собі своєчасний ремонт електроприладів, якісне виправлення поломок, не використання несправних електроприладів;
- обігрівання приміщення електронагрівальними приладами з відкритими нагрівальними елементами. Відкриті нагрівальні поверхні можуть спричинити пожежу, тому що в приміщенні знаходяться паперові документи, а папір – легкозаймистий предмет. З метою профілактики пожежі пропоную не використовувати відкриті обігрівальні прилади в приміщенні лабораторії;

- коротке замикання в електропроводці. З метою зменшення імовірності виникнення пожежі внаслідок короткого замикання необхідно, щоб електропроводка була схованою.

- влучення в будинок блискавки. У літній період під час грози можливе влучення блискавки внаслідок чого можливий пожежа. Щоб уникнути цього я рекомендую установити на даху будинку блискавковідвід;

- недотримання мір пожежної безпеки і паління в приміщенні також може спричиниш пожежу. Для усунення загоряння в результаті паління в приміщенні лабораторії пропоную категорично заборонити паління, а дозволити тільки в строго відведеному для цього місці.

За протипожежним режимом визначені: місця куріння, правила проїзду та стоянки транспортних засобів, порядок прибирання пилу й відходів, відключення від мережі електрообладнання у разі пожежі, огляду і закриття приміщень після закінчення операційного дня, проходження посадовими особами навчання та тестування знань з питань пожежної безпеки, проведення з програмістами протипожежйих інструктажів, організації експлуатації і обслуговування технічних засобів протипожежного захисту, проведення планово-попереджувальних ремонтів і оглядів інженерного обладнання, дії працівників при виявленні пожежі.

8.4 Розробка заходів по електробезпеці для приміщень з ПЕОМ

Згідно ДБН В.2.5 -28:2018 Державних будівельних норм ДБН В.2.5 -28:2018 [5] при проектуванні систем електропостачання, при монтажі силового електроустаткування і електричного освітлення і в будівлях і приміщеннях для ЕОМ необхідно дотримуватися вимог нормативно-технічної документації.

ПЕОМ є однофазним споживачем електроенергії, що живиться змінним струмом напругою 220В і частотою 50Гц, від мережі із заземленою нейтраллю. За засобами захисту людини від поразки електричним струмом ЕОМ повинне відповідати першому класу захисту. Захист від випадкового дотику до

струмоведучих частин забезпечують конструктивні, схемно-конструктивні і експлуатаційні заходи захисту. Комплекс необхідних заходів по електробезпеці визначається, виходячи з видів електроустановки, її номінальної напруги, умов середовища, типу приміщення і доступності електроустаткуванні.

В приміщенні розміщено декілька комп'ютерів, то кабель прокладають в металевих трубах і гнучких металевих рукавах з відведеннями. Якщо ЕОМ розміщені в центрі приміщення, електромережа прокладається в каналах або під знімною підлогою в металевих трубах і гнучких металевих рукавах.

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються ветикальні електроди: метелевий куток 50·50·5 мм., довжиною L=3 м., та горизонтальний електрод – метелева полоса з перетином 40·4 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір ρ_2 = 40 Ом·м). Умовна товщина верхнього шару грунта: H=0,4 м. Відстань між вертикальними заземлювачами (електродами) A=3 м. Глибина закладення горизонтального контура заземлення t=0,8 м. Опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр. 13-16 [6], або аналогічної.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,8+3/2=2,3 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho_2 = 1{,}36 \cdot 40 = 54{,}5 \text{ Ом·м.}$$

де $\psi = 1{,}36$ – табличне значення коефіцієнта сезонності для відповідной климатичной зони у багатошаровому грунті [6];

$\rho_1 = 50$ Ом·м. – табличне значення питомого опору верхнього шару грунта [11];

$\rho_2 = 40$ Ом·м. – табличне значення питомого опору нижнього шару грунта (глина) [11].

Зелёное выкинуть, заменить на:

$\rho = 40$ Ом·м. – табличне значення питомого опору нижнього шару грунта (глина) [11].

Еквівалентний діаметр ветикальньного електрода (кутка) [11]:

$$Dв = 0{,}95 \cdot К = 0{,}95 \cdot 50 = 47{,}5 \text{ мм.} = 0{,}0475 \text{м.}$$

де К= 50 мм. – розмір метелевого кутка (задан).

Опір розтіканню електричного струму одного електрода вертикальног заземлювача (без врахуванням заглиблення заземлювача) [11]:

$$R_O = 0{,}366(\rho/L)\lg(4L/Dв) = 0{,}366(54{,}5/3)\lg(4 \cdot 3/0{,}0475) = 15{,}9 \text{ Ом.}$$

Відношення A/L=3/3=1.

Опір розтіканню електричного струму одного електрода вертикальног заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O = 0{,}366(\rho/L)[\lg(2L/Dв)+(1/2)\lg((4t+L)/(4t-L))] =$$

$$= 0{,}366(54{,}5/3)[(\lg(2 \cdot 3/0{,}0475)+(1/2)\lg((4 \cdot 2{,}3+3)/(4 \cdot 2{,}3-3))] =$$

$$= 14{,}9 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів Кев=0,8 пр орієтовній кількості ветикальньних електродів, яке дорівнює 4 [11].

Визначаємо необхідну кількість вертикальних заземлювачів (без вхаруванн горизонтального заземлювача), при $R_{ЗН} = 4$ Ом :

$$N = R_O / (Кев \, R_{ЗН}) = 14{,}9 / (0{,}8 \cdot 4) = 4{,}66 \approx 5 \text{ шт.}$$

Визначаємо длину з’єднуючої полоси:

$$L_П = 1{,}05 \cdot A \cdot N = 1{,}05 \cdot 3 \cdot 4{,}66 = 14{,}6 \approx 15 \text{ м.}$$

Опір розтіканню електричного струму з’єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П = 0{,}366(\rho_2 \cdot K_П / L_П) \lg(2(L_П \cdot L_П)/(B \cdot t)) =$$

$$= 0{,}366(40 \cdot 5/14{,}6) \cdot \lg((2 \cdot 14{,}6^2)/(0{,}04 \cdot 0{,}8)) = 20{,}5 \text{ Ом.}$$

де $K_П$=5 – табличне значення кліматичного коефіцієнта питомого опору грунт для відповідної климатичной зони для з’єднуючої полоси [11]:

B = 40 мм. = 0,04 м.- ширина з’єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [11]:

$$R = (R_О \cdot R_П)/(R_О \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (14{,}9 \cdot 20{,}5)/(14{,}9 \cdot 0{,}75 + 4{,}66 \cdot 20{,}5 \cdot 0{,}8) = 3{,}5 \text{ Ом.}$$

де $\eta_П$ = 0,75 – табличне значення коефіцієнта екранування з’єднуючої полос [11].

Умова $R \le R_{ЗН}$ виконується ($3{,}5 \le 4$).

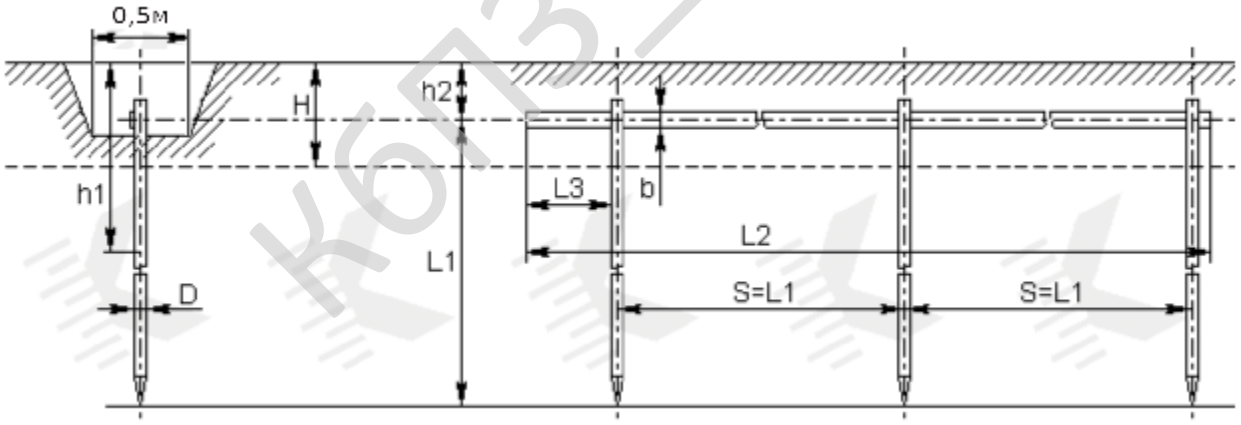

Рисунок 8.1 – Схема штучного заземлення

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

– Досліджена система комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

– На основі отриманих результатів досліджень створена програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм PRESENT.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 45923 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Чумак О.В. Дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.
2. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009
3. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.
4. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
5. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.
6. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. – Д.: НГУ, 2016. – 187 с.
7. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph.. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.
8. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.
9. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.
10. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

11. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

12. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

13. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

14. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

15. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

16. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

17. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

18. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

19. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

20. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

21. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

22. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

23. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

24. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

25. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

26. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

27. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

28. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

29. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

30. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

31. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

32. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

33. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

34. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

35. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

36. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

37. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

38. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

39. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

40. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

41. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

42. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

43. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

44. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

45. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

46. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

47. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". – Випуск 2 (118). т.2. – Х.: ХУПС – 2014. – С. 64-67

48. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. – 2014. – С. 240.

49. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. Коваленка О.В., Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки України від 26.02.2013 року № 1/11-4368. – Кіровоград: КНТУ 2013. – 257с.

50. Смірнов О.А., Дрєєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

51. Смірнов О.А., Доренський О.П., Дрєєв О.М. Аналіз процесів стиснення та відновлення зображень на основі цифрових методів. Наука і техніка Повітряних сил Збройних Сил України. – Випуск 3(12). – Х.: ХУПС. – 2013. – С.122-127.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 33-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута розробка заходів по електробезпеці для приміщень з ПЕОМ.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 116 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 12.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Буравченко К.О.

Дослідження та програмна реалізація системи комутації користувачів на основі централізованої служби миттєвого обміну повідомленнями

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2023 року

ChoixCouleurPanel.pas – параметри інтерфейсу клієнтської частини

```
unit ChoixCouleurPanel;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls,Registry, WinSkinStore, WinSkinData,
  ComCtrls, ExtDlgs;

type
  TChoixCouleur = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    GroupBox1: TGroupBox;
    RadioGroupSons: TRadioGroup;
    RadioGroupVisible: TRadioGroup;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    GroupBox4: TGroupBox;
    Image5: TImage;
    GroupBox5: TGroupBox;
    StaticText3: TStaticText;
    Couleur: TComboBox;
    Panel1: TPanel;
    BitBtn1: TBitBtn;
    GroupBox6: TGroupBox;
    GroupBox7: TGroupBox;
    GroupBox10: TGroupBox;
    GroupBox11: TGroupBox;
    CheckBox4: TCheckBox;
    CheckBox5: TCheckBox;
    GroupBox12: TGroupBox;
    Absent: TCheckBox;
    Temps: TEdit;
    OpenTextFileDialog1: TOpenTextFileDialog;
    SpeedButton1: TSpeedButton;
    GroupBox8: TGroupBox;
    Label1: TLabel;
    Fichier: TEdit;
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure CouleurClick(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
  private
    { Private declarations }
    procedure SystemeCommande(var Msg: TMessage);
      message WM_SysCommand; // перехоплення повідомлень SysCommand

     procedure ReadSkinfile( apath : string );

  public
    { Public declarations }
  end;

var
  ChoixCouleur: TChoixCouleur;

implementation
```

```
uses Unit_IM, Unit_MsgPerso;


{$R *.dfm}

procedure TChoixCouleur.FormCreate(Sender: TObject);
var
  Registre : TRegistry;
begin
  readskinfile(path);

  Registre:=TRegistry.Create;
  Registre.RootKey:=HKEY_LOCAL_MACHINE;

  Registre.OpenKey('\Software\IntraMSN\Couleur\',True);
  if Registre.ValueExists('Couleur') then
   Couleur.Text:=Registre.ReadString('Couleur');

  Registre.OpenKey('\Software\IntraMSN\Son\',True);
  if Registre.ValueExists('Son') then
    RadioGroupSons.ItemIndex := Registre.ReadInteger('Son');

  Registre.OpenKey('\Software\IntraMSN\Afficher\',True);
  if Registre.ValueExists('Afficher') then
    RadioGroupVisible.ItemIndex := Registre.ReadInteger('Afficher');

  Registre.OpenKey('\Software\IntraMSN\Image\',True);
  if Registre.ValueExists('Image') then
  begin
    Image5.Width := Registre.ReadInteger('Image');
    Image5.Height := Registre.ReadInteger('Image');
  end;

  Registre.OpenKey('\Software\IntraMSN\Transfert\',True);
  if Registre.ValueExists('Fichier') then
    Fichier.Text := Registre.ReadString('Fichier')
  else
    Fichier.Text := root;

  Registre.OpenKey('\Software\IntraMSN\Absent\',True);
  if Registre.ValueExists('Activer') then
    Absent.Checked := Registre.ReadBool('Activer');
  if Registre.ValueExists('Activer') then
    Temps.Text := Registre.ReadString('Temps');

  Registre.CloseKey;
  Registre.Free;
end;

procedure TChoixCouleur.BitBtn1Click(Sender: TObject);
var
  Registre : TRegistry;
begin
  Registre:=TRegistry.Create;
  Registre.RootKey:=HKEY_LOCAL_MACHINE;

  Registre.OpenKey('\Software\IntraMSN\Couleur\',True);
  Registre.WriteString('Couleur',  _IM.sd1.SkinFile);

  Registre.OpenKey('\Software\IntraMSN\Son\',True);
  Registre.WriteInteger('Son', RadioGroupSons.ItemIndex);

  Registre.OpenKey('\Software\IntraMSN\Afficher\',True);
  Registre.WriteInteger('Afficher', RadioGroupVisible.ItemIndex);

  Registre.OpenKey('\Software\IntraMSN\Image\',True);
  Registre.WriteInteger('Image',Image5.Width);
  _IM.ClientDataSet1IMAGE.DisplayWidth := Round(Image5.Width / 6);
```

```
  Registre.OpenKey('\Software\IntraMSN\Absent\',True);
  Registre.WriteBool('Activer',Absent.Checked);
  Registre.WriteString('Temps',Temps.Text);

  Registre.OpenKey('\Software\IntraMSN\Transfert\',True);
  Registre.WriteString('Fichier',Fichier.Text);

  Registre.CloseKey;
  Registre.Free;

  ChoixCouleur.Close;
end;

procedure TChoixCouleur.SpeedButton1Click(Sender: TObject);
begin
  if(OpenTextFileDialog1.Execute)Then
    Fichier.Text := OpenTextFileDialog1.FileName;
end;

procedure TChoixCouleur.SystemeCommande(var Msg: TMessage);
begin
  if Msg.wParam = sc_Close then ;
end;

procedure TChoixCouleur.Image1Click(Sender: TObject);
var
  i : integer;
begin
  Image5.Width := (Sender as TImage).Width;
  Image5.Height := (Sender as TImage).Height;
end;

procedure TChoixCouleur.CouleurClick(Sender: TObject);
begin
   _IM.sd1.SkinFile:=path+Couleur.Text;
   if not _IM.sd1.Active then _IM.sd1.Active:=true;
end;

procedure TChoixCouleur.ReadSkinfile( apath : string );
var
  sts: Integer ;
  SR: TSearchRec;
  list: Tstringlist;

  procedure AddFile;
  begin
    list.add(sr.name);
  end;

begin
  list:=Tstringlist.create;
  sts := FindFirst( apath + '*.skn' , faAnyFile , SR );
  if sts = 0 then begin
      if ( SR.Name <> '.' ) and ( SR.Name <> '..' ) then begin
          if pos('.', SR.Name) <> 0 then
            Addfile;
      end;
      while FindNext( SR ) = 0 do begin
          if ( SR.Name <> '.' ) and ( SR.Name <> '..' ) then begin
              if Pos('.', SR.Name) <> 0 then  Addfile;
          end;
      end;
  end ;
  FindClose( SR ) ;
  list.sort;
  Couleur.items.assign(list);
  list.free;
end;
```

```
end.
```

Клієнтська частина
Unit_IM.pas – основна частина клієнтського додатку

```
unit Unit_IM;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, WinSock, Buttons, ScktComp, ExtCtrls, Grids,
  ValEdit, Menus, Math, ShellAPI, cxControls, cxSplitter, cxStyles,
  cxCustomData, cxGraphics, cxFilter, cxData, cxDataStorage, cxEdit, cxImage,
  cxGridCustomTableView, cxGridTableView, cxGridCustomView, cxClasses,
  cxGridLevel, cxGrid, cxBlobEdit, ImgList, cxTextEdit, cxGridBandedTableView,
  DB, DBClient, dxGDIPlusClasses, cxDBData, cxGridDBTableView, cxImageComboBox,
  ActnList, XPStyleActnCtrls, ActnMan, cxLookAndFeels, cxLookAndFeelPainters,
Registry,
  WinSkinData, WinSkinStore, dxBar;

const
  WM_CAllBack = WM_USER;
  WM_MYMESSAGE = WM_USER + 100;

type
  T_IM = class(TForm)
    ClientSocket: TClientSocket;
    Memo_ICQ_OSCAR_: TMemo;
    cxSplitter1: TcxSplitter;
    ImageListold: TImageList;
    ClientDataSet1: TClientDataSet;
    ClientDataSet1N_ID: TIntegerField;
    ClientDataSet1IMAGE: TGraphicField;
    ClientDataSet1NOMPRENOM: TStringField;
    DS_SOURCE: TDataSource;
    ActionManager1: TActionManager;
    ActionAjouterGroupe: TAction;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    IMG_LIGNE: TImage;
    IMG_OCCUPE: TImage;
    IMG_PAUSE: TImage;
    ClientDataSet3: TClientDataSet;
    DS_GROUPE: TDataSource;
    ClientDataSet3N_ID: TIntegerField;
    ClientDataSet3GROUPE: TStringField;
    ClientDataSet1N_GROUPE: TIntegerField;
    Panel2: TPanel;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    ComboBox1: TComboBox;
    IMG_1MN: TImage;
    IMG_TEL: TImage;
    IMG_ABS: TImage;
    ActionEMAIL: TAction;
    traymenu: TPopupMenu;
    Image1: TImage;
    SpeedButton4: TSpeedButton;
    ActionNavigateur: TAction;
    NavigateurWeb1: TMenuItem;
    BoitederceptionMail1: TMenuItem;
    N1: TMenuItem;
    ActionQuitter: TAction;
    Quitter1: TMenuItem;
    Statut1: TMenuItem;
    EnLigne1: TMenuItem;
    Occup1: TMenuItem;
    Autlphone1: TMenuItem;
    EnPause1: TMenuItem;
```

```
Abs1mn1: TMenuItem;
Absent1: TMenuItem;
ClientDataSet1ORDI: TStringField;
SkinStore1: TSkinStore;
sd1: TSkinData;
Action1: TAction;
ActionFermer: TAction;
Action4: TAction;
Action5: TAction;
Action6: TAction;
Action7: TAction;
Action8: TAction;
Action9: TAction;
Action10: TAction;
Action11: TAction;
Action12: TAction;
Action13: TAction;
dxBarManager1: TdxBarManager;
dxBarManager1Bar1: TdxBar;
dxBarSubItem1: TdxBarSubItem;
dxBarSubItem2: TdxBarSubItem;
dxBarSubItem3: TdxBarSubItem;
dxBarSubItem4: TdxBarSubItem;
dxBarSubItem5: TdxBarSubItem;
dxBarButton1: TdxBarButton;
dxBarButton2: TdxBarButton;
dxBarButton3: TdxBarButton;
dxBarButton4: TdxBarButton;
dxBarSubItem6: TdxBarSubItem;
dxBarSubItem7: TdxBarSubItem;
dxBarButton5: TdxBarButton;
dxBarButton6: TdxBarButton;
dxBarButton7: TdxBarButton;
dxBarButton8: TdxBarButton;
dxBarButton9: TdxBarButton;
dxBarButton10: TdxBarButton;
dxBarButton11: TdxBarButton;
dxBarButton12: TdxBarButton;
dxBarButton13: TdxBarButton;
dxBarButton14: TdxBarButton;
TimerReconnexion: TTimer;
cxGrid3DBTableView1: TcxGridDBTableView;
cxGrid3Level1: TcxGridLevel;
cxGrid3: TcxGrid;
cxGrid3Level2: TcxGridLevel;
cxGrid3DBTableView2: TcxGridDBTableView;
cxGrid3DBTableView2N_ID: TcxGridDBColumn;
cxGrid3DBTableView2GROUPE: TcxGridDBColumn;
cxGrid3DBTableView1N_ID: TcxGridDBColumn;
cxGrid3DBTableView1NOMPRENOM: TcxGridDBColumn;
cxGrid3DBTableView1N_GROUPE: TcxGridDBColumn;
cxGrid3DBTableView1ORDI: TcxGridDBColumn;
cxGrid3DBTableView1IMAGE: TcxGridDBColumn;
ImageList1: TImageList;
procedure WM_CALLBACKPRO(var msg: TMessage); message wm_callBack;
procedure mvtFenetre(i: Integer);
function NomPcActuel: string;
function MessageInfo: string;
procedure FormCreate(Sender: TObject);
procedure TimerNombresClientsActuelsTimer(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
  ErrorEvent: TErrorEvent; var ErrorCode: Integer);
procedure AnalysePremiereInformation(MessageRecu: string);
procedure AnalyseDerniereInformation(MessageRecu: string);
```

```
    procedure ClientSocketRead(Sender: TObject; Socket: TCustomWinSocket);
    procedure LabeledEditMessageEcritKeyPress(Sender: TObject;
      var Key: Char);
    procedure Fermer1Click(Sender: TObject);
    procedure RendreVisiblePremierPlan;
    procedure PourquoiDeconnecte(raison: string; iden: integer);

    procedure DireQueOnSeDeconnecte;
    procedure BitBtnDeconnexionClick(Sender: TObject);
    procedure FormResize(Sender: TObject);

    procedure Minimize(Sender: TObject);

    procedure AnalyseMessageRecuParClient(Msg: string);
    procedure ActionAjouterGroupeExecute(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure ActionEMAILExecute(Sender: TObject);
    procedure ActionNavigateurExecute(Sender: TObject);
    procedure ActionQuitterExecute(Sender: TObject);
    procedure EnLigne1Click(Sender: TObject);
    procedure Action13Execute(Sender: TObject);
    procedure Action14Execute(Sender: TObject);
    procedure ActionFermerExecute(Sender: TObject);
    procedure Action10Execute(Sender: TObject);
    procedure TimerReconnexionTimer(Sender: TObject);
    procedure cxGrid3DBTableView1CellDblClick(Sender: TcxCustomGridTableView;
      ACellViewInfo: TcxGridTableDataCellViewInfo; AButton: TMouseButton;
      AShift: TShiftState; var AHandled: Boolean);

  private

  public

  end;

type
  TStructureListeConnecte = record
    LoginConnecte: string[30];
    NomOrdinateur: string[30];
    Iden: integer;
    img: string;
  end;

var
  Affichage: Boolean = False;
  root, path: string;
  _IM: T_IM;
  EditIPConnexion, EditPortClient, EditLogin: string;
  StructureOrdinateur, StructureOrdinateur002: array[1..52] of
TStructureListeConnecte;
  mvt, GroupeNB, NumeroArriveConnexion, NombresMaximumClients, NombreSecret:
integer;
  ServeurActif, ClientConnecter: boolean;
  Present: TDateTime;
  Hour, Min, Sec, MSec: Word;
  nbpersonne, _ICQ_OSCAR_HauteurDeDebut, _ICQ_OSCAR_LargeurDeDebut,
PageHauteurDebut, PageLargeurDebut: integer;
  TrayIcon: TNotifyIconData;
  blah: HICON;


function UserName(): string;


implementation

uses ChoixCouleurPanel, Unit_MsgPerso, AlertMsg, AudioVideo;

{$R *.dfm}
```

```
function QuelHeureEstIl: string;
begin
  Present := Now;
  DecodeTime(Present, Hour, Min, Sec, MSec);
  result := '[' + IntToStr(Hour) + ':' + IntToStr(Min) + ':' + IntToStr(Sec) +
']';
end;

function droite(substr: string; s: string): string;
begin
  if pos(substr, s) = 0 then result := '' else
    result := copy(s, pos(substr, s) + length(substr), length(s) - pos(substr,
s) + length(substr));
end;

function gauche(substr: string; s: string): string;
begin
  result := copy(s, 1, pos(substr, s) - 1);
end;

procedure T_IM.RendreVisiblePremierPlan;
begin
  if (ChoixCouleur.RadioGroupVisible.ItemIndex = 0) then
  begin
    Application.Restore;
    Application.BringToFront;
  end;
end;

function T_IM.NomPcActuel: string;
var
  Buffer: array[0..255] of char;
  BufferSize: DWORD;
begin
  BufferSize := sizeOf(Buffer);
  GetComputerName(@buffer, BufferSize);
  result := buffer;
end;


//Зменшення у трей форми

 procedure T_IM.Minimize;
begin
  mvtFenetre(1);
  _IM.Visible := False;
end;


//створення форми

 procedure T_IM.FormCreate(Sender: TObject);
var
  Registre: TRegistry;
  SysMenu: hMenu;
  IPServeur: string;
  I: Integer;
begin
 // Application.OnMinimize := Minimize;
  root := ExtractFilePath(ParamStr(0));
  path := root + 'vsskin\';

  Registre := TRegistry.Create;
  Registre.RootKey := HKEY_LOCAL_MACHINE;

  Registre.OpenKey('\Software\IntraMSN\Couleur\', True);
  if Registre.ValueExists('Couleur') then
    sd1.SkinFile := Registre.ReadString('Couleur');
```

```
  Registre.OpenKey('\Software\IntraMSN\Image\', True);
  if (Registre.ValueExists('Image')) then
    ClientDataSet1IMAGE.DisplayWidth := Round(Registre.ReadInteger('Image') /
6);

  Registre.CloseKey;
  Registre.Free;

  ClientDataSet1.CreateDataSet;
  GroupeNB := 1;
  EditIPConnexion := '10.1.1.27';
  EditPortClient := '2879';
  ComboBox1.Text := 'Можу розмовляти';
  SysMenu := GetSystemMenu(Handle, False);
  ModifyMenu(SysMenu, sc_Close, mf_ByCommand, sc_Close, '&Вихід з програми
!!!'#9'Alt+F4');
  SysMenu := GetSystemMenu(application.handle, false);
  ModifyMenu(SysMenu, sc_Close, mf_ByCommand, sc_Close, '&Вихід з програми
!!!'#9'Alt+F4');
  _IM.Height := Round(500 * (Screen.Width / 1024) / 1.3);
  _IM.Width := Round(300 * (Screen.height / 768) / 1.3);
  _ICQ_OSCAR_HauteurDeDebut := _IM.Height;
  _ICQ_OSCAR_LargeurDeDebut := _IM.Width;
  ChoixCouleur := TChoixCouleur.Create(_IM);
  CAudioVideo := TCAudioVideo.Create(_IM);
  Randomize;
  NombreSecret := RandomRange(1000, 9999);
  _IM.Left := screen.Width - _IM.Width;
  _IM.top := screen.height - _IM.height - 30;
  blah := application.Icon.Handle;
  Trayicon.cbSize := SizeOf(TNotifyIconData);
  Trayicon.Wnd := handle;
  Trayicon.szTip := 'eMessenger';
  Trayicon.uID := 1;
  TrayIcon.hIcon := blah;
  TrayIcon.uCallbackMessage := WM_CAllBack;
  Trayicon.uFlags := NIF_MESSAGE or NIF_ICON or NIF_TIP;
  Shell_NotifyIcon(NIM_ADD, @trayicon);


  mvtFenetre(1);
  _IM.Visible := true;
  mvtFenetre(-1);
   _IM.Paint;

  EditLogin := UserName();

  if (EditIPConnexion <> '') and (EditPortClient <> '') and
    ((strtoint(EditPortClient)) > 0) then
  begin
    IPServeur := EditIPConnexion;
    ClientSocket.Host := IPServeur;
    ClientSocket.Port := strtoint(EditPortClient);
    ClientSocket.Active := TRUE;
  end;

  ClientDataSet3.Insert;
  ClientDataSet3N_ID.Value := 1;
  ClientDataSet3GROUPE.Value := 'Gйnйral';
  ClientDataSet3.Post;
end;

procedure T_IM.WM_CALLBACKPRO(var msg: TMessage);
begin

  case msg.LParam of WM_LBUTTONDOWN:
      begin
        if _IM.Visible = true then
```

```
        begin
          mvtFenetre(1);
          _IM.Visible := False;
        end
        else
        begin
          mvt := 1;
          _IM.Visible := True;
          mvtFenetre(-1);
        end;
      end;
    WM_RBUTTONDOWN: traymenu.Popup(mouse.CursorPos.X, mouse.CursorPos.y);
  end;
end;

procedure T_IM.mvtFenetre(i: Integer);
var
  k, fin, offset: integer;
begin
  offset := screen.Height - GetSystemMetrics(SM_CYFULLSCREEN);
  mvt := 1;
  _IM.Left := screen.Width - _IM.Width;
  if (i < 0) then
    fin := screen.Height - _IM.Height - offset
  else
    fin := screen.Height;
  while (_IM.Top <> fin) do
  begin
    _IM.Top := _IM.Top + i;
    for k := 1 to 1000000 do
      mvt := 1;
  end;
  if (i < 0) then
    _IM.Top := screen.Height - _IM.Height - offset
  else
    _IM.Top := screen.Height;
  mvt := 0;
end;

function UserName(): string;
const
  cnMaxUserNameLen = 254;
var
  UserName2: string;
  nSize: DWord;
begin
  nSize := cnMaxUserNameLen - 1;
  SetLength(UserName2, cnMaxUserNameLen);
  GetUserName(Pchar(UserName2), nSize);
  SetLength(UserName2, nSize - 1);
  result := UserName2;
end;

procedure T_IM.TimerNombresClientsActuelsTimer(Sender: TObject);
begin
  application.ProcessMessages;
end;

//Час перез'єднання

procedure T_IM.TimerReconnexionTimer(Sender: TObject);
var
  IPServeur: string;
begin
  if not ClientConnecter then
  begin
    if (EditIPConnexion <> '') and (EditPortClient <> '') and
      ((strtoint(EditPortClient)) > 0) then
    begin
```

```
      IPServeur := EditIPConnexion;
      ClientSocket.Host := IPServeur;
      ClientSocket.Port := strtoint(EditPortClient);
      ClientSocket.Active := TRUE;
      ClientConnecter := True;
    end;
  end
end;

//Закриття форми

 procedure T_IM.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  ActionFermerExecute(Sender);
end;

//З'єднання з сокетом клієнта

 procedure T_IM.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
var
  MessageInitial: string;
begin
  Application.ProcessMessages;
  ClientConnecter := TRUE;
  MessageInitial := 'µ' + EditLogin + 'µ' + NomPcActuel + '«/\»' +
inttostr(NombreSecret) + 'ч';
  ClientSocket.Socket.SendText(#13 + MessageInitial);
  Application.ProcessMessages;
end;

//Роз'єднання з сокетом клієнта

 procedure T_IM.ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Application.ProcessMessages;
end;

//Помилка у сокеті клієнта

 procedure T_IM.ClientSocketError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  Application.ProcessMessages;
  ErrorCode := 0;
  Application.ProcessMessages;
  ClientSocket.Active := FALSE;
  ClientSocket.Close;
  ClientConnecter := FALSE;
end;

//Первиний аналіз даних

 procedure T_IM.AnalysePremiereInformation(MessageRecu: string);
var
  j, k, NombreIdentifiant: integer;
  login, NomOrdi: string;
  Remplir, LoginExisteDejaDesole: boolean;
begin
  Application.ProcessMessages;

  Login := Gauche('µ', MessageRecu);
  NomOrdi := Droite('µ', MessageRecu);
  NomOrdi := Gauche('«/\»', NomOrdi);
  NombreIdentifiant := StrToInt(Droite('«/\»', MessageRecu));

  Remplir := TRUE;
```

```
  LoginExisteDejaDesole := FALSE;

  for k := 0 to length(StructureOrdinateur) - 1 do
  begin
    if CompareStr(Login, StructureOrdinateur[k].LoginConnecte) = 0 then
    begin
      LoginExisteDejaDesole := TRUE;
    end;
  end;

  for j := 1 to NumeroArriveConnexion + 3 do
  begin
    if (StructureOrdinateur[j].LoginConnecte = '') and
      (StructureOrdinateur[j].NomOrdinateur = '') and
      (Remplir = TRUE) and (LoginExisteDejaDesole = FALSE) then
    begin
      StructureOrdinateur[j].LoginConnecte := login;
      StructureOrdinateur[j].NomOrdinateur := NomOrdi;
      StructureOrdinateur[j].Iden := NombreIdentifiant;
      StructureOrdinateur[j].Img := ComboBox1.Text;
      Remplir := FALSE;
    end;
  end;
end;

procedure T_IM.Action10Execute(Sender: TObject);
begin
  CAudioVideo.Show;
end;

procedure T_IM.Action13Execute(Sender: TObject);
begin
  ChoixCouleur.Show;
end;

procedure T_IM.Action14Execute(Sender: TObject);
var
  IPServeur: string;
begin
  if (EditIPConnexion <> '') and (EditPortClient <> '') and
    ((strtoint(EditPortClient)) > 0) then
  begin
    IPServeur := EditIPConnexion;
    ClientSocket.Host := IPServeur;
    ClientSocket.Port := strtoint(EditPortClient);
    ClientSocket.Active := TRUE;
  end;
end;

procedure T_IM.ActionFermerExecute(Sender: TObject);
begin
  Shell_NotifyIcon(Nim_DELETE, @trayicon);
  Fermer1Click(_IM);
end;

procedure T_IM.ActionAjouterGroupeExecute(Sender: TObject);
begin
  ClientDataSet3.Insert;
  GroupeNB := GroupeNB + 1;
  ClientDataSet3N_ID.Value := GroupeNB;
  ClientDataSet3GROUPE.Value := 'Groupe Temp';
  ClientDataSet3.Post;
end;

procedure T_IM.ActionEMAILExecute(Sender: TObject);
begin
  if FileExists('C:\Program Files\Mozilla Thunderbird\thunderbird.exe') then
    ShellExecute(handle, 'open', 'C:\Program Files\Mozilla
Thunderbird\thunderbird.exe', '', '', 0);
```

```
end;

procedure T_IM.ActionNavigateurExecute(Sender: TObject);
begin
  if FileExists('C:\Program Files\Mozilla Firefox\firefox.exe') then
    ShellExecute(handle, 'open', 'C:\Program Files\Mozilla Firefox\firefox.exe',
'', '', 0)
  else
    ShellExecute(handle, 'open', 'C:\Program Files\Internet
Explorer\iexplorer.exe', '', '', 0);
end;

procedure T_IM.ActionQuitterExecute(Sender: TObject);
begin
  Shell_NotifyIcon(Nim_DELETE, @trayicon);
  Fermer1Click(_IM);
end;

procedure T_IM.AnalyseDerniereInformation(MessageRecu: string);
var
  j, k, l, UnCranDeMoins, Identifiant: integer;
  login, NomOrdi: string;
begin
  Application.ProcessMessages;
  Login := Gauche('µ', MessageRecu);
  MessageRecu := Droite('µ', MessageRecu);
  NomOrdi := Gauche('«/\»', MessageRecu);
  Identifiant := strtoint(Droite('«/\»', MessageRecu));

  for j := 1 to 52 do
  begin
    if (CompareStr(StructureOrdinateur[j].LoginConnecte, Login) = 0)
      and (CompareStr(StructureOrdinateur[j].NomOrdinateur, NomOrdi) = 0)
      and (Identifiant = StructureOrdinateur[j].Iden) then
    begin
      StructureOrdinateur[j].LoginConnecte := '';
      StructureOrdinateur[j].NomOrdinateur := '';
      StructureOrdinateur[j].Iden := 0;
      StructureOrdinateur[j].Img := ComboBox1.Text;

      fillchar(StructureOrdinateur002, sizeof(StructureOrdinateur002), 0); //
Розмір структури дорівнює нулю
      UnCranDeMoins := 0;

      for k := 1 to 52 do
      begin
        if (StructureOrdinateur[k].LoginConnecte = '')
          and (StructureOrdinateur[k].NomOrdinateur = '')
          and (StructureOrdinateur[k].Iden = 0) then
        begin
          inc(UnCranDeMoins);
        end
        else
        begin
          StructureOrdinateur002[k - UnCranDeMoins].LoginConnecte :=
StructureOrdinateur[k].LoginConnecte;
          StructureOrdinateur002[k - UnCranDeMoins].NomOrdinateur :=
StructureOrdinateur[k].NomOrdinateur;
          StructureOrdinateur002[k - UnCranDeMoins].Iden :=
StructureOrdinateur[k].Iden;
          StructureOrdinateur002[k - UnCranDeMoins].Img :=
StructureOrdinateur[k].Img;
        end;
      end;

      fillchar(StructureOrdinateur, sizeof(StructureOrdinateur), 0);
      for l := 1 to 52 do
      begin
        if (StructureOrdinateur002[l].LoginConnecte <> '')
```

```
          and (StructureOrdinateur002[l].NomOrdinateur <> '')
          and (StructureOrdinateur002[l].Iden <> 0) then
        begin
          StructureOrdinateur[l].LoginConnecte :=
StructureOrdinateur002[l].LoginConnecte;
          StructureOrdinateur[l].NomOrdinateur :=
StructureOrdinateur002[l].NomOrdinateur;
          StructureOrdinateur[l].Iden := StructureOrdinateur002[l].Iden;
          StructureOrdinateur[l].Img := StructureOrdinateur002[l].Img;
        end;
      end;
    end;
  end;
end;

procedure T_IM.ComboBox1Change(Sender: TObject);
var
  j: integer;
  MessageStatut: string;
  Icon: TIcon;
  Image : TBitmap;
begin
  Icon := TIcon.Create;
  Image1.Picture := nil;
  ImageList1.GetIcon(ComboBox1.ItemIndex, Icon);
  ImageList1.GetBitmap(ComboBox1.ItemIndex,Image1.Picture.Bitmap);
  Trayicon.hIcon := Icon.Handle;
  Shell_NotifyIcon(Nim_Modify, @Trayicon);

      for j := 1 to 52 do
      begin
        if StructureOrdinateur[j].LoginConnecte = UserName then
          StructureOrdinateur[j].img := ComboBox1.Text;
      end;

      if ClientConnecter = TRUE then
      begin
        MessageStatut := '#СТАТУС#' + ComboBox1.Text + '#DE#' + EditLogin +
'#ЗАВЕРШЕННЯ РОБОТИ#';
        _IM.ClientSocket.Socket.SendText(#13 + MessageStatut);
      end;
end;

procedure T_IM.cxGrid3DBTableView1CellDblClick(
  Sender: TcxCustomGridTableView; ACellViewInfo: TcxGridTableDataCellViewInfo;
  AButton: TMouseButton; AShift: TShiftState; var AHandled: Boolean);
var
  Login: string;
begin
  if
((ACellViewInfo.GridRecord.Values[TcxGridDBTableView(Sender).GetColumnByFieldNam
e('NOMPRENOM').Index] <> '')) then
  begin
    Login :=
ACellViewInfo.GridRecord.Values[TcxGridDBTableView(Sender).GetColumnByFieldName(
'NOMPRENOM').Index];
    if CompareStr(Login, EditLogin) <> 0 then
    begin
      if assigned(MsgPerso) then
      begin
        MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' + Login +
' :';
        MsgPerso.Login.Caption :=
ACellViewInfo.GridRecord.Values[TcxGridDBTableView(Sender).GetColumnByFieldName(
'ORDI').Index];
        MsgPerso.LabeledEdit1.clear;
        MsgPerso.Show;
      end
      else
```

```
      begin
        MsgPerso := TMsgPerso.Create(_IM);
        MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' + Login +
' :';
        MsgPerso.Login.Caption :=
ACellViewInfo.GridRecord.Values[TcxGridDBTableView(Sender).GetColumnByFieldName(
'ORDI').Index];
        MsgPerso.cxGroupBox2.Caption := '  A : ' + Login + '  ';
        MsgPerso.Show;
      end;
    end
    else
    begin
      MessageDlg('Неможливо відправити повідомлення!', mtInformation, [mbOK],
0);
    end;
  end;
end;

function T_IM.MessageInfo: string;
var
  i: integer;
  MessageInfoListe_ICQ_OSCAR_eur, MessageInfoListe_ICQ_OSCAR_eur1: string;
begin
  Application.ProcessMessages;
  MessageInfoListe_ICQ_OSCAR_eur1 := '';
  MessageInfoListe_ICQ_OSCAR_eur := '';
  MessageInfoListe_ICQ_OSCAR_eur := '@щ*@';
  for i := 1 to NumeroArriveConnexion + 3 do
  begin
    if StructureOrdinateur[i].LoginConnecte <> '' then
    begin
      MessageInfoListe_ICQ_OSCAR_eur1 := MessageInfoListe_ICQ_OSCAR_eur1 + 'µ' +
StructureOrdinateur[i].LoginConnecte
        + '#IMG#' + StructureOrdinateur[i].img + '#ORDI#' +
StructureOrdinateur[i].NomOrdinateur;
    end;
  end;

  if MessageInfoListe_ICQ_OSCAR_eur1 <> '' then
  begin
    MessageInfoListe_ICQ_OSCAR_eur := MessageInfoListe_ICQ_OSCAR_eur +
MessageInfoListe_ICQ_OSCAR_eur1 + 'µ¤@*!@¤';
    result := MessageInfoListe_ICQ_OSCAR_eur;
  end
  else
  begin
    result := '??';
  end;
end;

//Читання з сокету клієнта

 procedure T_IM.ClientSocketRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  TEMPO: string;
begin
  Application.ProcessMessages;
  TEMPO := socket.ReceiveText;

  while (pos(#13, TEMPO) <> 0) do
  begin
    AnalyseMessageRecuParClient(gauche(#13, TEMPO));
    TEMPO := droite(#13, TEMPO);
  end;
  AnalyseMessageRecuParClient(TEMPO);
end;
```

```
procedure T_IM.AnalyseMessageRecuParClient(Msg: string);
var
  MessageRecuClient, MsgTemp, LoginEnvoi, LoginRecoi, MessageTexte: string;
  j, k, l: integer;
  TableauLocal: array[1..52] of integer;
  Personnage, Ordi, Image: string;
  Ajouter, beep: Boolean;
  Registre: TRegistry;



  newlabel : TLabel;
begin
  beep := False;
  Registre := TRegistry.Create;
  Registre.RootKey := HKEY_LOCAL_MACHINE;
  Registre.OpenKey('\Software\IntraMSN\Couleur\', True);
  Registre.OpenKey('\Software\IntraMSN\Son\', True);
  if Registre.ValueExists('Son') then
    if Registre.ReadInteger('Son') = 1 then
      beep := True;
  MessageRecuClient := Msg;
  if (CompareStr(copy(MessageRecuClient, 1, 5), '') = 0) and
(CompareStr(copy(MessageRecuClient, length(MessageRecuClient) - 5, 6), '¤@*!@¤')
= 0) then
  begin
    k := 0;
    fillchar(TableauLocal, sizeof(TableauLocal), 0);

    for j := 1 to length(MessageRecuClient) - 5 do
    begin
      if MessageRecuClient[j] = 'µ' then
      begin
        inc(k);
        TableauLocal[k] := j;
      end;
    end;

    if k <> 0 then
    begin
      for l := 1 to k - 1 do
      begin
        Personnage := copy(MessageRecuClient, TableauLocal[l] + 1,
(TableauLocal[l + 1]) - TableauLocal[l] - 1);
        if (CompareStr(Personnage, '') <> 0) and (comparestr(Personnage, ' ') <>
0) then
        begin
          Image := Gauche('#ORDI#', Droite('#IMG#', Personnage));
          Ordi := Droite('#ORDI#', Personnage);
          Personnage := Gauche('#IMG#', Personnage);

          Ajouter := TRUE;
          DS_SOURCE.DataSet.First;
          while not (DS_SOURCE.DataSet.Eof) do
          begin
            if comparestr(ClientDataSet1NOMPRENOM.Value, Personnage) = 0 then
            begin
              Ajouter := FALSE;
            end;
            DS_SOURCE.DataSet.Next;
          end;
          DS_SOURCE.DataSet.First;
          if Ajouter then
          begin
            DS_SOURCE.DataSet.Insert;
            nbpersonne := nbpersonne + 1;
            ClientDataSet1N_ID.Value := nbpersonne;
```

```
            if Image = 'Зайнятий' then
ClientDataSet1IMAGE.Assign(IMG_PAUSE.Picture.Bitmap);
            if Image = 'Можу розмовляти' then
ClientDataSet1IMAGE.Assign(IMG_LIGNE.Picture.Bitmap);
            if Image = 'Не турбувати' then
ClientDataSet1IMAGE.Assign(IMG_OCCUPE.Picture.Bitmap);
            if Image = 'Розмовляю по телефону' then
ClientDataSet1IMAGE.Assign(IMG_TEL.Picture.Bitmap);
            if Image = 'Тимчасово відсутній' then
ClientDataSet1IMAGE.Assign(IMG_1MN.Picture.Bitmap);
            if Image = 'Зайнятий' then
ClientDataSet1IMAGE.Assign(IMG_ABS.Picture.Bitmap);
            ClientDataSet1NOMPRENOM.Value := Personnage;
            ClientDataSet1ORDI.Value := Ordi;
            ClientDataSet1N_GROUPE.Value := 1;
            DS_SOURCE.DataSet.Post;
          end;
        end;
      end;
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 3), ' ') = 0 then
  begin
    cxGrid3DBTableView1.ClearItems;
    if length(MessageRecuClient) > 5 then
    begin
      MessageRecuClient := Droite(' ', MessageRecuClient);
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 5), ' ') = 0 then
  begin
    if CompareStr(copy(MessageRecuClient, length(MessageRecuClient) - 4, 5), '
') = 0 then
    begin
      MessageRecuClient := Gauche(' ', MessageRecuClient);
      MessageRecuClient := Droite(' ', MessageRecuClient);
      if beep then
        MessageBeep(MB_OK);
      RendreVisiblePremierPlan;

      if CompareStr(copy(MessageRecuClient, 1, 4), ' ') = 0 then
      begin
        MessageRecuClient := Droite(' ', MessageRecuClient);
        MsgTemp := Droite('>>', Gauche('з'єднання', MessageRecuClient));
        MsgTemp := MsgTemp + #13 + 'з'єднання';
        AlertMsgBox('З'єднання', MsgTemp, 0, false, 1000, 10, nil);
        Memo_ICQ_OSCAR_.Lines.Add(MessageRecuClient + '   ' + QuelHeureEstIl);
      end
      else
      begin
        AlertMsgBox('Роз'єднання', MessageRecuClient, 0, false, 1000, 10, nil);
        Memo_ICQ_OSCAR_.Lines.Add(MessageRecuClient);
      end;
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 8), '#СТАТУС#') = 0 then
  begin
    MessageRecuClient := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#', MessageRecuClient);
    MessageRecuClient := Droite('#СТАТУС#', MessageRecuClient);
    LoginEnvoi := Droite('#DE#', MessageRecuClient);
    MessageRecuClient := Gauche('#DE#', MessageRecuClient);
    if beep then
      MessageBeep(MB_OK);
    RendreVisiblePremierPlan;

    DS_SOURCE.DataSet.First;
```

```
    while not DS_SOURCE.DataSet.Eof do
    begin
      if (ClientDataSet1NOMPRENOM.Value = LoginEnvoi) then
      begin
        ClientDataSet1.Edit;
        if MessageRecuClient = 'Зайнятий' then
ClientDataSet1IMAGE.Assign(IMG_PAUSE.Picture.Bitmap);
        if MessageRecuClient = 'Можу розмовляти' then
ClientDataSet1IMAGE.Assign(IMG_LIGNE.Picture.Bitmap);
        if MessageRecuClient = 'Не турбувати' then
ClientDataSet1IMAGE.Assign(IMG_OCCUPE.Picture.Bitmap);
        if MessageRecuClient = 'Розмовляю по телефону' then
ClientDataSet1IMAGE.Assign(IMG_TEL.Picture.Bitmap);
        if MessageRecuClient = 'Тимчасово відсутній' then
ClientDataSet1IMAGE.Assign(IMG_1MN.Picture.Bitmap);
        if MessageRecuClient = 'Зайнятий' then
ClientDataSet1IMAGE.Assign(IMG_ABS.Picture.Bitmap);
        ClientDataSet1.Post;
      end;
      DS_SOURCE.DataSet.Next;
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 8), 'MsgPrive') = 0 then
  begin
    MessageRecuClient := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#', MessageRecuClient);
    MessageRecuClient := Droite('MsgPrive#DE#', MessageRecuClient);
    LoginEnvoi := Gauche('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#', MessageRecuClient);
    MessageRecuClient := Droite('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#', MessageRecuClient);
    LoginRecoi := Gauche('#ТІЛО ПОВІДОМЛЕННЯ#', MessageRecuClient);

    if (Comparestr(LoginEnvoi, EditLogin) = 0) or (Comparestr(LoginRecoi,
EditLogin) = 0) then
    begin
      MessageTexte := Droite('#ТІЛО ПОВІДОМЛЕННЯ#', MessageRecuClient);
      if beep then
        MessageBeep(MB_OK);
      RendreVisiblePremierPlan;
      if (Memo_ICQ_OSCAR_.Lines.Count > 10) then
        Memo_ICQ_OSCAR_.Lines.Clear;

      if (LoginEnvoi <> UserName) then
      begin
        if assigned(MsgPerso) then
        begin

          if (MsgPerso.Caption <> 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :') then
          begin
            MsgPerso := TMsgPerso.Create(_IM);
            MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :';
            MsgPerso.cxGroupBox2.Caption := '  A : ' + LoginEnvoi + '  ';
            MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' dit : ' + MessageTexte);
            MsgPerso.Show;
            FlashWindow(Application.Handle, true);
          end
          else
          begin
            MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' dit : ' + MessageTexte);
            FlashWindow(Application.Handle, true);
          end;
        end
        else
        begin
          MsgPerso := TMsgPerso.Create(_IM);
          MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :';
          MsgPerso.cxGroupBox2.Caption := '  A : ' + LoginEnvoi + '  ';
```

```
          MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' dit : ' + MessageTexte);
          MsgPerso.Show;
          FlashWindow(Application.Handle, true);
        end;
      end
      else
        MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' dit : ' + MessageTexte);
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 12), 'MsgTransfert') = 0 then
  begin
    MessageRecuClient := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#', MessageRecuClient);
    MessageRecuClient := Droite('MsgTransfert#DE#', MessageRecuClient);
    LoginEnvoi := Gauche('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#', MessageRecuClient);
    MessageRecuClient := Droite('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#', MessageRecuClient);
    LoginRecoi := Gauche('#ТІЛО ПОВІДОМЛЕННЯ#', MessageRecuClient);

    if (Comparestr(LoginEnvoi, EditLogin) = 0) or (Comparestr(LoginRecoi,
EditLogin) = 0) then
    begin
      MessageTexte := Droite('#ТІЛО ПОВІДОМЛЕННЯ#', MessageRecuClient);
      if beep then
        MessageBeep(MB_OK);
      RendreVisiblePremierPlan;
      if (Memo_ICQ_OSCAR_.Lines.Count > 10) then
        Memo_ICQ_OSCAR_.Lines.Clear;

      if (LoginEnvoi <> UserName) then
      begin
        if assigned(MsgPerso) then
        begin
          if (MsgPerso.Caption <> 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :') then
          begin
            MsgPerso := TMsgPerso.Create(_IM);
            MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :';
            MsgPerso.cxGroupBox2.Caption := '  A : ' + LoginEnvoi + '  ';
            MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' Ви відправити файл: ' +
MessageTexte);
            ////////////////

 newlabel := Tlabel.Create(MsgPerso.Memo1);
 with newlabel do
 begin
 Parent := MsgPerso.Memo1;
 Caption := 'testtt';
 end;

            MsgPerso.Memo1.Lines.InsertObject(0, 'Test', SpeedButton3);
            MsgPerso.Show;
            FlashWindow(Application.Handle, true);
          end
          else
          begin
            MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' Ви відправити файл: ' +
MessageTexte);
            FlashWindow(Application.Handle, true);
          end;
        end
        else
        begin
          MsgPerso := TMsgPerso.Create(_IM);
          MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :';
          MsgPerso.cxGroupBox2.Caption := '  A : ' + LoginEnvoi + '  ';
          MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' dit : ' + MessageTexte);
          MsgPerso.Show;
```

```
          FlashWindow(Application.Handle, true);
        end;
      end
      else
        MsgPerso.Memo1.Lines.Add('Ви пропонуєте передачу файлів');
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 9), 'MsgBouger') = 0 then
  begin
    MessageRecuClient := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#', MessageRecuClient);
    MessageRecuClient := Droite('MsgBouger#DE#', MessageRecuClient);
    LoginEnvoi := Gauche('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#', MessageRecuClient);
    MessageRecuClient := Droite('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#', MessageRecuClient);
    LoginRecoi := Gauche('#ТІЛО ПОВІДОМЛЕННЯ#', MessageRecuClient);

    if (Comparestr(LoginEnvoi, EditLogin) = 0) or (Comparestr(LoginRecoi,
EditLogin) = 0) then
    begin
      MessageTexte := Droite('#ТІЛО ПОВІДОМЛЕННЯ#', MessageRecuClient);
      if beep then
        MessageBeep(MB_OK);
      RendreVisiblePremierPlan;
      if (Memo_ICQ_OSCAR_.Lines.Count > 10) then
        Memo_ICQ_OSCAR_.Lines.Clear;

      if (LoginEnvoi <> UserName) then
      begin
        if assigned(MsgPerso) then
        begin
          if (MsgPerso.Caption <> 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :') then
          begin
            MsgPerso := TMsgPerso.Create(_IM);
            MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :';
            MsgPerso.cxGroupBox2.Caption := '  A : ' + LoginEnvoi + '  ';
            MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' ' + MessageTexte);
            MsgPerso.Show;
            Bouge := True;
            FlashWindow(Application.Handle, true);
          end
          else
          begin
            MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' ' + MessageTexte);
            Bouge := True;
            FlashWindow(Application.Handle, true);
          end;
        end
        else
        begin
          MsgPerso := TMsgPerso.Create(_IM);
          MsgPerso.Caption := 'Хочете відправити особисте повідомлення ' +
LoginEnvoi + ' :';
          MsgPerso.cxGroupBox2.Caption := '  A : ' + LoginEnvoi + '  ';
          MsgPerso.Memo1.Lines.Add(LoginEnvoi + ' ' + MessageTexte);
          MsgPerso.Show;
          Bouge := True;
          FlashWindow(Application.Handle, true);
        end;
      end
      else
        MsgPerso.Memo1.Lines.Add('Доступний тільки для тих, хто повинен
бачити!');
    end;
  end;

  if CompareStr(copy(MessageRecuClient, 1, 5), 'Підключено') = 0 then
  begin
```

```
    Memo_ICQ_OSCAR_.Lines.Add('Сервер роз'єднаний...  ' + QuelHeureEstIl);
    Application.ProcessMessages;
    cxGrid3DBTableView1.ClearItems;

    ClientSocket.Active := FALSE;
    ClientSocket.Close;
    ClientConnecter := FALSE;
  end;
end;

procedure T_IM.PourquoiDeconnecte(raison: string; iden: integer);
begin
  case StrToInt(raison) of
    001: begin
        MessageDlg('Введіть логін та пароль.', mtInformation, [mbOK], 0);
      end;
    003: begin
        MessageDlg(«Сервер зайнятий. Будь ласка, повторіть спробу пізніше.',
mtInformation, [mbOK], 0);
      end;
  end;

  Application.ProcessMessages;

  cxGrid3DBTableView1.ClearItems;

  ClientSocket.Active := FALSE;
  ClientSocket.Close;
  ClientConnecter := FALSE;
end;

{Зміна вкладки}

procedure T_IM.LabeledEditMessageEcritKeyPress(Sender: TObject;
  var Key: Char);
var
  DroitEcrire: boolean;
  MessageAEnvoyer: string;
begin
  if (Key = #13)
    and (ClientConnecter = TRUE) then
  begin
    key := #0;
    DroitEcrire := FALSE;
    DS_SOURCE.DataSet.First;

    while not DS_SOURCE.DataSet.Eof do
    begin
      if CompareStr(EditLogin, ClientDataSet1NOMPRENOM.Value) = 0 then
      begin
        DroitEcrire := TRUE;
      end;
      DS_SOURCE.DataSet.Next;
    end;
    if DroitEcrire then
    begin
      MessageAEnvoyer := ' ' + EditLogin + '  ';
      ClientSocket.Socket.SendText(#13 + MessageAEnvoyer);
    end;
  end;
  if (Key = #13) then
    key := #0;
end;

procedure T_IM.Fermer1Click(Sender: TObject);
begin
  ClipCursor(nil);
  if (not ClientConnecter) then
    Application.Terminate;
```

```
  if ClientConnecter = TRUE then
  begin
    DireQueOnSeDeconnecte;
    ClientSocket.Active := FALSE;
    sleep(1000);
    Application.Terminate;
  end;

end;

procedure T_IM.DireQueOnSeDeconnecte;
var
  MessageFinal: string;
begin
  Application.ProcessMessages;

  cxGrid3DBTableView1.ClearItems;

  if ClientConnecter then
  begin
    MessageFinal := '@DECO' + EditLogin + 'µ' + NomPcActuel + '«/\»' +
inttostr(NombreSecret) + 'K';
    ClientSocket.Socket.SendText(#13 + MessageFinal);
  end;

  ClientSocket.Active := FALSE;
  ClientSocket.Close;
  ClientConnecter := FALSE;

end;

procedure T_IM.EnLigne1Click(Sender: TObject);
var
  j: integer;
  MessageStatut: string;
  Icon: TIcon;
begin
  MessageStatut := (Sender as TMenuItem).Caption;
  Icon := TIcon.Create;
  DS_SOURCE.dataset.First;
  while not DS_SOURCE.DataSet.Eof do
  begin
    if UserName = ClientDataSet1NOMPRENOM.Value then
    begin
      ClientDataSet1.Edit;
      if MessageStatut = '&У мережі' then
      begin
        ImageList1.GetIcon(0, Icon);
        Image1.Picture.Bitmap := IMG_LIGNE.Picture.Bitmap;
        ClientDataSet1IMAGE.Assign(IMG_LIGNE.Picture.Bitmap);
      end;
      if MessageStatut = '&Зайнятий' then
      begin
        ImageList1.GetIcon(1, Icon);
        Image1.Picture.Bitmap := IMG_OCCUPE.Picture.Bitmap;
        ClientDataSet1IMAGE.Assign(IMG_OCCUPE.Picture.Bitmap);
      end;
      if MessageStatut = ''&Розмовляю по телефону'' then
      begin
        ImageList1.GetIcon(2, Icon);
        Image1.Picture.Bitmap := IMG_TEL.Picture.Bitmap;
        ClientDataSet1IMAGE.Assign(IMG_TEL.Picture.Bitmap);
      end;
      if MessageStatut = 'На перерві' then
      begin
        ImageList1.GetIcon(3, Icon);
        Image1.Picture.Bitmap := IMG_PAUSE.Picture.Bitmap;
        ClientDataSet1IMAGE.Assign(IMG_PAUSE.Picture.Bitmap);
```

```
      end;
      if MessageStatut = 'Тимчасово відсутній' then
      begin
        ImageList1.GetIcon(4, Icon);
        Image1.Picture.Bitmap := IMG_1MN.Picture.Bitmap;
        ClientDataSet1IMAGE.Assign(IMG_1MN.Picture.Bitmap);
      end;
      if MessageStatut = 'Відсутній' then
      begin
        ImageList1.GetIcon(5, Icon);
        Image1.Picture.Bitmap := IMG_ABS.Picture.Bitmap;
        ClientDataSet1IMAGE.Assign(IMG_ABS.Picture.Bitmap);
      end;
      ClientDataSet1.Post;
      Trayicon.hIcon := Icon.Handle;
      Shell_NotifyIcon(Nim_Modify, @Trayicon);

      for j := 1 to 52 do
      begin
        if StructureOrdinateur[j].LoginConnecte = UserName then
          StructureOrdinateur[j].img := ComboBox1.Text;
      end;

      if ClientConnecter = TRUE then
      begin
        MessageStatut := '#СТАТУС#' + ComboBox1.Text + '#DE#' + EditLogin +
'#ЗАВЕРШЕННЯ РОБОТИ#';
        _IM.ClientSocket.Socket.SendText(#13 + MessageStatut);
      end;
    end;
    DS_SOURCE.DataSet.Next;
  end;

end;

procedure T_IM.BitBtnDeconnexionClick(Sender: TObject);
begin
  DireQueOnSeDeconnecte;
end;

procedure T_IM.FormResize(Sender: TObject);
begin
  Application.OnMinimize := Minimize;
end;



initialization
  NumeroArriveConnexion := 0;

end.
```

Серверна частина
Unit_IM.pas – основна частина серверного додатку

```
unit Unit_IM;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls,WinSock, Buttons, ScktComp, ExtCtrls, Grids,
  ValEdit, Menus, Math,ShellAPI;

const
    WM_CAllBack = WM_USER;

type
  T_IM = class(TForm)
    PageControl: TPageControl;
    TabSheetServeur: TTabSheet;
    GroupBox1: TGroupBox;
    LabelIP: TLabel;
    ServerSocket: TServerSocket;
    TimerInformations_ICQ_OSCAR_eur: TTimer;
    BitBtnLancerS: TBitBtn;
    TimerNombresClientsActuels: TTimer;


    procedure TabSheetServeurShow(Sender: TObject);
    function TrouverIP(ordinateur : string) : string;
    function NomPcActuel : string;
    function MessageInfo : string;
    procedure FormCreate(Sender: TObject);
    procedure ServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure BitBtnLancerSClick(Sender: TObject);
    procedure ServerSocketClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure AnalysePremiereInformation(MessageRecu : string);
    procedure AnalyseDerniereInformation(MessageRecu : string);
    procedure TimerInformations_ICQ_OSCAR_eurTimer(Sender: TObject);
    procedure NomClientParti;
    procedure PageControlChanging(Sender: TObject;
      var AllowChange: Boolean);

    procedure LabeledEdit1KeyPress(Sender: TObject; var Key: Char);
    procedure Fermer1Click(Sender: TObject);
    procedure PasLeDroitDeSeConnecter(Login : string; Ordi : string; iden :
integer);
    procedure VientDeSeDeconnecterVolontairement(MessageRecu : string);
    procedure VientDeSeDeconnecterCarVire(MessageRecu : string);
    procedure TropDeMondeSurLeServeur(MessageRecu : string);
    procedure ReponseDeMessageStatut(MessageRecu: string; MessageEnvoi: string);
    procedure ReponseDeMessagePrive(MessageRecu : string; MessageEnvoi :
string);
    procedure AnalyseMessageRecuParServeur(Msg : String);
    procedure ServerSocketClientError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);


  private
  public
  protected
```

```
        IsServer: Boolean;
  end;

type
    TStructureListeConnecte = record
         LoginConnecte : string[30];
         NomOrdinateur : string[30];
         Iden : integer;
         img : string;
    end;

var
  _IM: T_IM;
  EditPortServeur : string;
  StructureOrdinateur,StructureOrdinateur002 :array[1..52] of
TStructureListeConnecte;

ComboBoxNombresClients,NumeroArriveConnexion,NombresMaximumClients,NombreSecret
: integer;
  ServeurActif,ClientConnecter : boolean;
  Present: TDateTime;
  Hour, Min, Sec,MSec: Word;

_ICQ_OSCAR_HauteurDeDebut,_ICQ_OSCAR_LargeurDeDebut,PageHauteurDebut,PageLargeur
Debut : integer;
  TrayIcon : TNotifyIconData;
  blah : HICON;
  mvt : Integer;

function UserName():string;


implementation

{$R *.dfm}

//функція визначення часу

function QuelHeureEstIl : string;
begin
   Present:= Now;
   DecodeTime(Present, Hour, Min, Sec,MSec);
   result := '['+IntToStr(Hour)+':'+IntToStr(Min)+':'+IntToStr(Sec)+']';
end;
function droite(substr: string; s: string): string;
begin
  if pos(substr,s)=0 then result:='' else
    result:=copy(s, pos(substr, s)+length(substr), length(s)-pos(substr,
s)+length(substr));
end;
function gauche(substr: string; s: string): string;
begin
  result:=copy(s, 1, pos(substr, s)-1);
end;

//Функція пошуку IP-адреси

function T_IM.TrouverIP(ordinateur : string) : string;
var
   WSAData : TWSAData;
   Name,Address : String;
   Phe : PHostEnt;
begin
    WSAStartup(2,WSAData);
    SetLength(Name,255);
    Phe := GetHostByName(PChar(ordinateur));
    with Phe^ do
    Address := Format ('%d.%d.%d.%d' , [Byte(h_addr^[0]),Byte(h_addr^[1]),
                                        Byte(h_addr^[2]),Byte(h_addr^[3])]);
```

```
    WSACleanup;
    TrouverIP := Address;
end;

function T_IM.NomPcActuel : string;
var
  Buffer : array[0..255] of char;
  BufferSize : DWORD;
begin
  BufferSize := sizeOf(Buffer);
  GetComputerName(@buffer, BufferSize);
  result := buffer;
end;

//процедура визначення списку серверів

procedure T_IM.TabSheetServeurShow(Sender: TObject);
begin
        LabelIP.Caption := TrouverIP(NomPcActuel);
        GroupBox1.Caption := 'Вибрати сервер: '+ NomPcActuel;
        ComboBoxNombresClients := 999;
end;

//Створення форми чату


//створення форми

 procedure T_IM.FormCreate(Sender: TObject);
var
   SysMenu: hMenu;
begin
  SysMenu := GetSystemMenu(Handle, False);
  ModifyMenu(SysMenu, sc_Close, mf_ByCommand, sc_Close, 'Покинути програмне
забезпечення обміну інформацією у мережі  під керуванням ОС
Windows!!!'#9'Alt+F4');
  SysMenu := GetSystemMenu(application.handle,false);
  ModifyMenu(SysMenu, sc_Close, mf_ByCommand, sc_Close, 'Покинути програмне
забезпечення обміну інформацією у мережі  під керуванням ОС
Windows!!!'#9'Alt+F4');



   Randomize;
   NombreSecret := RandomRange(1000,9999);

  _IM.Left := screen.Width - _IM.Width ;
  _IM.top := screen.height - _IM.height-30 ;

  blah := application.Icon.Handle;
  Trayicon.cbSize := SizeOf(TNotifyIconData);
  Trayicon.Wnd := handle;
  Trayicon.szTip := 'Обмін повідомленями';
  Trayicon.uID := 1;
  TrayIcon.hIcon := blah;
  TrayIcon.uCallbackMessage := WM_CAllBack;
  Trayicon.uFlags := NIF_MESSAGE or NIF_ICON or NIF_TIP;
  Shell_NotifyIcon(NIM_ADD,@trayicon);

  _IM.Visible := true;

  _IM.Paint;

  EditPortServeur := '2879';

end;

//Введення імені користувача
```

```
function UserName():string;
const
  cnMaxUserNameLen = 254;
var
 UserName2 : string;
 nSize : DWord;
 begin
 nSize := cnMaxUserNameLen - 1;
 SetLength(UserName2, cnMaxUserNameLen);

 GetUserName(Pchar(UserName2), nSize);

 SetLength(UserName2, nSize -1);

 result := UserName2;
 end;

//Прийняття даних з сокету серверу

procedure T_IM.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  IsServer := True;
end;

//З'єднання сокетів сервера та клієнта програмного забезпечення обміну
інформацією у мережі  під керуванням ОС Windows

procedure T_IM.ServerSocketClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
     inc(NumeroArriveConnexion);
     TimerInformations_ICQ_OSCAR_eur.Enabled := TRUE;
end;

//Закриття форми


//Закриття форми

 procedure T_IM.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   Fermer1Click(_IM);
end;

//Запуск та відключення серверу

procedure T_IM.BitBtnLancerSClick(Sender: TObject);
var
  i : integer;
begin
   if  (CompareStr('Запуск серверу',BitBtnLancerS.Caption)=0)  then
   begin
      if (EditPortServeur <> '') and ((strtoint(EditPortServeur)) > 0) then
      begin
      try
        NombresMaximumClients := ComboBoxNombresClients;
        ServerSocket.Port := strtoint(EditPortServeur);
        ServerSocket.Active := True;
        BitBtnLancerS.Caption := 'Відключення серверу';
        ServeurActif := TRUE;
        NombresMaximumClients := ComboBoxNombresClients;
      except on ESocketError do
      begin
         MessageDlg('Ви не можете запустити 2 сервери на одному комп'ютері.',
mtInformation, [mbOK], 0);
         BitBtnLancerS.Caption := 'Запуск серверу';
          TimerInformations_ICQ_OSCAR_eur.Enabled := FALSE;
          TimerInformations_ICQ_OSCAR_eur.Interval := 3000;
```

```
          fillchar(StructureOrdinateur,sizeof(StructureOrdinateur),0);
          fillchar(StructureOrdinateur002,sizeof(StructureOrdinateur002),0);
          NumeroArriveConnexion :=0;
          ServerSocket.Active := FALSE;
          ServeurActif := FALSE;
      end;
      end;
      end;
   end
   else if (CompareStr(BitBtnLancerS.Caption,'Відключення серверу') =0) then
        begin
          for i:=0 to NumeroArriveConnexion-1 do
          begin
             ServerSocket.Socket.Connections[i].SendText(#13+'Підключено');  //
відправлення даних клієнту
          end;
          BitBtnLancerS.Caption := 'Запуск серверу';
          TimerInformations_ICQ_OSCAR_eur.Enabled := FALSE;
          TimerInformations_ICQ_OSCAR_eur.Interval := 3000;
          fillchar(StructureOrdinateur,sizeof(StructureOrdinateur),0);
          fillchar(StructureOrdinateur002,sizeof(StructureOrdinateur002),0);
          NumeroArriveConnexion :=0;
          ServerSocket.Active := FALSE;
          ServeurActif := FALSE;
        end;

end;

// Роз'єднання сокетів сервера та клієнта програмного забезпечення обміну
інформацією у мережі  під керуванням ОС Windows

procedure T_IM.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
      dec(NumeroArriveConnexion);
      if NumeroArriveConnexion = 0 then
      begin
        TimerInformations_ICQ_OSCAR_eur.Enabled := FALSE;
        TimerInformations_ICQ_OSCAR_eur.Interval := 3000;
      end;
end;

// Помилки з'єднання сокетів сервера та клієнта програмного забезпечення обміну
інформацією у мережі  під керуванням ОС Windows

procedure T_IM.ServerSocketClientError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent; var ErrorCode: Integer);
var Rapport: string;
begin
  case ErrorEvent of
    eeGeneral: Rapport := 'Несподівана помилка' + Socket.RemoteAddress;
    eeSend: Rapport := 'Помилка з'єднання сокетів' + Socket.RemoteAddress;
    eeReceive: Rapport := 'Помилка читання з'єднання сокетів' +
Socket.RemoteAddress;
    eeConnect: Rapport := 'Запит на з'єднання неможливий ' +
Socket.RemoteAddress;
    eeDisconnect: Rapport := 'Помилка закриття  з'єднання ' +
Socket.RemoteAddress;
    eeAccept: Rapport := 'Помилка прийняття запиту з'єднання клієнта ' +
Socket.RemoteAddress;
  else
  end;
  ErrorCode := 0;
end;

// Читання з'єднання сокетів сервера та клієнта програмного забезпечення обміну
інформацією у мережі  під керуванням ОС Windows

procedure T_IM.ServerSocketClientRead(Sender: TObject;Socket: TCustomWinSocket);
```

```
var
    TEMPO : string;
begin
  Application.ProcessMessages;
  TEMPO := socket.ReceiveText;

  while (pos(#13,TEMPO) <> 0) do
  begin
     AnalyseMessageRecuParServeur(gauche(#13,TEMPO));
     TEMPO := droite(#13,TEMPO);
  end;
  AnalyseMessageRecuParServeur(TEMPO);
end;

//Аналіз повідомлення отриманого сервером

procedure T_IM.AnalyseMessageRecuParServeur(Msg : String);
var
    MessageRecuServeur, MessagePrivePourLesClient: string;
begin
     MessageRecuServeur := Msg;
     if copy(MessageRecuServeur,1,1) = 'µ' then
     begin
          if NombresMaximumClients+1 > ServerSocket.Socket.ActiveConnections
then
          begin
            MessageRecuServeur := Gauche('ч',MessageRecuServeur);

AnalysePremiereInformation(copy(MessageRecuServeur,2,length(MessageRecuServeur)-
1));
          end
          else
          begin
              MessageRecuServeur := Gauche('ч',MessageRecuServeur);

TropDeMondeSurLeServeur(copy(MessageRecuServeur,2,length(MessageRecuServeur)-
1));
          end;
     end;
     if Comparestr(copy(MessageRecuServeur,1,5),'@DECO') = 0 then
     begin
              MessageRecuServeur := Gauche('К',MessageRecuServeur);
              MessageRecuServeur := Droite('@DECO',MessageRecuServeur);
              AnalyseDerniereInformation(MessageRecuServeur);
              NomClientParti;
              VientDeSeDeconnecterVolontairement(MessageRecuServeur);
     end;
     if Comparestr(copy(MessageRecuServeur, 1, 8), '#СТАТУС#') = 0 then
  begin
    MessagePrivePourLesClient := MessageRecuServeur;
    MessageRecuServeur := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#', MessageRecuServeur);
    MessageRecuServeur := Droite('#СТАТУС#', MessageRecuServeur);
    ReponseDeMessageSTATUT(MessageRecuServeur, MessagePrivePourLesClient);
  end;

     if Comparestr(copy(MessageRecuServeur,1,8),'MsgPrive')=0 then
     begin
         MessagePrivePourLesClient := MessageRecuServeur;
         MessageRecuServeur := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#',MessageRecuServeur);
         MessageRecuServeur := Droite('MsgPrive#DE#',MessageRecuServeur);
         ReponseDeMessagePrive(MessageRecuServeur,MessagePrivePourLesClient);
     end;

     if Comparestr(copy(MessageRecuServeur,1,12),'MsgTransfert')=0 then
     begin
         MessagePrivePourLesClient := MessageRecuServeur;
         MessageRecuServeur := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#',MessageRecuServeur);
         MessageRecuServeur := Droite('MsgTransfert#DE#',MessageRecuServeur);
         ReponseDeMessagePrive(MessageRecuServeur,MessagePrivePourLesClient);
```

```
     end;

     if Comparestr(copy(MessageRecuServeur,1,9),'MsgBouger')=0 then
     begin
         MessagePrivePourLesClient := MessageRecuServeur;
         MessageRecuServeur := Gauche('#ЗАВЕРШЕННЯ РОБОТИ#',MessageRecuServeur);
         MessageRecuServeur := Droite('MsgBouger#DE#',MessageRecuServeur);
         ReponseDeMessagePrive(MessageRecuServeur,MessagePrivePourLesClient);
     end;
end;

// Визначення статусу повідомлення

procedure T_IM.ReponseDeMessageStatut(MessageRecu: string; MessageEnvoi:
string);
var
  LoginEnvoi: string;
  i: integer;
begin
  LoginEnvoi := Droite('#DE#', MessageRecu);
  MessageRecu := Gauche('#DE#', MessageRecu);

  for i := 0 to NumeroArriveConnexion - 1 do
  begin
    ServerSocket.Socket.Connections[i].SendText(#13 + MessageEnvoi);
  end;
end;

//Відповісти на особисте повідомлення

procedure T_IM.ReponseDeMessagePrive(MessageRecu : string; MessageEnvoi :
string);
var
   LoginEnvoi, LoginRecoi, MessageTexte : string;
   i : integer;
begin
    LoginEnvoi := Gauche('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#',MessageRecu);
    MessageRecu := Droite('#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#',MessageRecu);
    LoginRecoi := Gauche('#ТІЛО ПОВІДОМЛЕННЯ#',MessageRecu);
    MessageTexte := Droite('#ТІЛО ПОВІДОМЛЕННЯ#',MessageRecu);

    for i:=0 to NumeroArriveConnexion-1 do
    begin
      ServerSocket.Socket.Connections[i].SendText(#13+MessageEnvoi);
    end;

end;
procedure T_IM.TropDeMondeSurLeServeur(MessageRecu : string);
var
   i,iden : integer;
   Login,NomOrdi,MessageInfoDepart : string;
begin
   //EditLogin.Text + 'µ' + NomPcActuel+'«/\»'+inttostr(NombreSecret)
   Application.ProcessMessages;
   Login := Gauche('µ',MessageRecu);
   MessageRecu := Droite('µ',MessageRecu);
   NomOrdi := Gauche('«/\»',MessageRecu);
   iden := strtoint(Droite('«/\»',MessageRecu));

   MessageInfoDepart :=
'TuEsVirй°J°'+Login+'µ'+NomOrdi+'«/\»'+inttostr(iden)+'003';

   for i:=0 to NumeroArriveConnexion-1 do
   begin
      ServerSocket.Socket.Connections[i].SendText(#13+MessageInfoDepart);
   end;
end;

procedure T_IM.VientDeSeDeconnecterVolontairement(MessageRecu : string);
```

```
var
   Login,NomOrdi : string;
   m : integer;
begin
   Application.ProcessMessages;
   Login := Gauche('µ',MessageRecu);
   MessageRecu := Droite('µ',MessageRecu);
   NomOrdi := Gauche('«/\»',MessageRecu);

   for m:=0 to NumeroArriveConnexion-1 do
   begin
       ServerSocket.Socket.Connections[m].SendText(#13+'Сервер'+' >>  '+Login+'
('+NomOrdi+')'+' роз'єднаний...'+'');
   end;
end;

procedure T_IM.VientDeSeDeconnecterCarVire(MessageRecu : string);
var
   m : integer;
   Login,NomOrdi : string;
begin
   //EditLogin.Text + 'µ' + NomPcActuel+'«/\»'+inttostr(NombreSecret)
   Application.ProcessMessages;
   Login := Gauche('µ',MessageRecu);
   MessageRecu := Droite('µ',MessageRecu);
   NomOrdi := Gauche('«/\»',MessageRecu);

   for m:=0 to NumeroArriveConnexion-1 do
   begin
       ServerSocket.Socket.Connections[m].SendText(#13+'Сервер'+' >>  '+Login+'
('+NomOrdi+')'+' з'єднаний...'+'');
   end;
end;

//Первиний аналіз даних

 procedure T_IM.AnalysePremiereInformation(MessageRecu : string);
var
    i,j,k,NombreIdentifiant : integer;
    login,NomOrdi,MessageInfoArrive : string;
    Remplir,LoginExisteDejaDesole : boolean;
begin
     //EditLogin.Text + 'µ' + NomPcActuel + '«/\»' + NombreSecret
     Application.ProcessMessages;

     Login := Gauche('µ',MessageRecu);
     NomOrdi := Droite('µ',MessageRecu);
     NomOrdi := Gauche('«/\»',NomOrdi);
     NombreIdentifiant := StrToInt(Droite('«/\»',MessageRecu));

     Remplir := TRUE;
     LoginExisteDejaDesole := FALSE;

             for k := 0 to length(StructureOrdinateur)-1 do
             begin
                 if CompareStr(Login,StructureOrdinateur[k].LoginConnecte)=0
then
                 begin
                     LoginExisteDejaDesole := TRUE;
                 end;
             end;

             for j := 1 to NumeroArriveConnexion + 3 do
             begin
                if (StructureOrdinateur[j].LoginConnecte = '') and
                   (StructureOrdinateur[j].NomOrdinateur = '') and
                   (Remplir = TRUE) and (LoginExisteDejaDesole = FALSE) then
                begin
                   StructureOrdinateur[j].LoginConnecte := login;
```

```
                  StructureOrdinateur[j].NomOrdinateur :=  NomOrdi;
                  StructureOrdinateur[j].Iden := NombreIdentifiant;
                  StructureOrdinateur[j].img := 'Можу розмовляти';
                  Remplir := FALSE;
               end;
            end;

            if not Remplir then
            begin

              MessageInfoArrive := ''Сервер'+' >>  '+Login+' ('+NomOrdi+')'+'
з'єднаний...'+''';
               for i:=0 to NumeroArriveConnexion-1 do
               begin

ServerSocket.Socket.Connections[i].SendText(#13+MessageInfoArrive);
               end;
            end;

            if LoginExisteDejaDesole then
                  PasLeDroitDeSeConnecter(login,NomOrdi,NombreIdentifiant);
end;

//Помилка з'єднання

Procedure T_IM.PasLeDroitDeSeConnecter(Login : string; Ordi : string; iden :
integer);
var
   MessageInfoDepart : string;
   i : integer;
begin
   MessageInfoDepart :=
'TuEsVirй°J°'+Login+'µ'+Ordi+'«/\»'+inttostr(iden)+'001';

   for i:=0 to NumeroArriveConnexion-1 do
   begin
      ServerSocket.Socket.Connections[i].SendText(#13+MessageInfoDepart);
   end;
end;

// Останній аналіз інформації

procedure T_IM.AnalyseDerniereInformation(MessageRecu : string);
var
    j,k,l,UnCranDeMoins,Identifiant : integer;
    login,NomOrdi : string;
begin
     Application.ProcessMessages;
     Login := Gauche('µ',MessageRecu);
     MessageRecu := Droite('µ',MessageRecu);
     NomOrdi := Gauche('«/\»',MessageRecu);
     Identifiant := strtoint(Droite('«/\»',MessageRecu));

         for j:=1 to 52 do
         begin
             if (CompareStr(StructureOrdinateur[j].LoginConnecte, Login)=0)
                and (CompareStr(StructureOrdinateur[j].NomOrdinateur,
NomOrdi)=0)
                and (Identifiant = StructureOrdinateur[j].Iden) then
             begin
                StructureOrdinateur[j].LoginConnecte := '';
                StructureOrdinateur[j].NomOrdinateur := '';
                StructureOrdinateur[j].Iden := 0;
                 StructureOrdinateur[j].Img := 'Можу розмовляти';


fillchar(StructureOrdinateur002,sizeof(StructureOrdinateur002),0); // Розмір
структури дорівнює нулю
                UnCranDeMoins :=0;
```

```
                for k :=1 to 52 do
                begin
                     if (StructureOrdinateur[k].LoginConnecte = '')
                     and (StructureOrdinateur[k].NomOrdinateur ='')
                     and (StructureOrdinateur[k].Iden = 0)  then
                     begin
                         inc(UnCranDeMoins);
                     end
                     else
                     begin
                          StructureOrdinateur002[k-UnCranDeMoins].LoginConnecte
:= StructureOrdinateur[k].LoginConnecte;
                          StructureOrdinateur002[k-UnCranDeMoins].NomOrdinateur
:= StructureOrdinateur[k].NomOrdinateur;
                          StructureOrdinateur002[k-UnCranDeMoins].Iden :=
StructureOrdinateur[k].Iden;
                          StructureOrdinateur002[k-UnCranDeMoins].img :=
StructureOrdinateur[k].img;
                     end;
                end;

                fillchar(StructureOrdinateur,sizeof(StructureOrdinateur),0);
                for l :=1 to 52 do
                begin
                    if (StructureOrdinateur002[l].LoginConnecte <> '')
                       and (StructureOrdinateur002[l].NomOrdinateur <>'')
                       and (StructureOrdinateur002[l].Iden <> 0) then
                    begin
                         StructureOrdinateur[l].LoginConnecte :=
StructureOrdinateur002[l].LoginConnecte;
                         StructureOrdinateur[l].NomOrdinateur :=
StructureOrdinateur002[l].NomOrdinateur;
                         StructureOrdinateur[l].Iden :=
StructureOrdinateur002[l].Iden;
                         StructureOrdinateur[l].img :=
StructureOrdinateur002[l].img;
                    end;
                end;
            end;
        end;
end;

//Запис часу введення повідомлення

procedure T_IM.TimerInformations_ICQ_OSCAR_eurTimer(Sender: TObject);
var
   i : integer;
begin
     {відправляє повідомлення з з'єднаннь усіх ПК клієнта до сервера}
     for i:=0 to NumeroArriveConnexion-1 do
     begin
        ServerSocket.Socket.Connections[i].SendText(#13+MessageInfo);  //
відправлення даних клієнту
     end;
end;

//Інформація про повідомлення

function T_IM.MessageInfo : string;
var
    i : integer;
    MessageInfoListe_ICQ_OSCAR_eur,MessageInfoListe_ICQ_OSCAR_eur1 : string;
begin
     Application.ProcessMessages;
     MessageInfoListe_ICQ_OSCAR_eur1 := '';
     MessageInfoListe_ICQ_OSCAR_eur := '';
     MessageInfoListe_ICQ_OSCAR_eur := '@щ*@';
     for i:=1 to NumeroArriveConnexion + 3 do
```

```
     begin
        if StructureOrdinateur[i].LoginConnecte <> '' then
        begin
            MessageInfoListe_ICQ_OSCAR_eur1 := MessageInfoListe_ICQ_OSCAR_eur1 +
'µ'
            + StructureOrdinateur[i].LoginConnecte + '#IMG#' +
StructureOrdinateur[i].img +
            '#ORDI#'+StructureOrdinateur[i].NomOrdinateur;
        end;
     end;

     if MessageInfoListe_ICQ_OSCAR_eur1 <> '' then
     begin
       MessageInfoListe_ICQ_OSCAR_eur := MessageInfoListe_ICQ_OSCAR_eur +
MessageInfoListe_ICQ_OSCAR_eur1 + 'µ¤@*!@¤';
       result:= MessageInfoListe_ICQ_OSCAR_eur;
     end
     else
     begin
         result:= '??';
     end;

end;

//Виведення імені користувача, з яким спілкуємся

procedure T_IM.NomClientParti;
var
   i : integer;
   MessageInfoDepart : string;
begin
     MessageInfoDepart := ' ';
     for i:=0 to NumeroArriveConnexion-1 do
     begin
        ServerSocket.Socket.Connections[i].SendText(#13+MessageInfoDepart);  //
відправлення даних клієнту
     end;
end;

{Зміна вкладки}
procedure T_IM.PageControlChanging(Sender: TObject;
  var AllowChange: Boolean);
begin
  if ((Sender as TPageControl).ActivePage = TabSheetServeur) and (ServeurActif =
TRUE) then
     AllowChange := FALSE
end;

procedure T_IM.LabeledEdit1KeyPress(Sender: TObject; var Key: Char);
var
  i : integer;
  MessageAEnvoyer : string;
begin
        if (Key = #13) and (ServeurActif = TRUE) then
        begin
           key := #0;
           for i:=0 to NumeroArriveConnexion-1 do
           begin
             ServerSocket.Socket.Connections[i].SendText(#13+'Сервер'+' >>
'+надсилає повідомлення'');  // відправлення даних клієнту
           end;
        end;
        if (Key = #13) then
              key := #0;
end;

procedure T_IM.Fermer1Click(Sender: TObject);
var
   i : integer;
```

```
begin
  ClipCursor(nil); // Мишка знаходиться у полі програмного забезпечення.
  if (not ClientConnecter) and (not ServeurActif) then
       Application.Terminate;

  if ServeurActif = TRUE then
  begin
    TimerNombresClientsActuels.Enabled := FALSE;
    TimerNombresClientsActuels.Interval := 100;
    for i:=0 to NumeroArriveConnexion-1 do
    begin
        ServerSocket.Socket.Connections[i].SendText(#13+'Підключено');  //
відправлення даних клієнту
    end;
    ServerSocket.Active := FALSE;
    Application.Terminate;
  end;

end;


{Управління звуком}
initialization
  NumeroArriveConnexion :=0;

end.
```

Unit_IM.pas – параметри інтерфейсу серверної частини

```
unit ChoixCouleurPanel;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TChoixCouleur = class(TForm)
    GroupBox1: TGroupBox;
    BitBtn1: TBitBtn;
    RadioGroupSons: TRadioGroup;
    RadioGroupVisible: TRadioGroup;
    GroupBoxCouleur: TGroupBox;
    ColorDialog1: TColorDialog;
    Panel1: TPanel;
    StaticText1: TStaticText;
    Button1: TButton;
    StaticText2: TStaticText;
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Panel1Click(Sender: TObject);
  private
    { Private declarations }
    procedure SystemeCommande(var Msg : TMessage);
    message WM_SysCommand; // перехоплення повідомлень SysCommand

  public
    { Public declarations }
  end;

var
  ChoixCouleur: TChoixCouleur;

implementation

uses Unit_IM;


{$R *.dfm}

procedure TChoixCouleur.FormCreate(Sender: TObject);
var
   Style : LongInt; // видалити рядок заголовка
begin
   {Наступні 4 рядка можуть зробити невидимим заголовок}
   Style := GetWindowLong(Handle,GWL_STYLE); // Встановлення поточного стилю
   Style := Style and not WS_CAPTION;     // Видаляє поточний стиль для
відображення заголовка
   SetWindowLong(Handle,GWL_STYLE,Style);   // Робить зміни
   SetWindowPos(Handle,0,0,0,0,0,SWP_FRAMECHANGED or SWP_NOMOVE or SWP_NOSIZE or
SWP_NOZORDER);  // Останій рядок

end;

procedure TChoixCouleur.BitBtn1Click(Sender: TObject);
var
  n: Integer;  // Служить для активації і CtrlAltShift та AltTab
begin
     SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, @n, 0); // Дозволяє закрити
CtrlAltShift та AltTab
     ClipCursor(nil); // Мишка знаходиться у полі програмного забезпечення.
```

```
     Couleur := ChoixCouleur.Panel1.Color;
     _IM.Colorier;
     ChoixCouleur.Close;
end;

procedure TChoixCouleur.SystemeCommande(var Msg : TMessage);
begin
    //успадкований;
 // Інструкція так, що повідомлення обробляються зазвичай
    if Msg.wParam = sc_Close then ; // Закривається по Alt-F4
end;



procedure TChoixCouleur.FormShow(Sender: TObject);
var
   n: Integer;  // Служить для активації і CtrlAltShift та AltTab
   Rect: TRect;   // Обмеження зони для мишки
begin
   application.ProcessMessages;
   SystemParametersInfo(SPI_SCREENSAVERRUNNING, Integer(TRUE), @n, 0);  //
Служить для деактивації і CtrlAltShift та AltTab

   { Перетворює координати листа на екран}
   Rect.TopLeft:= ClientToScreen(ClientRect.TopLeft);
   Rect.BottomRight:= ClientToScreen(ClientRect.BottomRight);
   ClipCursor(@Rect); // Обмеження переміщення миші у клієнтської області
плагіна.

   ChoixCouleur.Panel1.Color := Couleur;
   ChoixCouleur.GroupBox1.Color := Couleur;
   ChoixCouleur.RadioGroupSons.Color := Couleur;
   ChoixCouleur.RadioGroupVisible.Color := Couleur;
   ChoixCouleur.GroupBoxCouleur.Color := Couleur;

end;

procedure TChoixCouleur.Button1Click(Sender: TObject);
begin
   Panel1.Color := clBtnFace;
end;

procedure TChoixCouleur.Panel1Click(Sender: TObject);
var
  n: Integer;  // Служить для активації і CtrlAltShift та AltTab
  Rect: TRect;   // Обмеження зони для мишки
begin
  SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, @n, 0); // Дозволяє закрити
CtrlAltShift та AltTab
  ClipCursor(nil); // Мишка знаходиться у полі програмного забезпечення.

  If (ColorDialog1.Execute=True) Then
  Begin
     Panel1.Color:=ColorDialog1.Color;
     Repaint;
  End;

  SystemParametersInfo(SPI_SCREENSAVERRUNNING, Integer(TRUE), @n, 0);  //
Служить для деактивації і CtrlAltShift та AltTab
  { Перетворює координати листа на екран}
  Rect.TopLeft:= ClientToScreen(ClientRect.TopLeft);
  Rect.BottomRight:= ClientToScreen(ClientRect.BottomRight);
  ClipCursor(@Rect); // Обмеження переміщення миші у клієнтської області
плагіна.

end;

end.
```

Unit_MsgPerso.pas - передача текстових повідомлень

```
unit Unit_MsgPerso;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, cxGraphics, cxControls, cxLookAndFeels, cxLookAndFeelPainters,
  cxContainer, cxEdit, IdAntiFreezeBase, IdAntiFreeze, IdCustomTCPServer,
  IdTCPServer, IdCmdTCPServer, IdExplicitTLSClientServerBase, IdFTPServer,
  IdUserAccounts, IdComponent, IdTCPConnection, IdTCPClient, IdFTP,
  IdBaseComponent, IdZLibCompressorBase, IdCompressorZLibEx, ExtCtrls,
  cxProgressBar, cxSplitter, StdCtrls, cxGroupBox, Buttons,
  IdFTPListOutput, Registry, IdFTPList, IdWinSock2 ;

type
  TMsgPerso = class(TForm)
    Panel1: TPanel;
    SpeedButton5: TSpeedButton;
    Login: TLabel;
    SpeedButton1: TSpeedButton;
    BitBtnEnvoyermsgPrive: TBitBtn;
    BitBtn1: TBitBtn;
    Panel2: TPanel;
    cxGroupBox1: TcxGroupBox;
    LabeledEdit1: TEdit;
    cxGroupBox2: TcxGroupBox;
    Memo1: TMemo;
    cxSplitter1: TcxSplitter;
    FontDialog1: TFontDialog;
    Ouvrir: TOpenDialog;
    TimerWizz: TTimer;
    IdCompressorZLibEx2: TIdCompressorZLibEx;
    IdFTP1: TIdFTP;
    IdUserManager1: TIdUserManager;
    IdFTPServer1: TIdFTPServer;
    IdAntiFreeze1: TIdAntiFreeze;
    Timer1: TTimer;
    Transfert: TPanel;
    cxProgressBar1: TcxProgressBar;
    Taux_Transfert: TLabel;
    Niveau_Transfert: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    procedure IdFTPServer1StoreFile(ASender: TIdFTPServerContext;
      const AFileName: string; AAppend: Boolean; var VStream: TStream);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure SpeedButton5Click(Sender: TObject);
    procedure BitBtnEnvoyermsgPriveClick(Sender: TObject);
    procedure TimerWizzTimer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure IdFTPServer1RetrieveFile(ASender: TIdFTPServerContext;
      const AFileName: string; var VStream: TStream);
    procedure IdFTPServer1RemoveDirectory(ASender: TIdFTPServerContext;
      var VDirectory: string);
    procedure IdFTPServer1MakeDirectory(ASender: TIdFTPServerContext;
      var VDirectory: string);
    procedure IdFTPServer1ListDirectory(ASender: TIdFTPServerContext;
      const APath: string; ADirectoryListing: TIdFTPListOutput; const ACmd,
      ASwitches: string);
    procedure IdFTPServer1GetFileSize(ASender: TIdFTPServerContext;
      const AFilename: string; var VFileSize: Int64);
    procedure IdFTPServer1DeleteFile(ASender: TIdFTPServerContext;
      const APathName: string);
    procedure IdFTPServer1ChangeDirectory(ASender: TIdFTPServerContext;
```

```
      var VDirectory: string);
    procedure IdFTP1Work(ASender: TObject; AWorkMode: TWorkMode;
      AWorkCount: Integer);
    procedure IdFTP1WorkBegin(ASender: TObject; AWorkMode: TWorkMode;
      AWorkCountMax: Integer);
    procedure IdFTP1WorkEnd(ASender: TObject; AWorkMode: TWorkMode);
    procedure FontDialog1Apply(Sender: TObject; Wnd: HWND);
  private
    function ReplaceChars(APath: string): string;
    function GetSizeOfFile(AFile: string): Integer;
  public
    FileSize: integer;
    FileName: string;
    STime: extended;
    AbortTransfer: boolean;
  end;

var
  MsgPerso: TMsgPerso;
  Bouge: Boolean;
  cpt: integer = 0;
  AppDir: string;

implementation

uses IdFTPCommon, Unit_IM;

{$R *.dfm}

var
  ServeurEnReception: Boolean = False;
  ServeurAdresseClient: string = '';
  ClientFichier: file;

function GaucheNDroite(substr: string; s: string; n: integer): string;
var i: integer;
begin
  S := S + substr;
  for i := 1 to n do
  begin
    S := copy(s, pos(substr, s) + length(substr), length(s) - pos(substr, s) +
length(substr));
  end;
  result := copy(s, 1, pos(substr, s) - 1);
end;

function TrouverNomLoginQuiRecoitMsg: string;
var
  Login: string;
begin
  Login := MsgPerso.Caption;
  Login := GaucheNDroite(' ', Login, 7);
  result := Login;
end;


procedure TMsgPerso.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  //LabeledEdit1.Font := FontDialog1.Font;
end;

procedure TMsgPerso.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if ServeurEnReception and
    (messagedlg('Ви впевнені, що хочете покинути програму?', mtconfirmation,
[mbYes, mbNo], 0) = mrNo) then
  begin
     //нічого не робить
  end
```

```
  else
  begin
    IdFTP1.Disconnect;
    MsgPerso.Destroy;
    MsgPerso := nil;
  end;
end;

function TMsgPerso.ReplaceChars(APath: string): string;
var
  s: string;
begin
  s := StringReplace(APath, '/', '\', [rfReplaceAll]);
  s := StringReplace(s, '\\', '\', [rfReplaceAll]);
  Result := s;
end;

//Створення форми

procedure TMsgPerso.FormCreate(Sender: TObject);
var
  Registre: TRegistry;
  tmpCouleur: integer;
begin
  IdFTPServer1.Active := True;
  if IdFTP1.Connected then IdFTP1.Disconnect;
end;

//Визначення розміру файлу

function TMsgPerso.GetSizeOfFile(AFile: string): Integer;
var
  FStream: TFileStream;
begin
  try
    FStream := TFileStream.Create(AFile, fmOpenRead);
    try
      Result := FStream.Size;
    finally
      FreeAndNil(FStream);
    end;
  except
    Result := 0;
  end;
end;

procedure TMsgPerso.IdFTP1Work(ASender: TObject; AWorkMode: TWorkMode;
  AWorkCount: Integer);
var
  S: string;
  TotalTime: TDateTime;
  H, M, Sec, MS: Word;
  DLTime: Double;
  AverageSpeed: extended;
begin
  Application.ProcessMessages;
  TotalTime := Now - STime;
  DecodeTime(TotalTime, H, M, Sec, MS);
  Sec := Sec + M * 60 + H * 3600;
  DLTime := Sec + MS / 1000;
  if DLTime > 0 then
  begin
    AverageSpeed := (AWorkCount / 1024) / DLTime;
    S := FormatFloat('0.00 Kb/s', AverageSpeed);
    Taux_Transfert.Caption := S;
  end;
 // if AbortTransfer then IdFTP1.Abort;
  cxProgressBar1.EditValue := AWorkCount/FileSize * 100;
```

```
  Niveau_Transfert.Caption := IntToStr(AWorkCount) + '/' + IntToStr(FileSize) +
' octets';
end;

procedure TMsgPerso.IdFTP1WorkBegin(ASender: TObject; AWorkMode: TWorkMode;
  AWorkCountMax: Integer);
begin
  ServeurEnReception := True;
  AbortTransfer := false;
  STime := Now;
  Memo1.Lines.Add(FileName);
  Taux_Transfert.Caption := '0.00 Kb/s';
  if FileSize < AWorkCountMax then FileSize := AWorkCountMax;
  Niveau_Transfert.Caption := '0 / ' + IntToStr(FileSize) + ' octets';
end;

procedure TMsgPerso.IdFTP1WorkEnd(ASender: TObject; AWorkMode: TWorkMode);
begin
  // SpeedButton2.Visible := false;
  //Memo1.Lines.Clear;
  //if AbortTransfer then Memo2.Lines.Add('Скасування надсилання: ' + FileName)
 // else
  if (FileName <> '') then Memo1.Lines.Add('Передача даних: ' + FileName);
  FileSize := 0;
  FileName := '';
  IdFTP1.Disconnect;
  ServeurEnReception := False;
end;

procedure TMsgPerso.IdFTPServer1ChangeDirectory(ASender: TIdFTPServerContext;
  var VDirectory: string);
begin
  ASender.CurrentDir := VDirectory;
end;

procedure TMsgPerso.IdFTPServer1DeleteFile(ASender: TIdFTPServerContext;
  const APathName: string);
begin
  DeleteFile(ReplaceChars(AppDir + ASender.CurrentDir + '\' + APathname));
end;

procedure TMsgPerso.IdFTPServer1GetFileSize(ASender: TIdFTPServerContext;
  const AFilename: string; var VFileSize: Int64);
var
  LFile: string;
begin
  LFile := ReplaceChars(AppDir + AFilename);
  try
    if FileExists(LFile) then
      VFileSize := GetSizeOfFile(LFile)
    else
      VFileSize := 0;
  except
    VFileSize := 0;
  end;
end;

procedure TMsgPerso.IdFTPServer1ListDirectory(ASender: TIdFTPServerContext;
  const APath: string; ADirectoryListing: TIdFTPListOutput; const ACmd,
  ASwitches: string);
var
  LFTPItem: TIdFTPListItem;
  SR: TSearchRec;
  SRI: Integer;
begin
  ADirectoryListing.DirFormat := doUnix;
  SRI := FindFirst(AppDir + APath + '\*.*', faAnyFile - faHidden - faSysFile,
SR);
  while SRI = 0 do
```

```
  begin
    LFTPItem := ADirectoryListing.Add;
    LFTPItem.FileName := SR.Name;
    LFTPItem.Size := SR.Size;
    LFTPItem.ModifiedDate := FileDateToDateTime(SR.Time);
    if SR.Attr = faDirectory then
      LFTPItem.ItemType := ditDirectory
    else
      LFTPItem.ItemType := ditFile;
    SRI := FindNext(SR);
  end;
  FindClose(SR);
  SetCurrentDir(AppDir + APath + '\..');
end;

procedure TMsgPerso.IdFTPServer1MakeDirectory(ASender: TIdFTPServerContext;
  var VDirectory: string);
begin
  if not ForceDirectories(ReplaceChars(AppDir + VDirectory)) then
  begin
    raise Exception.Create('Unable to create directory');
  end;
end;

procedure TMsgPerso.IdFTPServer1RemoveDirectory(ASender: TIdFTPServerContext;
  var VDirectory: string);
var
  LFile: string;
begin
  LFile := ReplaceChars(AppDir + VDirectory);
end;

procedure TMsgPerso.IdFTPServer1RetrieveFile(ASender: TIdFTPServerContext;
  const AFileName: string; var VStream: TStream);
begin
  VStream := TFileStream.Create(ReplaceChars(AppDir + AFilename), fmOpenRead);
end;

procedure TMsgPerso.IdFTPServer1StoreFile(ASender: TIdFTPServerContext;
  const AFileName: string; AAppend: Boolean; var VStream: TStream);
begin
  if not Aappend then
    VStream := TFileStream.Create(ReplaceChars(AppDir + AFilename), fmCreate)
  else
    VStream := TFileStream.Create(ReplaceChars(AppDir + AFilename), fmOpenWrite)
end;


function TrouverIP(ordinateur: string): string;
var
  WSAData: TWSAData;
  Name, Address: string;
  Phe: PHostEnt;
begin
  WSAStartup(2, WSAData);
  SetLength(Name, 255);
  Phe := GetHostByName(PChar(ordinateur));
  with Phe^ do
    Address := Format('%d.%d.%d.%d', [Byte(h_addr^[0]), Byte(h_addr^[1]),
Byte(h_addr^[2]), Byte(h_addr^[3])]);
  WSACleanup;
  TrouverIP := Address;
end;


procedure TMsgPerso.SpeedButton1Click(Sender: TObject);
var
  MessageBouger: string;
begin
```

```
  MessageBouger := 'MsgBouger' + '#DE#' + EditLogin + '#ЗАГОЛОВОК ПОВІДОМЛЕННЯ#'
+ TrouverNomLoginQuiRecoitMsg + '#ТІЛО ПОВІДОМЛЕННЯ#vient de te rйveiller !
:'')#ЗАВЕРШЕННЯ РОБОТИ#';
    _IM.ClientSocket.Socket.SendText(#13 + MessageBouger);
  Bouge := True;
end;

procedure TMsgPerso.SpeedButton5Click(Sender: TObject);
var
  MessageTransfert: string;
begin
  if not Ouvrir.Execute then Exit;
  AssignFile(ClientFichier, Ouvrir.FileName);
  MessageTransfert := 'MsgTransfert' + '#DE#' + EditLogin + '#ЗАГОЛОВОК
ПОВІДОМЛЕННЯ#' + TrouverNomLoginQuiRecoitMsg + '#ТІЛО ПОВІДОМЛЕННЯ#' +
ExtractFileName(Ouvrir.FileName) + '#ЗАВЕРШЕННЯ РОБОТИ#';
  _IM.ClientSocket.Socket.SendText(#13 + MessageTransfert);
  if not IdFTP1.Connected then
  begin
    IdFTP1.Host := TrouverIP(Login.Caption);
    IdFTP1.UserName := 'root';
    IdFTP1.Password := '';
    IdFTP1.Connect;
  end;
  if (IdFTP1.Connected) then
  begin
    Transfert.Visible := True;
    IdFTP1.TransferType := ftBinary;
    IdFTP1.Put(Ouvrir.FileName, ExtractFileName(Ouvrir.FileName));
    Reset(ClientFichier, 1);
  end;
end;

procedure TMsgPerso.TimerWizzTimer(Sender: TObject);
begin
  if Bouge then
  begin
    if cpt = 10 then
    begin
      Bouge := False;
      cpt := 0;
    end
    else
    begin
      if (cpt mod 2) = 0 then
        MsgPerso.Left := MsgPerso.Left + 5
      else
        MsgPerso.Left := MsgPerso.Left - 5;
    end;
    cpt := cpt + 1;
  end;
  Exit;
end;

procedure TMsgPerso.BitBtn1Click(Sender: TObject);
begin
  if (FontDialog1.Execute) then
  begin
    LabeledEdit1.Font := FontDialog1.Font;
    Memo1.Font := FontDialog1.Font;
  end;
end;


procedure TMsgPerso.BitBtnEnvoyermsgPriveClick(Sender: TObject);
var
  MessagePrive: string;
begin
  if LabeledEdit1.Text <> '' then
```

```
  begin
    if ClientConnecter = TRUE then
    begin
      MessagePrive := 'MsgPrive' + '#DE#' + EditLogin + '#ЗАГОЛОВОК
ПОВІДОМЛЕННЯ#' + TrouverNomLoginQuiRecoitMsg + '#ТІЛО ПОВІДОМЛЕННЯ#' +
LabeledEdit1.Text + '#ЗАВЕРШЕННЯ РОБОТИ#';
      _IM.ClientSocket.Socket.SendText(#13 + MessagePrive);
    end;
  end;
  LabeledEdit1.Clear;
end;


end.
```

AlertMsg.pas - передача програмних повідомлень

```
unit AlertMsg;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Buttons, ImgList;

type
  TAlertMsgF = class(TForm)
    ExecutionTimer: TTimer;
    IconImg: TImage;
    IconsList: TImageList;
    FondImg: TImage;
    TitleLbl: TLabel;
    TextLbl: TLabel;
    procedure ExecutionTimerTimer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    OGhigh: integer; // Оригінальний розмір форми
    WaitBeforeDownCounter: integer;
    WaitBeforeDownTime: integer;
    AlertWindowState : byte; // 0 = змонтувати; 1 = зупинка; 2 = походження; 3 =
інактивації - див. «Const»
    IsRunning: boolean;
  end;

procedure AlertMsgBox(ATitle, AText: string; AIcon:integer=0;
ABeep:boolean=false; WaitBeforeDown:integer=1000; StepTime:integer=10;
OnTextClick:TNotifyEvent=nil);

var
 AlertMsgF: TAlertMsgF;

const // ВИЗНАЧЕННЯ КОНСТАНТ ДЛЯ ПОЛЕГШЕННЯ ВИКОРИСТАННЯ ПРОЦЕДУР АПЕЛЯЦІЇ
 ICON_NONE=0; ICON_INFO=1; ICON_QUESTION1=2; ICON_QUESTION2=3; ICON_WARNING1=4;
 ICON_WARNING2=5; ICON_QUIT=6; ICON_STOP=7; ICON_GO=8; ICON_SECURE=9;
 ICON_VALIDATE=10; ICON_SEARCH=11; ICON_SENDMAIL=12; ICON_NEWMAIL=13;
ICON_USER=14;
 // КОНСТАНТИ ДЛЯ КРАЩОГО РОЗУМІННЯ КОДУ
 WS_UP=0; WS_STOP=1; WS_DOWN=2; WS_DISABLED=3;

implementation

{$R *.dfm}

// ПАРАМЕТРИ ІНІЦІАЛІЗАЦІЇ
procedure TAlertMsgF.FormCreate(Sender: TObject);
var Rect: TRect;
begin
 DoubleBuffered := true;
 OGhigh := ClientHeight;
 FormStyle := fsStayOnTop;
 Left:=Screen.Width-ClientWidth;
 SystemParametersInfo(SPI_GETWORKAREA, 0, @Rect, 0);
 Top := Screen.Height - (Screen.Height - Rect.Bottom)-1; //
 ClientHeight := 1;                                      //
 IsRunning := false;                                     //
end;


// ЗВЕРНЕННЯ ДО MESSAGEBOX
procedure AlertMsgBox(ATitle, AText: string; AIcon:integer=0;
ABeep:boolean=false; WaitBeforeDown:integer=1000; StepTime:integer=10;
OnTextClick:TNotifyEvent=nil);
begin
```

```
 with AlertMsgF do begin


   IsRunning := true;


   // УСТАНОВКА ВІКНА (ТЕКСТ І ЗНАЧОК)
   Caption := ATitle;
   TitleLbl.Caption := ATitle;
   TextLbl.Caption := AText;
   if AIcon = 0 then begin
    IconImg.Picture := nil;
    TitleLbl.Left := 3; TitleLbl.Width := 173;
   end else begin
    IconImg.Picture := nil;
    IconsList.GetBitmap(AIcon-1,IconImg.Picture.Bitmap);
    TitleLbl.Left := 21; TitleLbl.Width := 155;
   end;


   // ПОРЯДОК НАЛАШТУВАННЯ ОДНОГО КЛІКА НА ПОВІДОМЛЕННЯ
   if Assigned(OnTextClick) then begin
     TextLbl.Cursor := crHandPoint;
     TextLbl.OnClick := OnTextClick;
   end else begin
     TextLbl.Cursor := crDefault;
     TextLbl.OnClick := nil;
   end;


   WaitBeforeDownTime := WaitBeforeDown div StepTime;
   ExecutionTimer.Interval := StepTime;
   AlertWindowState := WS_UP;
   Show;
   if ABeep then Beep;
   ExecutionTimer.Enabled := true;
 end;
end;


procedure TAlertMsgF.ExecutionTimerTimer(Sender: TObject);
begin

 if AlertWindowState = WS_DISABLED then begin
  ExecutionTimer.Enabled := false; exit;


 end else if AlertWindowState = WS_UP then begin
       ClientHeight := ClientHeight + 1;
       Top := Top - 1;
       if ClientHeight=OGhigh then begin
        WaitBeforeDownCounter := 0;
        AlertWindowState:=WS_STOP;
       end;


 end else if AlertWindowState = WS_STOP then begin
      inc(WaitBeforeDownCounter);
      if WaitBeforeDownCounter = WaitBeforeDownTime then begin
        WaitBeforeDownCounter := 0;
        AlertWindowState:=WS_DOWN;
      end;


 end else if AlertWindowState = WS_DOWN then begin
       ClientHeight := ClientHeight - 1;
       Top := Top + 1;
       if ClientHeight=1 then begin
        AlertWindowState:=WS_DISABLED;
```

```
        ExecutionTimer.Enabled := false;
        Visible := false;
        IsRunning := false;
       end;
 end;


 FormStyle := fsStayOnTop;
 Application.ProcessMessages;
end;

end.
```

AudioVideo.pas - відео- та аудіозв'язок

```
unit AudioVideo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls,Registry, WinSkinStore, WinSkinData,
  AMixer, MMSystem, ComCtrls, dxGDIPlusClasses, Camera;

type
  TCAudioVideo = class(TForm)
    Mixer: TAudioMixer;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    GroupBox2: TGroupBox;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label3: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    LabelStereo: TLabel;
    Image2: TImage;
    ComboBox1: TComboBox;
    ComboBox2: TComboBox;
    ComboBox3: TComboBox;
    TrackBar: TTrackBar;
    CheckBox: TCheckBox;
    GroupBox3: TGroupBox;
    GroupBox4: TGroupBox;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Image1: TImage;
    ComboBox4: TComboBox;
    ComboBox5: TComboBox;
    ComboBox6: TComboBox;
    TrackBar1: TTrackBar;
    CheckBox1: TCheckBox;
    GroupBox5: TGroupBox;
    GroupBox6: TGroupBox;
    Label5: TLabel;
    Image3: TImage;
    TrackBar2: TTrackBar;
    Camera1: TCamera;
    Panel1: TPanel;
    BitBtn1: TBitBtn;
    TrackBar3: TTrackBar;
    Label10: TLabel;
    Label11: TLabel;
    TrackBar4: TTrackBar;
    procedure FormCreate(Sender: TObject);
    procedure ComboBox3Change(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure MixerControlChange(Sender: TObject; MixerH, ID: Integer);
    procedure TrackBarChange(Sender: TObject);
    procedure CheckBoxClick(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure TrackBar2Change(Sender: TObject);
    procedure TabSheet2Show(Sender: TObject);
    procedure TabSheet1Show(Sender: TObject);
    procedure ComboBox6Change(Sender: TObject);
    procedure ComboBox4Change(Sender: TObject);
```

```
    procedure ComboBox5Change(Sender: TObject);
    procedure TrackBar3Change(Sender: TObject);
    procedure TrackBar4Change(Sender: TObject);
  private
    { Private declarations }
    Setting:Boolean;
  public
    { Public declarations }
  end;

var
  CAudioVideo: TCAudioVideo;

implementation

uses Unit_IM, Unit_MsgPerso;


{$R *.dfm}

procedure TCAudioVideo.FormCreate(Sender: TObject);
var A:Integer;
begin
  For A := 0 to Mixer.MixerCount - 1 do
    ComboBox3.Items.Add (Mixer.ProductName);
  If (ComboBox3.Items.Count > 0) then
    ComboBox3.ItemIndex := 0;
  ComboBox3Change (Sender);

  For A := 0 to Mixer.MixerCount - 1 do
    ComboBox6.Items.Add (Mixer.ProductName);
  If (ComboBox6.Items.Count > 0) then
    ComboBox6.ItemIndex := 0;
  ComboBox6Change (Sender);

end;

procedure TCAudioVideo.MixerControlChange(Sender: TObject; MixerH, ID: Integer);
begin
     ComboBox2Change(Self);
end;

procedure TCAudioVideo.TabSheet1Show(Sender: TObject);
begin
     if(Camera1.Actif)Then
      Camera1.Actif := False;
end;

procedure TCAudioVideo.TabSheet2Show(Sender: TObject);
begin
  if(not Camera1.Actif)Then
    Camera1.Actif := True;
end;

procedure TCAudioVideo.TrackBar2Change(Sender: TObject);
begin
Camera1.FramesPreview := TrackBar2.Position;
end;

procedure TCAudioVideo.TrackBar3Change(Sender: TObject);
begin
Camera1.FramesCaptura := TrackBar3.Position;
end;

procedure TCAudioVideo.TrackBar4Change(Sender: TObject);
begin
Camera1.Secondes:=TrackBar4.Position;
end;
```

```
procedure TCAudioVideo.TrackBarChange(Sender: TObject);
begin
   If (not Setting) then
  begin
    Setting:=True;
    Mixer.SetVolume (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;

procedure TCAudioVideo.BitBtn1Click(Sender: TObject);
begin
  Close;
end;

procedure TCAudioVideo.CheckBoxClick(Sender: TObject);
begin
      If not Setting then
  begin
    Setting:=True;
    Mixer.SetVolume (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,TrackBar.Position,TrackBar.Position,Integer(CheckBox.Checked));
    Setting:=False;
  end;
end;

procedure TCAudioVideo.ComboBox1Change(Sender: TObject);
var A:Integer;
begin
  ComboBox2.Items.Clear;
  For A:=0 to Mixer.Destinations[ComboBox1.ItemIndex].Connections.Count-1 do

ComboBox2.Items.Add(Mixer.Destinations[ComboBox1.ItemIndex].Connections[A].Data.
szName);
  If ComboBox2.Items.Count>0 then
  begin
    ComboBox2.ItemIndex:=0;
    ComboBox2Change (Self);
  end;
end;

procedure TCAudioVideo.ComboBox2Change(Sender: TObject);
var L,R,M:Integer;
    VD,MD:Boolean;
    Stereo:Boolean;
    IsSelect:Boolean;
begin
  Mixer.GetVolume (ComboBox1.ItemIndex,ComboBox2.ItemIndex-
1,L,R,M,Stereo,VD,MD,IsSelect);
  Setting:=True;
  TrackBar.Visible:=not VD;
  Label1.Visible:=not VD;
  Label3.Visible:=VD;
  If TrackBar.Visible then
    TrackBar.Position:=L;
  CheckBox.Visible:=not MD;
  Label2.Visible:=not MD;
  Label4.Visible:=MD;
  If CheckBox.Visible then
    CheckBox.Checked:=M<>0;
  If (Stereo) then
    LabelStereo.Caption := '- Стерео -'
  else
    LabelStereo.Caption := '- Моно -';
  Setting:=False;
end;

procedure TCAudioVideo.ComboBox3Change(Sender: TObject);
```

```
var A:Integer;
begin
  If (ComboBox3.ItemIndex >= 0) AND (ComboBox3.ItemIndex < Mixer.MixerCount)
then
    Mixer.MixerId := ComboBox3.ItemIndex;
  ComboBox1.Items.Clear;
  If Mixer.MixerCount>0 then
  begin
    For A:=0 to Mixer.Destinations.Count-2 do
      ComboBox1.Items.Add (Mixer.Destinations[A].Data.szName);
    If ComboBox1.Items.Count>0 then
    begin
      ComboBox1.ItemIndex:=0;
      ComboBox1Change (Self);
    end;
  end
  else
  begin
    ComboBox1.OnChange:=nil;
    ComboBox2.OnChange:=nil;
    TrackBar.OnChange:=nil;
    CheckBox.OnClick:=nil;
    MessageDlg ('Немає міксеру у системі!',mtError,[mbOK],0);
  end;
  Setting:=False;
end;

procedure TCAudioVideo.ComboBox4Change(Sender: TObject);
var A:Integer;
begin
  ComboBox5.Items.Clear;
  For A:=0 to Mixer.Destinations[ComboBox4.ItemIndex + 1].Connections.Count-1 do
    ComboBox5.Items.Add(Mixer.Destinations[ComboBox4.ItemIndex +
1].Connections[A].Data.szName);
  If ComboBox5.Items.Count>0 then
  begin
    ComboBox5.ItemIndex:=0;
    ComboBox5Change (Self);
  end;
end;

procedure TCAudioVideo.ComboBox5Change(Sender: TObject);
var L,R,M:Integer;
    VD,MD:Boolean;
    Stereo:Boolean;
    IsSelect:Boolean;
begin
  Mixer.GetVolume (ComboBox4.ItemIndex,ComboBox5.ItemIndex-
1,L,R,M,Stereo,VD,MD,IsSelect);
  Setting:=True;
  TrackBar1.Visible:=not VD;
  Label6.Visible:=not VD;
  Label8.Visible:=VD;
  If TrackBar1.Visible then
    TrackBar1.Position:=L;
  CheckBox1.Visible:=not MD;
  Label7.Visible:=not MD;
  Label9.Visible:=MD;
  If CheckBox1.Visible then
    CheckBox1.Checked:=M<>0;
  Setting:=False;
end;

procedure TCAudioVideo.ComboBox6Change(Sender: TObject);
var A:Integer;
begin
  If (ComboBox6.ItemIndex >= 0) AND (ComboBox6.ItemIndex < Mixer.MixerCount)
then
    Mixer.MixerId := ComboBox6.ItemIndex;
```

```
  ComboBox4.Items.Clear;
  If Mixer.MixerCount>0 then
  begin
    For A:=1 to Mixer.Destinations.Count-1 do
      ComboBox4.Items.Add (Mixer.Destinations[A].Data.szName);
    If ComboBox4.Items.Count>0 then
    begin
      ComboBox4.ItemIndex:=0;
      ComboBox4Change (Self);
    end;
  end
  else
  begin
    ComboBox4.OnChange:=nil;
    ComboBox5.OnChange:=nil;
    TrackBar.OnChange:=nil;
    CheckBox.OnClick:=nil;
    MessageDlg ('Немає міксеру у системі!',mtError,[mbOK],0);
  end;
  Setting:=False;
end;

end.
```

About.pas - довідка

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    Image5: TImage;
    Image6: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи комутації
користувачів на основі централізованої служби миттєвого обміну повідомленнями');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Буравченко К.О.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент  Чумак Олексій Васильович ');
Memo1.Lines.Add('                   гр. КН-22М-2');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Кіровоград 2014');
Memo1.Lines.Add('');

end;

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```