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 обміну інформацією у мережі на основі протоколу Signal”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-1
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Кострик В.О.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 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

Кострику Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 32-13 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 обміну інформацією у мережі на основі протоколу Signal*

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

Кострик В.О. Дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal. 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 обміну інформацією у мережі на основі протоколу Signal.

Метою розробки є дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal.

Об'єктом дослідження є процес обміну інформацією у мережі на основі протоколу Signal.

Предметом дослідження є методи обміну інформацією у мережі на основі протоколу Signal.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal.

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 Embarcadero Delphi.

Ключові слова: комп'ютерні науки, обміну інформацією, мережа, Signal

ABSTRACT

Kostryk V.O. Research and software implementation of a network information exchange system based on the Signal protocol. 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 the system of information exchange in the network based on the Signal protocol.

The purpose of the development is the research and software implementation of the information exchange system in the network based on the Signal protocol.

The object of the study is the process of information exchange in the network based on the Signal protocol.

The subject of research is methods of information exchange in the network based on the Signal protocol.

The research methods are based on the methods of the theory of building computer networks, the methods of mathematical statistics, and the methods of software development.

The result of the work is the software implementation of the information exchange system in the network based on the Signal protocol.

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 Embarcadero Delphi environment.

Keywords: computer science, information exchange, network, Signal

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи..... 6
1.2 Область застосування..... 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 8
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 23
2.3 Розгорнута постановка завдання 29
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 30
3.1 Опис функціонування системи 30
3.2 Розробка структурної схеми..... 38
3.3 Розробка функціональної схеми 41
3.4 Розробка діаграми процесів..... 43
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 46
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 46
4.2 Захист розробленого програмного забезпечення..... 61
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 64
6 НАУКОВА НОВИЗНА 67

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 68
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 68
7.2 Розрахунок трудомісткості розробки програмної продукції 70
7.3 Визначення чисельності виконавців і планового фонду зарплати 72
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 77
7.5 Визначення собівартості розробки та ціни програмної продукції. 81
7.6 Визначення об'єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції. 84
7.7 Визначення експлуатаційних витрат 85
7.8 Визначення економічної ефективності програмної продукції. 86
7.9 Висновок 88
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 89
8.1 Вступ 89
8.2 Аналіз умов праці 90
8.3 Розробка заходів з охорони праці 92
8.4 Пожежна безпека 95
8.5 Дослідження інформаційного навантаження на програміста 97
8.6 Висновки до розділу 100
9 ОСНОВНІ ВИСНОВКИ 101
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 103

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ОС	–	операційна система
BMP	–	графічний формат
JPEG	–	графічний формат
GIF	–	графічний формат
IP	–	інтернет протокол
Signal	–	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. Спілкування змінилося раз і назавжди, коли люди почали використовувати гаджети та програми для здійснення дзвінків та обміну текстовими повідомленнями. На перший погляд все чудово. Ми можемо спілкуватися з іншими людьми з будь-якої точки світу та витрачати менше грошей на міжнародні дзвінки. Всі перераховані вище фактори сприяють популярності месенджерів.

Однак є певні недоліки, які підривають нашу довіру до зазначених месенджерів – конфіденційність і безпека даних. Ось чому захищені програми обміну повідомленнями стають дедалі популярнішими. Програма для обміну повідомленнями Signal, популярне рішення для обміну зашифрованими повідомленнями, є чудовим прикладом того, як зробити спілкування безпечним.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal.

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

– Огляд існуючих систем обміну інформацією у мережі на основі протоколу Signal.

– Дослідження системи обміну інформацією у мережі на основі протоколу Signal.

– Програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal.

Об'єктом дослідження є процес обміну інформацією у мережі на основі протоколу Signal.

Предметом дослідження є методи обміну інформацією у мережі на основі протоколу Signal.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод обміну інформацією у мережі на основі протоколу Signal.

– Розроблено вітчизняний продукт обміну інформацією у мережі на основі протоколу Signal, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі обміну інформацією у мережі на основі протоколу Signal.

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 обміну інформацією у мережі на основі протоколу Signal, 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

Миттєва передача повідомлень (англ. Instant messenger, IM) є формою комунікації в реальному часі між двома або більшою кількістю користувачів. Користувачі в IM спілкуються за допомогою передачі тексту. Текст передається через комп'ютери, з'єднані по мережі Інтернет. Сучасні мессенджери відомі й сьогодні завоювали популярність, їх представляє Signal.

Також є інші мессенджери. Серед найбільш знаменитих – QIP, Miranda і багато які інші.

Вибираючи мережу для спілкування, більшість людей орієнтуються на переваги своїх друзів або колег по роботі. Але якщо частина з них використовує, наприклад, Signal, а інша – MSN, доводиться встановлювати відразу два клієнтських додатки. При цьому за рубежем досить серйозною популярністю користуються також мережі AIM і Yahoo!, а в Україні нерідко використовують Skype і Googe Talk.

Крім того, окремо варто виділити досить популярну технологію Jabber, що на відміну від деяких інших систем інтернет-пейджинга (IM – систем миттєвої передачі повідомлень) повністю відкрита й безкоштовна (мережі AIM, Signal, MSN і Yahoo! – рекламно-оплачувані за рахунок показу баннерів). Головне перевага Jabber у тому, що будь-яка людина вправі відкрити свій власний сервер Jabber, що навіть може бути ізольований від Глобальної мережі.

1.2 Область застосування

Отже, що робить приватний месенджер Signal унікальним, як працює Signal і чому його вважають попередником найбезпечніших безкоштовних

програм для обміну повідомленнями на ринку? Щоб відповісти на ці запитання, розглянемо програму месенджера Signal докладніше.

Що таке додаток Signal і як він працює

Отже, як працює Signal? Програма для обміну повідомленнями Signal була створена на основі існуючих програм RedPhone і TextSecure і була запущена в березні 2015 року компанією Open Whisper Systems. Додаток безкоштовний для користувачів і отримує дохід лише від пожертвувань і грантів.

Програму для обміну повідомленнями Signal також відрізняє від інших програм для чату те, що вихідний код Signal доступний на GitHub для всіх, хто хоче ознайомитися з ним або перевірити наявність недоліків у безпеці. Фактично, у 2016 році Signal пройшов незалежний аудит. Результатом аудиту стала програма для шифрування сигналів, яка офіційно визнана безпечною.

Крім того, їх репозиторій налаштовано за допомогою BitHub, щоб досвідчені розробники могли легко налаштувати та розгорнути Signal і заробити на цьому гроші, якщо їхній запит на отримання буде прийнято.

Signal дозволяє здійснювати зашифровані дзвінки з будь-якої точки світу; те саме стосується текстових повідомлень Signal. На відміну від SMS, зашифровані повідомлення Signal захищені наскрізним шифруванням. Читайте далі, щоб дізнатися більше про це питання.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal, 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

Розглянемо існуюче програмне забезпечення обміну інформацією в мережі.

Імовірно, існує сотня чи більше різних месенджерів. Але який із них поважає вашу конфіденційність, стійкий до спроб цензури з боку урядів і великих технологій і простий у використанні для норм?

Я обмежив це порівняння лише десятьма месенджерами:

– XMPP, піонер FOSS і об'єднаних миттєвих повідомлень. XMPP – це просто протокол, а не програма; у цій статті він представлений програмою Android Conversations.

– Delta Chat, який базується на звичайній електронній пошті і, отже, має найбільшу теоретичну базу користувачів серед усіх федеративних месенджерів.

– Matrix, новий і дуже перспективний федеративний протокол, тут представлений провідним клієнтським елементом.

– Signal, найпопулярніший месенджер FOSS із наскрізним (E2E) шифруванням.

– Telegram, найпоширеніший месенджер FOSS і лідер ринку в деяких країнах.

– Threema, найпоширеніший месенджер FOSS із шифруванням E2E і можливістю анонімних облікових записів.

– Facebook Messenger, другий за популярністю месенджер у світі.

– Line, найпопулярніший месенджер в Японії, Тайвані та Таїланді.

– Viber, оскільки це найпопулярніший месенджер у деяких країнах.

– WhatsApp, найпопулярніший месенджер №1 у світі.

Деякі месенджери, які є актуальними, але не потрапили до списку:

– iMessage (+FaceTime): недоступний на Android.

– RCS (Google Messages): недоступний на iOS.

– Будь-що інше, створене Google: Google Chat, Google Meet, Google Duo, Google Hangouts, важко встежити, що вони просувають на даний момент. Вони продовжують застаріти та переосмислювати свої програми для обміну миттєвими повідомленнями. Слідувати за цим надто втомливо, і вони все одно ніколи не зроблять прориву. Треба було зберегти Google Talk, який був просто клієнтом XMPP.

– Skype: не є лідером ринку в жодній країні (більше), канібалізовано Microsoft Teams.

– Briar: одноранговий месенджер із найбільшим рівнем розуму, але низьким рівнем сприйняття, обмеженою функціональністю (лише текстові повідомлення), недоступний для iOS.

– Tox: ще один добре відомий P2P-месенджер, підтримує більше платформ, ніж Briar, але має низьку популярність.

– ICQ: одне з перших і дуже популярних миттєвих повідомлень. Він все ще існує, але з невеликою кількістю активних користувачів.

– WeChat: лідер ринку в Китаї та третій за популярністю чат у світі, але це кошмар конфіденційності, і немає причин використовувати WeChat, якщо ви не китаєць або живете в Китаї.

– KakaoTalk: лідер ринку в Південній Кореї, але низький рівень впровадження в інших країнах.

– IRC: протокол із багатьма клієнтами та об’єднанням серверів. На відміну від XMPP, наприклад, IRC зосереджується переважно на групових чатах, а не на миттєвих повідомленнях 1:1.

– Discord: досить популярний, але здебільшого зосереджений на групових і голосових чатах. Кошмар конфіденційності.

– Кік: Досить популярний, але ніде не №1

– Snapchat: досить популярний, але більше схожий на соціальні медіа, ніж на чисте миттєве повідомлення, і не №1 в жодній країні.

– QQ: другий за популярністю месенджер у Китаї. Але якщо WeChat не потрапить у список, QQ також не потрапить.

– Wire, Wickr, Slack, Microsoft Teams, BBMe, Mattermost: фокус на корпоративних користувачів.

– Simplex, Session, Jami: чудова конфіденційність, але низька адаптація.

– Instagram, LinkedIn: переважно соціальні мережі, а не месенджер.

– Mumble, Teamspeak: фокус голосового чату.

Навколо безліч месенджерів, і на перший погляд вони мало чим відрізняються. Однак, якщо придивитися ближче, відмінності стануть очевидними під поверхнею, особливо в тому, що стосується безпеки та захисту конфіденційності. Наступне порівняння показує, як найпопулярніші месенджери протистоять один одному.

WhatsApp – найпопулярніший месенджер, і в цьому його найбільша перевага. Однак слабкий захист конфіденційності, який є результатом бізнес-моделі сервісу, є ще більшим недоліком.

Meta (попередній Facebook), власник WhatsApp, фінансується за рахунок продажу цільової реклами. Ця бізнес-модель вимагає максимально детальної інформації про користувача. Таким чином, WhatsApp не можна використовувати без розкриття особистої інформації, а дані користувачів використовуються Meta в маркетингових цілях.

Threema фінансується за рахунок продажу додатків. Послугою можна користуватися без надання будь-яких особистих даних, а система з самого початку розроблена таким чином, щоб генерувати якомога менше даних користувача, наскільки це технічно можливо.

Конфіденційність за проектом

Здебільшого Threema та WhatsApp пропонують однакові функції, і майже немає різниці в зручності використання. Однак, коли йдеться про безпеку та конфіденційність даних, ці дві служби відрізняються майже на кожному рівні. Threema була розроблена з нуля з урахуванням високого рівня безпеки та зменшення обсягу даних, тоді як бізнес-модель WhatsApp базується на використанні особистих даних у маркетингових цілях.

Надання особистої інформації

Щоб використовувати звичайні служби чату, необхідно розкрити особисту інформацію. WhatsApp, наприклад, вимагає від користувачів надати свій номер телефону. Threema, з іншого боку, не змушує користувачів розкривати будь-яку особисту інформацію. Замість телефонного номера унікальним ідентифікатором є Threema ID (тобто випадковий рядок символів). Пов’язати номер телефону чи адресу електронної пошти зі своїм ідентифікатором Threema можна, але необов’язково. Отже, Threema можна використовувати повністю анонімно, що є важливою функцією з точки зору захисту конфіденційності: якщо немає персональних даних, жодні особисті дані ніколи не можуть бути використані неправомірно.

Доступ до адресної книги

WhatsApp потрібен доступ до адресної книги користувача. Контактні дані передаються на сервер, де дані зберігаються постійно. Без доступу до адресної книги користуватися послугою неможливо (без серйозних обмежень). Threema, з іншого боку, дозволяє користувачам вирішувати, надавати чи ні доступ до адресної книги. Програма повністю функціональна без доступу до адресної книги. Якщо користувач вирішує синхронізувати адресну книгу з метою пошуку контактів на Threema, контактні дані хешуються, надсилаються на сервер і негайно видаляються з сервера після завершення синхронізації.

Відкритий код і зовнішній аудит

Щоб забезпечити повну прозорість, програми Threema є відкритими. Завдяки відтворюваним збіркам також є засіб для перевірки того, що опублікований код (на даний момент програми для Android) дійсно відповідає програмам, доступним для завантаження. Крім того, Threema регулярно залучає зовнішніх експертів для проведення комплексних перевірок безпеки. WhatsApp, з іншого боку, не є відкритим кодом, і не було опубліковано жодних незалежних перевірок безпеки. Це означає, що немає способу перевірити заяви компанії щодо безпеки та захисту конфіденційності.

Бізнес-модель диктує обробку метаданих

Метадані – це всі дані, які беруть участь у спілкуванні, крім фактичного вмісту, наприклад, будь-яка доступна інформація про відправника й одержувача, властивості повідомлення, а також дата й час та інші обставини передачі. Facebook, власник і оператор WhatsApp, фінансується за рахунок продажу цільової реклами, і тому має економічний інтерес до метаданих, які є максимально промовистими. Систематично збираючи метадані та поєднуючи їх із даними з інших служб (наприклад, Instagram), можна створювати детальні профілі користувачів. Завдяки цим профілям користувачів рекламу можна продавати за високими цінами, оскільки її можна націлювати на конкретні демографічні групи.

Threema базується на прозорій бізнес-моделі, яка сумісна з кредо «Конфіденційність за проектом». Послуга фінансується за рахунок продажу програми, тобто користувачі платять за послугу. Система була розроблена з нуля з урахуванням безпеки та зменшення кількості метаданих. Створюються лише дані, необхідні для роботи служби, і вони ніколи не зберігаються довше, ніж технічно необхідно.

І Signal, і Threema були розроблені з урахуванням безпеки та конфіденційності. Signal користується видатною репутацією серед експертів, і це,

безперечно, надійне рішення з точки зору безпеки. Що стосується конфіденційності, однак, у порівнянні з Threema з'являється вражаючий недолік.

Signal вимагає від користувачів розкривати особисту інформацію. З іншого боку, Threema можна використовувати анонімно: користувачам не потрібно вказувати свій номер телефону чи адресу електронної пошти. Той факт, що Signal, будучи постачальником IT-послуг у США, підпадає під дію Закону CLOUD, лише погіршує цей дефіцит конфіденційності.

Оскільки Telegram уже пропонував додаткове наскрізне шифрування, коли ця технологія ще не була широко поширеною, її іноді неправильно називають «безпечною» донині. Однак Telegram принципово відрізняється від безпечних рішень, таких як Threema, оскільки це, по суті, хмарне рішення.

За замовчуванням Telegram постійно зберігає повідомлення на своєму сервері, де вони теоретично можуть бути прочитані постачальником послуг у будь-який час. Threema, з іншого боку, виключає можливість того, що будь-хто, крім передбачуваного одержувача, може читати повідомлення завдяки наскрізному шифруванню. Після доставки повідомлення негайно й безповоротно видаляються з сервера Threema.

Однак його легкий підхід до безпеки та конфіденційності дозволяє Telegram пропонувати деякі функції, які орієнтовані на безпеку рішення, такі як Threema, не можуть забезпечити без значних технічних зусиль.

Месенджери не враховуються

Месенджери на основі протоколу Matrix, такі як Element, не були взяті до уваги, оскільки об'єднання призводить до значних недоліків конфіденційності. Наприклад, повідомлення та метадані постійно зберігаються на **всіх** залучених серверах, що означає, що кожен оператор сервера може відстежувати, хто з ким спілкується в який момент часу. Таким же чином, для всіх операторів серверів очевидно, хто є членами груп, і оператор домашнього сервера користувача, теоретично, навіть може отримати доступ до його списку контактів.

З точки зору безпеки та захисту конфіденційності, Threema не зрівняється. Це єдина служба, якою можна користуватися анонімно, тобто без надання особистої інформації (наприклад, номера телефону чи адреси електронної пошти). Signal також розроблено з урахуванням безпеки та захисту даних, але оскільки це служба США, вона підпадає під дію Закону CLOUD і вимагає від користувачів розкривати особисту інформацію.

Telegram – це хмарне рішення, яке не можна вважати «безпечним» за будь-яким визначенням цього слова: повідомлення не тільки не зашифровані за умовчанням, вони постійно зберігаються на сервері, де постачальник послуг (або хакери) можуть читати їх у будь-який час. WhatsApp застосовує наскрізне шифрування; однак дані користувача можуть використовуватися в маркетингових цілях відповідно до політики конфіденційності, послуга не відповідає GDPR і потребує як персональних даних, так і доступу до адресної книги.

Перераховані сервіси мають більш-менш схожий спектр функцій. Звичайно, усі вони підтримують текстові та голосові повідомлення, а також голосові та відеодзвінки. Однак завдяки своїй безпеці Telegram може запропонувати найбільше функцій. І Telegram, і Threema дозволяють користувачам залишатися анонімними для своїх партнерів по чату.

Незважаючи на те, що Signal зосереджується на безпеці та конфіденційності, сервіс підтримує «самознищення» повідомлень, що є суперечливою функцією з точки зору безпеки. [6]

Усі сервіси можна використовувати на настільних комп’ютерах. Будучи хмарним рішенням, Telegram також має певні переваги в цій категорії. Signal, WhatsApp і Threema майже на одному рівні з точки зору портативності, яка скоро зміниться з появою рішення Threema для кількох пристроїв.

Threema для iOS: чат на комп'ютері без підключення до смартфону

Нова інноваційна настільна програма Threema доступна для всіх користувачів iOS у вигляді стабільної бета-версії, яку можна використовувати як щоденний драйвер.

Наступне покоління програми для настільних комп’ютерів було перероблено з нуля, і воно не лише має повністю оновлений і сучасний інтерфейс користувача зі значно швидшим часом відгуку, але також пропонує підтримку кількох пристроїв: навіть коли ваш смартфон вимкнено або не підключені до Інтернету, ви можете спілкуватися в чаті з комп’ютера.

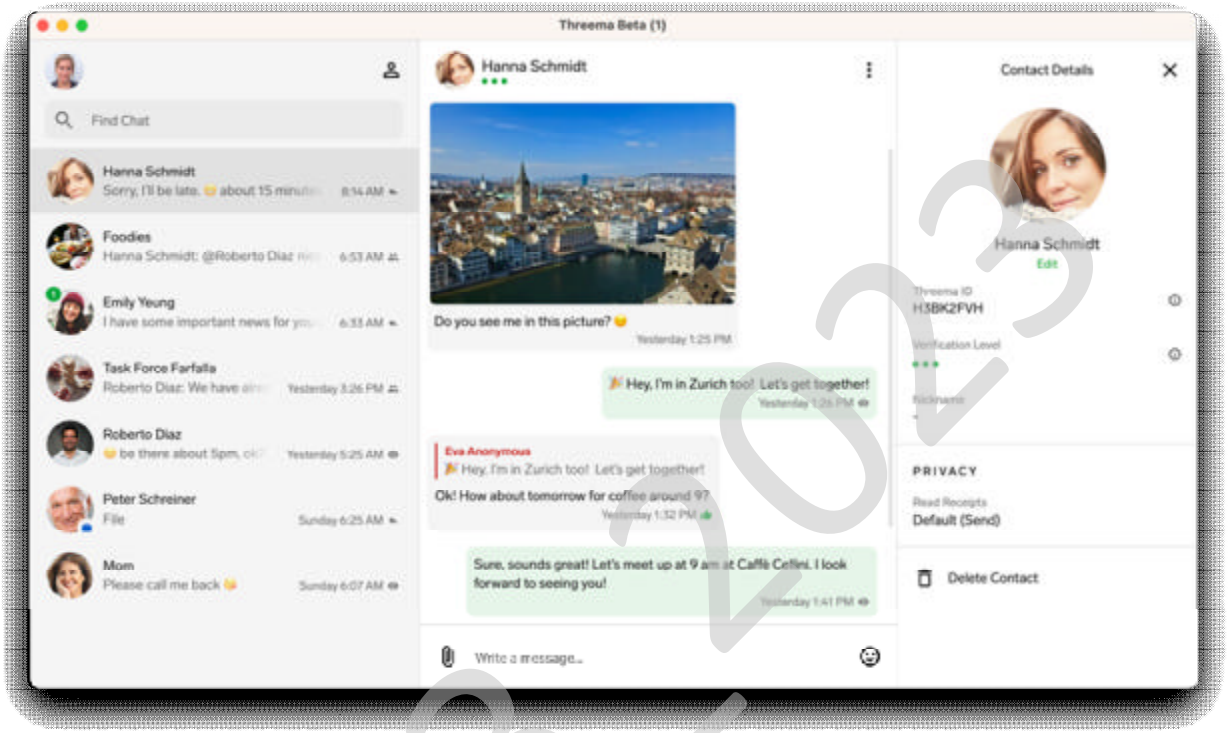

Рисунок 2.1 – Оптимізований і зрозумілий інтерфейс користувача: відомості про контакт і групу тепер відображаються на окремій панелі поруч із вікном чату

При розробці інноваційної архітектури для кількох пристроїв ми приділили особливу увагу обмеженню метаданих. Ось чому «сервер-посередник», який синхронізує повідомлення між пов’язаними пристроями, логічно відокремлений від сервера чату: сервер чату (який усе ще пересилає повідомлення від одного ідентифікатора Threema до іншого) не знає, який ідентифікатор Threema ID зберігає на сервері. сервер-посередник належить.

Сервер-посередник, у свою чергу, не знає ідентифікатора Threema групи пристроїв. Детальний технічний огляд буде опубліковано пізніше.

Ось як ви використовуєте нову програму для комп'ютера:

1. Настільний комп'ютер: завантажте поточну бета-версію Threema для настільного комп'ютера

2. Пристрій iOS: оновіть програму Threema до версії 5.5.3.

3. Програма Threema для iOS: перейдіть до «Налаштування > Настільний комп'ютер/Інтернет > Зв'язаний пристрій (бета-версія)», натисніть «Додати пристрій» і дотримуйтеся вказівок на екрані.

Threema 2.0 для робочого столу – це бета-версія програмного забезпечення, яке все ще розробляється. Деякі функції поки що недоступні, але будуть постійно додаватися.

Підтримка пристроїв Android триває.

Порівняння комунікаційних інструментів для компаній

1. Дізнайтеся, чим інструменти для співпраці та програми для взаємодії відрізняються від безпечних месенджерів для бізнесу.

На перший погляд, незліченні комунікаційні рішення, доступні для компаній, здаються майже нерозрізненими. Однак при ближчому розгляді можна виявити значні відмінності, зокрема щодо безпеки та конфіденційності. Це порівняння порівнює інструменти бізнес-комунікації один з одним і показує, у чому полягають відмінності.

Крім Threema Work, є кілька інших бізнес-месенджерів, які стверджують, що вони орієнтовані на безпеку та конфіденційність: Wire, Wickr від Amazon і Stashcat. Однак у багатьох відношеннях різні інструменти комунікації для компаній підходять до безпеки та конфіденційності дуже по-різному, деякі з яких не зосереджені ні на безпеці, ні на конфіденційності. Наприклад, деякі з так званих «захищених» служб зв'язку та співпраці не мають основ захисту даних, таких як послідовне наскрізне шифрування, або їхня архітектура не є децентралізованою.

З іншого боку, відмінності між інструментами безпечного зв'язку менш разючі, що стосується функцій. Хоча деякі служби підтримують аудіо- та відеоконференції, більшість із них не застосовують наскрізне шифрування до конференц-дзвінків. Завдяки конфігурації додатка в панелі управління всі параметри конфігурації завжди доступні в Threema Work, навіть якщо система MDM недоступна. Крім того, при використанні Threema Work не потрібні додаткові послуги для надсилання «інформаційних бюлетенів» всьому персоналу або окремим відділам.

Незважаючи на те, що Threema Work пропонує найвищий рівень безпеки та найповніший захист даних, це все одно найдоступніший і, безумовно, найпопулярніший бізнес-месенджер, особливо враховуючи базу користувачів звичайної програми Threema.

У таких інструментах, як Google Chat і MS Teams, де співпраця займає центральне місце, спілкування відіграє другорядну роль. Ці інструменти для співпраці не тільки пропонують широкий спектр функцій, але й забезпечують глибоку інтеграцію в існуючу інфраструктуру та плавний перехід від одного пристрою до іншого. Все це має високу ціну, оскільки безпека та захист конфіденційності відходять на другий план. Google Chat і MS Teams зберігають повідомлення на центральному сервері та не використовують наскрізне шифрування, що означає, що вони не підходять для обміну конфіденційними даними в компаніях. Використання інструментів для спільної роботи на смартфоні часто буває незручним і рідко ефективним через величезну кількість функцій, налаштувань і опцій. На мобільних пристроях користувачі зазвичай віддають перевагу інтуїтивно зрозумілим і простим програмам, таким як WhatsApp або Threema Work.

На відміну від інструментів для співпраці, бізнес-месенджер Threema Work фокусується на безпеці та захисті даних, і, на відміну від американських комунікаційних рішень, немає жодних сумнівів щодо його відповідності GDPR. Замість непрозорої міжконтинентальної серверної мережі Threema Work

використовує лише локальні сервери в Швейцарії *, і завдяки обмеженню основних функцій забезпечується легкий і ефективний зв'язок між командами, а також між начальством і співробітниками.

Основна мета додатків для взаємодії – це поширення інформації. Їх мета – повністю замінити електронну пошту та покращити взаємодію між керівництвом, керівниками та співробітниками за допомогою внутрішніх соціальних мереж і подібних каналів зв'язку. З цим прагненням програми для взаємодії прагнуть досягти набагато більшого, ніж прості програми для чату. Однак залишається відкритим питання, чи дійсно персонал використовує функції, відмінні від функції чату, належним чином. У багатьох випадках програма для чату, яка оптимізована, містить лише основні функції, а також підтримує комунікацію зверху вниз, є більш ефективною, ніж програма для взаємодії, яка поєднує багато різних і часто непов'язаних інструментів в одному місці та наповнює персонал усією інформацією.

З точки зору безпеки та конфіденційності комунікаційні програми, такі як Beekeper і Staffbase, не зрівняються з бізнес-месенджером Threema Work. Вони базуються на централізованій архітектурі та не використовують наскрізне шифрування, що означає, що постачальник послуг цих корпоративних засобів зв'язку теоретично може отримати доступ до будь-якого вмісту в будь-який час.

На відміну від програм чату для приватних користувачів, бізнес-рішення пропонують консоль, де адміністратори компанії можуть керувати співробітниками. Завдяки високій гнучкості Threema Work сервіс легко адаптується до потреб кожної організації. У сценаріях BYOD Threema Work (на відміну від інших рішень) дозволяє налаштувати програму без зовнішньої системи MDM. Threema Work поєднує в собі широкі параметри попередньої конфігурації з простим розповсюдженням програм.

Усі рішення підтримують інтеграцію в служби каталогів. Зовнішні користувачі можуть приєднатися до Threema Work без необхідності завершувати виснажливий процес реєстрації та навіть без надання номера телефону чи

електронної адреси. Крім того, Threema Work також можна обмежити закритою групою користувачів.

Не пропонувати опцію архівування чатів користувачів на центральному сервері (наприклад, з метою відповідності) є навмисним рішенням Threema Work, оскільки це, теоретично, дозволить постачальнику послуг отримати доступ до вмісту повідомлень, що, у свою чергу, відобразить переваги безпеки наскрізного шифрування абсолютно марні.

Як і слід було очікувати, Threema Work значно випереджає конкурентів щодо захисту даних. Від інтеграції користувача, де не потрібні ні адреса електронної пошти, ні номер телефону, до розташування та роботи сервера, до суворого уникнення збору даних, інші служби повинні взяти Threema Work як орієнтир. Threema Work – єдиний сервіс, який підтримує власні сервери.

Стовпами безпеки Threema Work є послідовне наскрізне шифрування всього вмісту, з одного боку, і його децентралізована архітектура, з іншого боку. Хоча не дивно, що інструменти для співпраці та програми для взаємодії не пропонують наскрізне шифрування, варто відмітити, що існують також так звані «захищені» бізнес-месенджери, які не захищають обмінювані повідомлення за допомогою цієї техніки.

Threema Work – не єдина служба, яка регулярно залучає зовнішніх експертів для проведення комплексних перевірок безпеки. Деякі основні відмінності між Threema Work та іншими службами полягають у перевірці контактів і безпеці консолі адміністрування.

Чим менш безпечна послуга, тим простіше її реалізувати. І навпаки, певні функції недоступні (або лише в обмеженій мірі) у безпечних рішеннях, які використовують наскрізне шифрування.

Threema Work надає всі основні функції, які користувачі очікують від сучасного месенджера, а також деякі корисні інструменти для повсякденної роботи, наприклад політику неробочого часу та функцію опитування. Основною

перевагою дотримання найнеобхіднішого є простота використання, яку пропонує Threema Work.

Окрім таких аспектів, як безпека та функціональність, ціна також є ключовим фактором при виборі корпоративного месенджера.

На відміну від Threema Work та інших безпечних месенджерів, рішення для співпраці MS Teams і Google Chat недоступні як окремі продукти з окремим тарифним планом. Не всі постачальники послуг прозорі щодо своїх цін, але згідно з наявною інформацією, Threema Work є найдоступнішим варіантом.

Trillian

Trillian – один із самих потужних і гарних інструментів обміну миттєвими повідомленнями. Програма може працювати із протоколами Signal і Jabber. За замовчуванням протокол Jabber відключений, необхідний його окремий запуск із налаштувань програми (розділ «Доповнення»).

Trillian дозволяє активувати одночасно трохи користувачів, що дає програмі перевага в родинах з одним комп'ютером, де кожний член родини жити не може без обміну миттєвими повідомленнями. Історія повідомлень дуже нагадує офісний органайзер, наприклад, Outlook. Всі сеанси спілкування акуратно розкладені по датах. Одне клацання мишею по даті на календарі, і ви перенесіться в «той» день вашого спілкування. І це ще не все – під календарем розташовується графік вашої активності. Ніж швидше ви обмінювалися в той момент повідомленнями зі співрозмовником, тим вище піднімається риса. До недоліків інтернет-пейджеру варто віднести той факт, що Trillian погано розуміє українське кодування. Повідомлення з використанням кирилиці можуть прочитати всі, а повідомлення, які приходять, не завжди читабельні.

Висновки:

– В ідеальному світі кожен би використовував федеративні месенджери, щоб жодна компанія не мала права блокувати користувачів або продавати компанію великим технологічним олігополістам, таким як Meta, Apple, Google,

Microsoft або Amazon, і щоб люди могли самостійно розміщувати своїх власних серверів і таким чином обійти спроби цензури.

- Найперспективнішим на даний момент здається Matrix. Він має гідні клієнтські програми на всіх основних платформах, підтримує всі важливі функції та має додаткове виявлення контактів.
- XMPP має дуже легке і просте в налаштуванні серверне програмне забезпечення. Крім того, це стандартизований протокол, який існує з 1999 року і є основою для інших месенджерів, таких як Whatsapp. Основна проблема з XMPP полягає в тому, що немає клієнтської програми, яка б працювала в усіх основних операційних системах і реалізувала всі важливі функції (наприклад, відеодзвінки). Я вважаю, що це перешкода для більш широкого впровадження серед менш технологічно схильних («О, у вас є iPhone? Тоді вам потрібно завантажити Siskin, а не Conversations»).
- Перевага Delta Chat полягає в тому, що кожен уже має адресу електронної пошти, тому є мільярди потенційних користувачів, які вже зберегли один одного в своїх адресних книгах. Однак у протоколу електронної пошти є свої обмеження, наприклад, ви не можете здійснювати голосові чи відеодзвінки. Через це я не думаю, що Delta Chat матиме реальні шанси коли-небудь замінити WhatsApp.

– Signal – наступний найкращий вибір. Він має найкращу конфіденційність серед усіх основних месенджерів (витік метаданих XMPP і Matrix, Signal – ні), його можна використовувати як програму для SMS за замовчуванням на Android, що є великою перевагою, коли ви намагаєтесь переконати друзів і родину встановити її, і його інтерфейс і налаштування знайомі всім, хто користувався WhatsApp. Однак Signal потрібен ваш номер телефону, тому ви не можете зареєструватися анонімно, якщо у вашій країні заборонено анонімне придбання SIM-карт. В іншому випадку основним недоліком є те, що Signal використовує централізовані сервери, отже, завжди існує ризик того, що Signal буде придбана компанією, яка вороже ставиться до

конфіденційності. Крім того, розробники Signal не допускають сторонніх клієнтів, тому користувачі нішевих операційних систем виключені.

– Telegram має багато функцій і заохочує розробку сторонніх клієнтів. Це, мабуть, найкращий месенджер, якщо ви не надто дбаєте про конфіденційність. Основним недоліком є те, що чати не шифруються E2E за замовчуванням, а секретні чати, зашифровані E2E, не синхронізуються між пристроями. Крім того, він централізований і в минулому був змушений цензурувати (наприклад, у 2022 році вони заблокували понад 60 публічних каналів для німецьких користувачів після того, як їм погрожував уряд Німеччини).

– Threema хороша з точки зору конфіденційності та дозволяє користувачам реєструватися, не повідомляючи свій номер телефону. Основна проблема, яку я бачу, полягає в тому, що програма коштує грошей. Це небагато, і добре знати, як фінансується розробка, але це головний фактор для широкого впровадження.

– Серед великих пропрієтарних месенджерів Whatsapp, Line і Viber E2E шифрують усі повідомлення за замовчуванням, що чудово. Однак ви ніколи не можете бути впевнені, що компанія чи уряд не мають ключів для доступу до ваших повідомлень, оскільки програми мають закритий код і не підлягають перевірці. Крім того, резервні копії приватних месенджерів у хмарному чаті не шифруються E2E. Якщо резервні копії не мають наскрізного шифрування, то навіть якщо ви не завантажуєте свою історію чату в хмару, ваш співрозмовник може це зробити, і це фактично робить решту шифрування E2E безглуздим, оскільки Apple/Google і уряд США може читати ваші повідомлення в резервній хмарі. WhatsApp пропонує E2EE для резервного копіювання як додаткову функцію, однак, якщо *всі* учасники чату не ввімкнули її, це шифрування можна обійти. Крім того, резервні копії на рівні пристрою на iPhone зберігатимуть незашифровані чати Whatsapp в iCloud, якщо користувач не ввімкнув «розширений захист даних» (резервне копіювання пристрою E2EE), що є функцією вибору. Таким чином, більш імовірно, що ваш співрозмовник має

незашифровану резервну копію чату десь у хмарі, і що, зрештою, наскрізне шифрування WhatsApp є просто театром конфіденційності, якщо хтось «серйозний» (наприклад, уряд США) хоче бачити ваші чати.

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 обміну інформацією у мережі на основі протоколу Signal.

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

Програми обміну повідомленнями мають мільярди користувачів по всьому світу. За даними дослідження Messenger People, база користувачів WhatsApp і Facebook Messenger нараховує 1,5 мільярда користувачів у всьому світі. Це означає, що дані мільйонів користувачів знаходяться під загрозою зламу.

Щороку відбувається багато порушень даних. Business Insider заявляє, що в 2018 році особиста інформація мільйонів людей по всьому світу була скомпрометована.

Безпека програми Signal є головною проблемою для розробників, які стоять за програмою. Ось чому програма використовує наскрізне шифрування, оскільки це означає, що всі повідомлення шифруються перед надсиланням і можуть бути розшифровані лише на пристрої одержувача. Єдиний спосіб прочитати повідомлення – на пристрої відправника або одержувача.

Програма приватного обміну повідомленнями Signal покладається на такі криптографічні протоколи:

– Розширений потрійний Діффі-Хеллман (X3DH).
– Алгоритм Double Ratchet, Curve25519.
– AES-256.
– HMAC-SHA256.

Застосовувані протоколи забезпечують захист від MITM (man-in-the-middle).

Після одного конкретного оновлення Signal усі голосові та відеодзвінки були захищені тим самим протоколом Signal, який використовувався для захисту лише текстових повідомлень. Цей протокол був розроблений у 2013 році компанією Open Whisper Systems і вперше реалізований у додатках TextSecure×,

на основі яких пізніше було розроблено додаток для обміну повідомленнями Signal.

Шифрування сигналу також надає користувачам додаткові функції безпеки, такі як захист повідомлень і сповіщень за допомогою парольної фрази. Клавіатура працює в режимі інкогніто і не збирає дані про те, який текст набирається. Крім того, повідомлення про зникнення сигналу досить корисні, як і ті, які використовує Snapchat.

Додаток для обміну повідомленнями Signal також надає механізм перевірки ідентичності ваших контактів за допомогою унікального безпечного номера (відбитка пальця).

Що саме означає бути в безпеці? За даними Electronic Frontier Foundation (EFF), існує сім критеріїв для оцінки того, наскільки безпечним є додаток для чату. Вони є:

– зв'язок, зашифрований під час передачі;

– жоден провайдер не має доступу до ключа, за допомогою якого зашифровано зв’язок;

– незалежна перевірка особи кореспондента;

– захистити минулі комунікації, якщо ключі будуть вкрадені;

– код відкритий для незалежного перегляду;

– добре задокументований криптографічний дизайн;

– незалежний аудит безпеки.

На відміну від інших додатків, месенджер Signal відповідає всім стандартам. Крім того, нижче ви можете побачити оцінку безпеки програми для обміну зашифрованими повідомленнями Signal у порівнянні з іншими програмами для чату:

Разом із ідеєю розробки власного додатка для однорангового чату ви також повинні враховувати всі ризики та можливості, які це може принести. Основною особливістю приватного месенджера Signal є зашифрований обмін повідомленнями, але він, безперечно, має більше функцій, про які варто згадати.

Фактично, програма може виконати всі вимоги навіть для найвибагливіших користувачів:

– Реєстрація за номером телефону. Зручність – головне, а коли справа доходить до реєстрації, усе стає ще простіше, коли вам не потрібно запам'ятовувати паролі чи дані для входу. Ось чому програма обміну повідомленнями Signal використовує номер телефону та код підтвердження для перевірки реєстрації та входу користувача.

– Повідомлення, що зникають. Користувач може встановити таймер від 5 секунд до 1 тижня для зникнення всіх побачених повідомлень. Неможливо навіть зробити скріншот чату, тому що програма просто не дозволяє це зробити. Однак push-сповіщення з повідомленнями про зникнення Signal (незалежно від того, зникає воно чи ні) можна зробити за допомогою знімка екрана, оскільки безпека програми для обміну повідомленнями Signal, яка працює у фоновому режимі, не може блокувати стандартні функції пристрою.

– Голосові та відеодзвінки. Додаток для обміну повідомленнями Signal надає своїм користувачам можливість здійснювати кришталево чисті та безпечні голосові та відеодзвінки, що робить додаток придатним для ділового спілкування.

– Групові чати. За допомогою безпечних чатів «один на один» користувачі можуть вести приватні зашифровані розмови зі своїми друзями. Крім того, сервер програми обміну повідомленнями Signal не має доступу до будь-яких метаданих груп, включаючи піктограми, заголовки та списки учасників.

– Обмін контентом і розваги. Оскільки додаток для обміну повідомленнями Signal досить популярний, він ніколи не припиняє розвиватися та впроваджувати нові функції. Поки що програма безпечного чату дозволяє користувачам ділитися не тільки текстами, але й гіфками, фотографіями, відео, місцезнаходженням, будь-яким документом чи файлом і навіть голосовими повідомленнями (це дуже зручно для швидкого обміну інформацією).

– Особливості платформи. На Android користувачі можуть встановити приватний месенджер Signal як програму за замовчуванням для SMS/MMS, яка

дозволяє надсилати та отримувати SMS-повідомлення користувачам, які не є користувачами Signal, або за відсутності підключення до Інтернету. Єдине застереження полягає в тому, що ці повідомлення не шифруються.

– Безпека та шифрування. Впровадження протоколів безпеки – завдання не з легких, скоріше, потребує величезних зусиль. Однак є кілька порівняно простіших варіантів. Один із них, наприклад, використовує Telegram API (ще один безпечний додаток для чату). Перевагою є те, що вам не потрібно розробляти бек-енд і базу даних, що заощадить ваш час і гроші. Це рішення також має свої недоліки; у вас немає доступу до бази даних або контролю над нею, тому неможливо змінити потік або бути на 100% впевненим у безпеці даних користувача.

Відсутність наскрізного шифрування не означає, що вся ваша історія чату буде пошкоджена та використана з поганими намірами. Насправді, на відміну від додатка для обміну повідомленнями Signal, багато відомих месенджерів отримали свою популярність і базу користувачів без впровадження суперпротоколів безпеки. Візьмемо, наприклад, такі гіганти, як Facebook Messenger і WhatsApp, які лише нещодавно стали шифруватись за допомогою протоколу Signal (розробленого Open Whisper Systems).

Більшість із нас не часто ділиться дуже конфіденційними даними у своїх повідомленнях. Однак наскрізне шифрування служить додатковим заходом безпеки, коли ви надсилаєте будь-яку особисту інформацію, як-от платіжні дані, номери соціального страхування, імена користувачів, паролі тощо. Наскрізне шифрування сигналу та зникнення повідомлень можуть дати вам спокій і впевненість у безпеці даних.

Конфіденційні дані користувача можуть бути викрадені або, що ще гірше, використані шантажистами чи оприлюднені. Таким чином, питання безпеки даних стає ще більш відчутним. І саме тут такі програми, як Signal, потрапляють у центр уваги.

Отже, що таке Signal і що робить його найбезпечнішим додатком для обміну повідомленнями? Приватний месенджер Signal був побудований на основі існуючих додатків RedPhone і TextSecure і був запущений у березні 2015 року компанією Open Whisper Systems. Він використовує протоколи обміну повідомленнями з наскрізним шифруванням (Curve25519, AES-256 і HMAC-SHA256), щоб захистити зв'язок і забезпечити відсутність атак MITM (людина посередині). Що також відрізняє його від інших програм для чату, так це те, що його вихідний код доступний на GitHub для всіх, хто хоче вивчити його або перевірити наявність недоліків у безпеці.

Але що саме означає бути в безпеці? За даними Electronic Frontier Foundation (EFF), існує сім критеріїв для оцінки того, наскільки безпечним є додаток для чату. Вони є:

– передача зашифрована;
– відсутність доступу провайдера до зашифрованого ключа;
– незалежна перевірка особи кореспондента;
– захистити минулі комунікації, якщо ключі будуть вкрадені;
– код відкритий для незалежного перегляду;
– добре задокументований криптографічний дизайн;
– незалежний аудит безпеки.

Відповідно до цих критеріїв, Signal вважається гравцем А.

Як розробити безпечну програму обміну миттєвими повідомленнями, як-от Signal

Основною функцією програми Signal є обмін приватними миттєвими повідомленнями, але вона безумовно має більше функцій, про які варто згадати. Фактично, програма може виконати всі вимоги навіть для найвибагливіших користувачів.

Реєстрація за номером телефону

Зручність – головне, і коли справа доходить до реєстрації, стає набагато простіше, коли вам не потрібно запам'ятовувати паролі чи дані для входу. Ось

чому Signal використовує надісланий номер телефону та код підтвердження для перевірки реєстрації або входу користувача.

Повідомлення, що зникають

Користувач може встановити таймер від 5 секунд до 1 тижня для зникнення всіх побачених повідомлень. Неможливо навіть зробити скріншот чату, тому що програма просто не дозволяє цього робити.

Голосові та відеодзвінки

Signal надає своїм користувачам можливість здійснювати кристально чисті та безпечні голосові та відеодзвінки, тому ця програма також підходить для ділового спілкування.

Групові чати

Разом із безпечними чатами «один на один» користувачі також можуть вести приватні зашифровані розмови зі своїми друзями. Крім того, сервер Signal не має доступу до будь-яких метаданих груп, включаючи значки, заголовки та списки учасників.

Обмін контентом і розваги

Signal ніколи не припиняє розвиватися та впроваджувати нові функції. Поки що додаток дозволяє ділитися не тільки текстом, але й gif-файлами, фото, відео, місцем розташування, будь-яким документом або файлом і навіть голосовими повідомленнями.

Особливості платформи

На Android користувачі можуть встановити Signal як програму для SMS/MMS за замовчуванням, яка дозволяє надсилати й отримувати SMS-повідомлення користувачам, які не користуються Signal, або якщо немає підключення до Інтернету. Єдине, що ці повідомлення не зашифровані.

Безпека та шифрування

Впровадження протоколів безпеки непросте завдання. Це вимагає величезних зусиль. Звичайно, є порівняно прості варіанти. Одним із них є, наприклад, використання Telegram API (ще одна безпечна програма для чату).

Перевагою є те, що вам не потрібно буде розробляти бек-енд і базу даних, що заощадить ваш час і гроші. Але це рішення також має свої недоліки. Ви не матимете доступу до бази даних або контролю над нею, тому буде неможливо змінити потік або бути на 100% впевненим у безпечному розміщенні даних користувача.

Відсутність наскрізного шифрування не означає, що вся ваша історія чату буде пошкоджена та використана з поганими намірами. Насправді, багато відомих месенджерів починали і завойовували свою популярність і базу користувачів без впровадження суперпротоколів безпеки. Візьмемо для прикладу таких гігантів, як Facebook Messenger або WhatsApp, які почали шифруватися лише нещодавно за допомогою протоколу Signal (розробленого Open Whisper Systems).

Більшість із нас рідко ділиться конфіденційними даними у своїх повідомленнях. Однак наскрізне шифрування служить додатковим заходом безпеки, коли ви надсилаєте свою особисту інформацію, як-от платіжні дані, номер соціального страхування, ім’я користувача, пароль тощо. Отже, використання наскрізного шифрування та зникнення повідомлень може дати вам сердечність і впевненість.

Отже, скільки коштує розробка такого рішення, як додаток для обміну повідомленнями Signal? Ціна та терміни значною мірою залежатимуть від функцій, які зрештою матиме ваш месенджер, їхньої складності, дизайну програми (на основі власних або користувацьких елементів керування) і постачальника, якого ви наймете. Ці фактори ускладнюють надання точної оцінки, не знаючи деталей.

Існують сотні різноманітних програм для обміну повідомленнями, але лише кілька десятків із них здобули широку популярність і успіх. Просто клонувати існуючу програму для обміну повідомленнями Signal – не дуже гарна ідея сама по собі. Щоб успішно вийти на ринок, вам слід подумати про інноваційні та унікальні функції або розробити нішевий додаток. Іншими

словами, зробити його відмінним від того, що вже існує – виділити його з натовпу. Візьміть найкращі практики приватного месенджера Signal і додайте свій власний штрих.

Давайте використаємо найпростіші функції, які має мати кожна програма для чату, щоб розрахувати мінімальний необхідний бюджет для такого рішення, як програма для обміну повідомленнями Signal. Вони включають наступне:

Реєстрація:

– Увійти за допомогою номера телефону.

– Підтвердження номера телефону.

Контакти:

– Доступ до всіх контактів.

– Сегментація контактів на ті, у яких встановлений месенджер і не встановлений.

Запрошення та обмін:

– Можливість запросити друзів або поширити інформацію за допомогою вбудованої функції обміну.

Чат:

– Обмін миттєвими повідомленнями один на один.

– Статуси повідомлень (прочитані, непрочитані).

– Редагувати або видаляти повідомлення.

– Надсилайте фотографії з галереї або камери.

– Push-повідомлення.

Додаткові можливості:

– Голосові повідомлення.

– Наклейки.

Розробляючи абсолютно новий безпечний чат-сервіс, слід враховувати, що приватний месенджер Signal не є єдиним у світі. Є деякі сильні конкуренти, такі як Telegram, WhatsApp, Google Allo та Facebook Messenger, і це лише деякі.

Незважаючи на однакову основну функціональність, кожна з цих програм має власні налаштування та унікальні функції, які роблять їх маяками в нескінченному океані програм.

Якщо ви вважаєте, що ваша ідея програми може принести користь користувачам і задовольнити їхні потреби, її однозначно варто спробувати.

3.2 Розробка структурної схеми

Структурна схема розробленого програмного забезпечення обміну інформацією у мережі на основі протоколу Signal зображена на рисунку 3.1.

Розроблене програмне забезпечення складається з наступних блоків:

Серверна частина:

– База даних користувачів.

– База даних інформації про користувачів.

– База даних оффлайн повідомлень.

Клієнтська частина:

– Головне меню.

– Блок авторизації.

– Блок відображення контактів.

– Блок налаштувань.

– Блок встановлення статусів.

– Блок пошуку.

– Блок обміну текстовими повідомленнями.

– Блок обміну мовною інформацією.

– Блок обміну відеоінформацією.

– Блок історії повідомлень по контактам.

У основі програмного забезпечення лежить протокол Signal. Signal – відкритий, але не вільний мережний протокол, що забезпечує обмін миттєвими й

оффлайновими текстовими повідомленнями. У цей момент використовується для систем: AIM (компанія AOL, керована Time Warner).

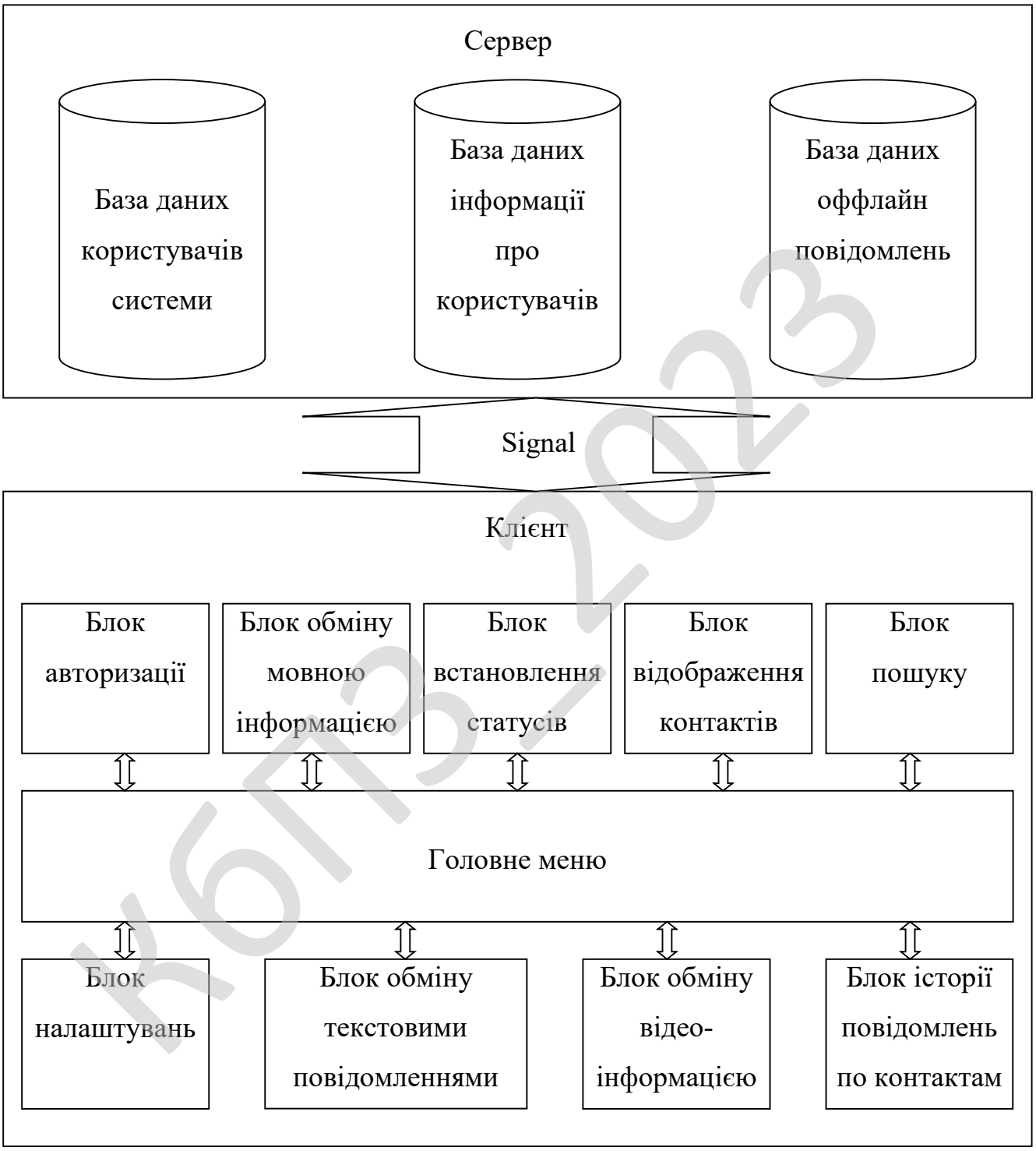

Рисунок 3.1 – Структурна схема системи

Особливості протоколу:

– Кожному користувачеві видається UIN (Unique Identification Number) – унікальний ідентифікаційний номер, по якому користувач однозначно визначається системою й іншими користувачами. У цей час для сумісності з AIM замість UIN використовується поняття ScreenName.

– Користувач має можливість вибрати собі нік, що відіграє роль особистого імені в його повідомленнях. На відміну від UIN, ніки не унікальні для кожного користувача.

– В AOL Instant Messenger функцію UIN грають SN (Screen Name) – так звані екранні імена, унікальні для кожного користувача.

– Протокол підтримує кілька станів, у яких може перебувати користувач. Стани встановлюються користувачем.

Стани:

– Online – доступний.

– Free for chat (F4C) – вільний.

– Away – вдалині від комп'ютера (довго не працював).

– Not available (N/A) – недоступний.

– Occupied – зайнятий.

– Do not disturb (DND) – не турбувати.

– Invisible – не бачимий.

– Offline – відключений.

У програмах-клієнтах сторонніх розроблювачів деякі стани можуть бути відсутніми або мати місце додаткові.

Особливості програми

Програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal надає всі функції, які ви очікуєте від найсучаснішого месенджера для організацій

Основні функції програми:

– Надсилайте текстові та голосові повідомлення.

– Здійснюйте голосові та відеодзвінки.

– Проведення групових дзвінків.

– Надсилайте файли будь-якого типу (PDF, Office документи тощо).

– Діліться фотографіями, відео та місцями.

– Використовуйте програму на робочому столі.

Особливості:

– Створіть опитування.

– Автоматично вимикати сповіщення в неробочий час.

– Мовчки погоджуйтеся або не погоджуйтеся з отриманими повідомленнями.

– Приховати конфіденційні чати та захистити їх паролем за допомогою PIN-коду або відбитка пальця (Android).

– Виберіть темну або світлу тему.

– Підтвердьте свої контакти за допомогою QR-коду.

– Додайте форматування тексту до повідомлень.

– Створення списків розсилки.

– Цитуйте текстові повідомлення.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема програмного продукту, розробленого, у результаті виконання магістерського проектування, складається із взаємодії наступних функціональних блоків:

– Блок інтерфейсу користувача.

– Блок поділу контактів по групах.

– Блок різних тем та скинів для програми.

– Блок повного налаштування інтерфейсу: колір, шрифт, іконки й інше.

– Блок списку контактів, що має повну функціональність. Відображення он-лайн/офф-лайн користувачів зі значками й всією необхідною інформацією.

– Блок відправлення й прийому повідомлень он-лайн/офф-лайн між користувачами через сервер.

– Блок реалізації різних контактів у вкладках одного вікна відправлення повідомлень.

– Блок передачі й прийому файлів від інших користувачів, у тому числі й мультимедійних.

– Блок відправлення й прийому URL.

– Блок установки різних статусів для різних контактів.

– Блок збереження історії вхідних і вихідних повідомлень.

– Блок додавання/видалення користувачів і редагування інформації про користувача.

– Список тих, хто бачить – будьте видимі тільки для деяких користувачів.

– Список тих, хто не бачить – будьте не видимі тільки для деяких користувачів.

– Інформація про останнє перебування користувача у он-лайн.

– Відправлення листа контактів іншому користувачеві.

– Блок пошуку контактів по e-mail/uin/alias/інформації про користувача.

– Блок відправлення й одержання повідомлення про набір тексту.

– Блок підтримки HTTP-проксі.

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

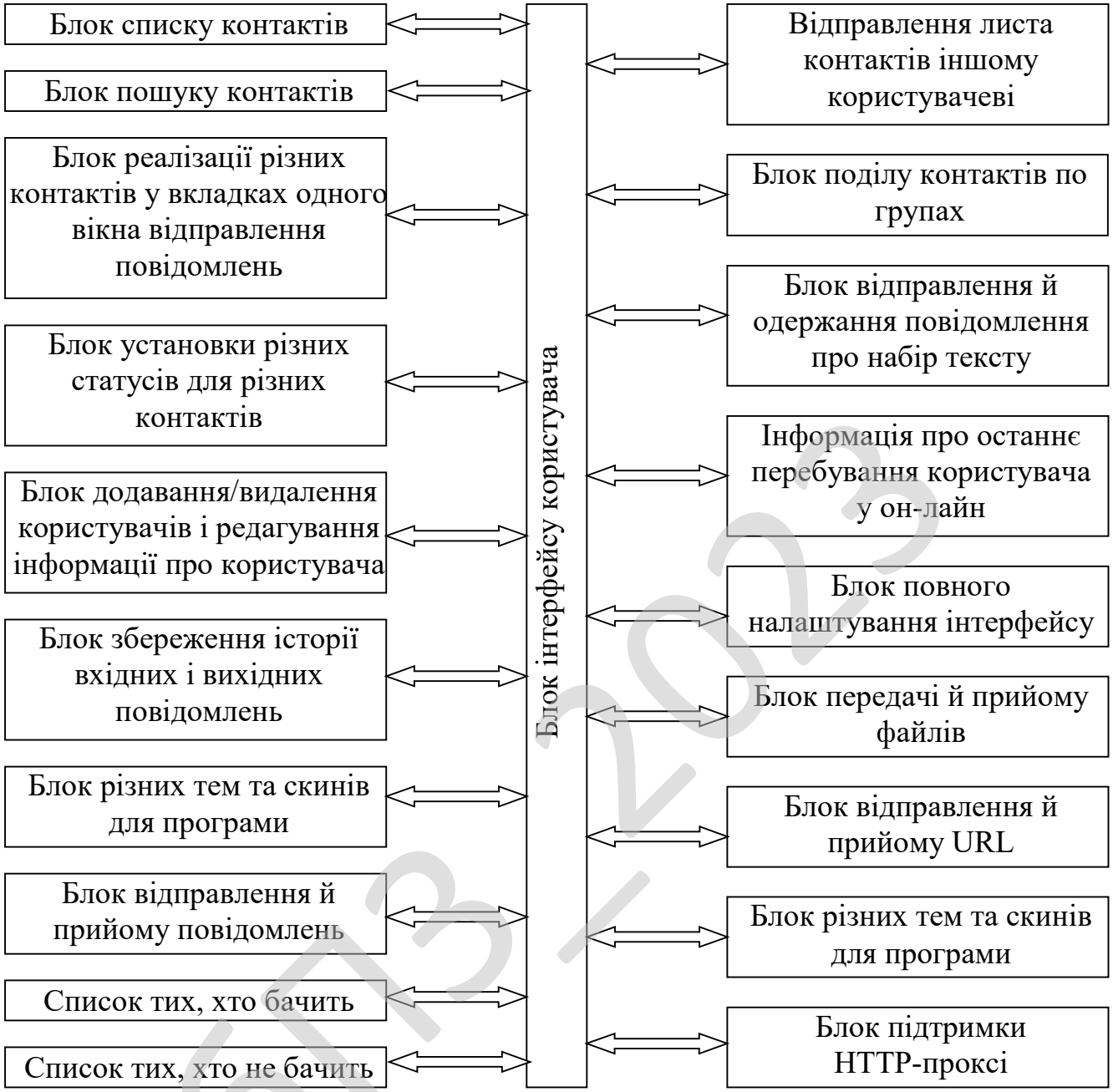

Рисунок 3.2 – Функціональна схема системи

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3. З нього видно, що першим процесом, який запускається у системі, є процес авторизації.

Він взаємодіє з наступним процесами:

– Процес реєстрації.

– Процес підключення до серверу.

Останній процес, у свою чергу, взаємодіє з процесом виведення основного вікна програми.

Процес виведення основного вікна програми взаємодіє з наступними процесами:

– Процес відключення від серверу, який є завершальним у системі.

– Процес виведення списку контактів.

– Процес виведення/приховання списку функцій.

Процес виведення списку контактів взаємодіє з наступними процесами:

– Процес редагування списку контактів.

– Процес вибору контакту зі списку.

Рисунок 3.3 – Діаграма взаємодії процесів

Процес вибору контакту зі списку взаємодіє з наступними процесами:

– Процес обміну текстовими повідомленнями.

– Процес обміну файлами.

– Процес відео– та аудіозв’язку.

Процес виведення/приховання списку функцій взаємодіє з процесом списку функцій, який, у свою чергу, взаємодіє з наступними процесами:

– Процес редагування своїх даних.

– Процес зміни параметрів програми.

– Процес пошуку/додавання/видалення контактів.

– Процес перегляду історії повідомлень.

– Процес формування приватних списків.

– Процес анти-спам фільтру.

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.

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 кроків.

Спершу відбувається виведення вікна авторизації. Після цього, якщо необхідно провести реєстрацію, то відбувається реєстрація користувача, з завданням логіну та паролю. Після цього користувач вводить свій логін та пароль для доступу до програми.

Якщо авторизація пройшла успішно, тоді користувач отримує доступ до програми й виводиться головне вікно програми.

Якщо ж авторизація пройшла не успішно, тоді видається вікно про помилку, й користувачеві пропонується ще раз пройти процедуру авторизації.

Після виведення головного вікна програми відбувається виведення списку контактів та їх статусів.

Якщо користувач обирає встановлення/зміну статусу, тоді відбувається встановлення або зміна статусу користувача, з відповідного списку.

Якщо користувач обирає редагування списку контактів, тоді відбуваються наступні дії:

– Пошук, додавання, видалення контактів.

– Створення груп, та занесення в них контактів.

Якщо користувач обирає натиснення на контакт, тоді він отримує можливість виконувати наступні дії:

– Вивести вікно діалогів.

– Запускається підпрограма обміну даними.

Якщо користувач завершує діалог, то він може або далі працювати з системою, або завершити роботу з системою.

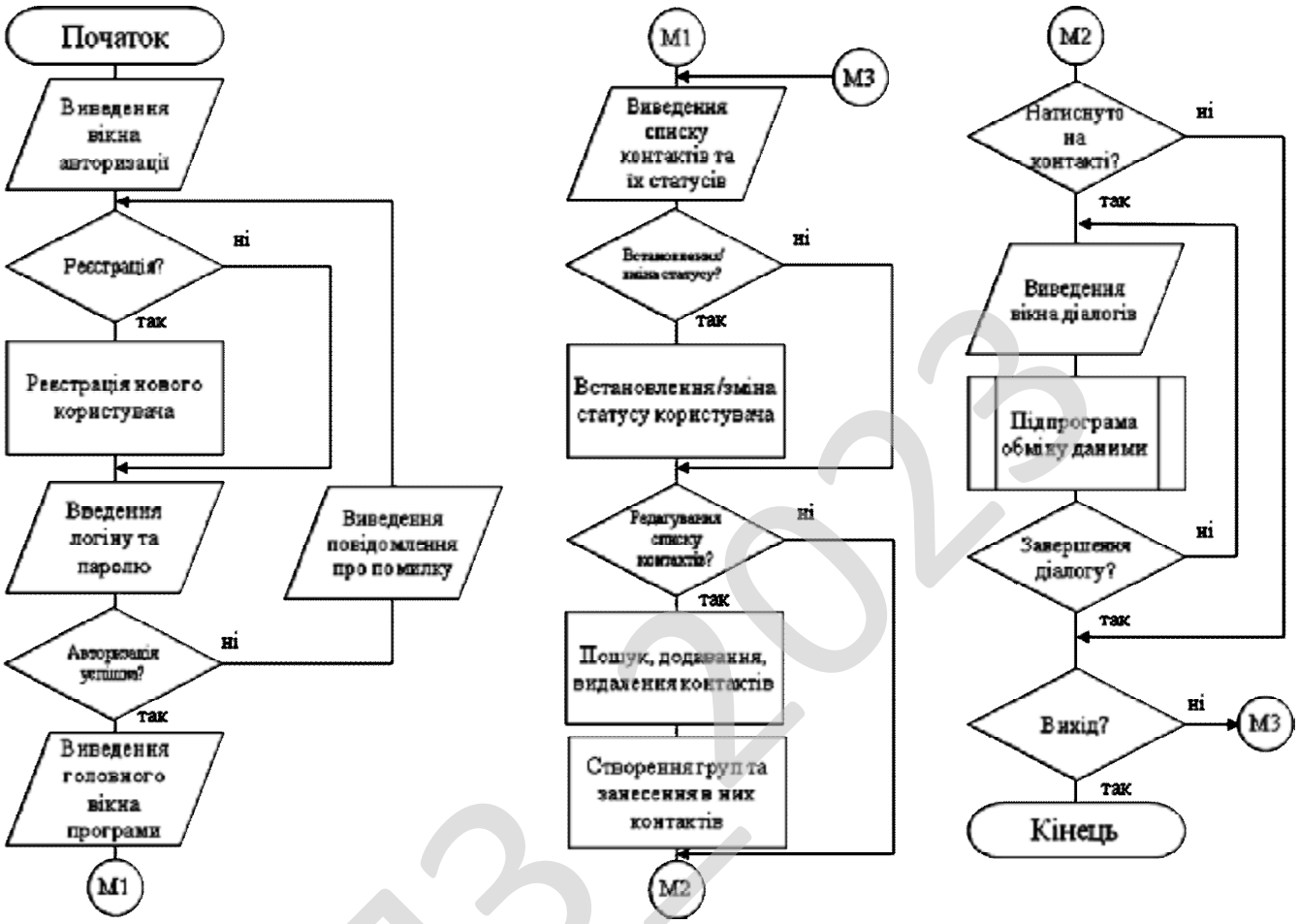

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображено блок-схему підпрограми обміну текстовими повідомленнями. Вона працює наступним чином. Спершу виводиться вікно діалогів.

Якщо користувач обрав одержання повідомлення, то він отримує повідомлення й має можливість на нього відповісти.

Якщо користувач обрав створення повідомлення, то він його створює й відсилає у мережу.

Якщо користувач починає форматувати текст, то він встановлює

параметри форматування тексту, або його частини.

Якщо користувач обирає режим цитування, тоді він виконує наступні дії:

– Виділяє частину тексту в історії повідомлень.

– Додає цитування в текст повідомлення.

Якщо користувач обирає додавання смайлу, тоді він виконує наступні дії:

– Обирає смайл зі списку.

– Додає в текст повідомлення обраний смайл.

Якщо користувач обирає відправлення повідомлення, тоді він відправляє створене повідомлення.

Це усі функції, які може виконати користувач у підпрограмі обміну текстовими повідомленнями.

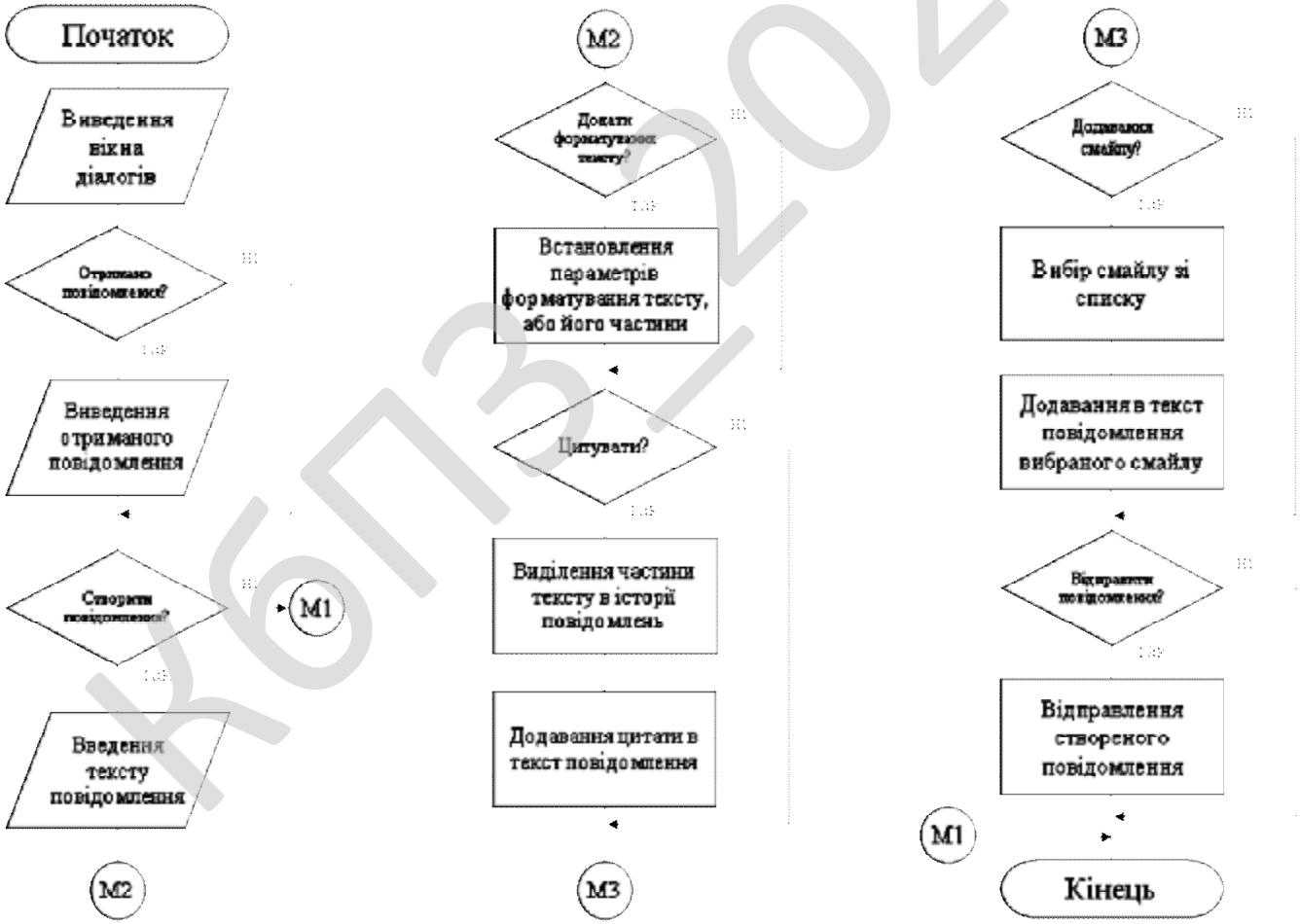

Рисунок 4.2 – Блок-схема підпрограми обміну текстовими повідомленнями

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. Двобайтні 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.

Передача даних при використанні UDP

Ми нарешті-те добралися до вивчення того, заради чого сокети й створювалися: як передавати й одержувати з їхньою допомогою дані. За традицією почнемо розгляд з більше простого протоколу 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. Нагадаємо, що 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*.

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 тимі ж причинам звичайно використовується Recv, а не RecvFrom.

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

Для початку взаємодії клієнт повинен з'єднається із сервером за допомогою функції Connect. Ми вже знайомі із цією функцією, але у випадку TCP вона виконує трохи інші дії. 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.

Вище ми говорили, що 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. Як ми вже говорили вище, 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.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів

диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено основне вікно програми. Воно складається з наступних елементів:

– Вікно контактів.

– Номер Signal.

– Редагування списку контактів.

– Приховати праву частину вікна.

– Кнопка вибору статусу.

– Редагування своїх даних.

– Пошук користувача.

– Додавання контакту.

– Видалення контакту.

– Архів повідомлень.

– Приватні списки.

– Анти-спам фільтр.

– Включити/виключити звук.

– Передати файл.

– Подзвонити.

– Параметри.

На рисунку 5.2 відображено вікно діалогів, з якого видно, що воно складається з наступних логічних блоків:

– Вікно набору повідомлення.

– Вікно обміну повідомленнями.

– Архів.

– Дані користувача.

– Блок смайликів та редагування тексту.

Рисунок 5.1 – Головне вікно програми

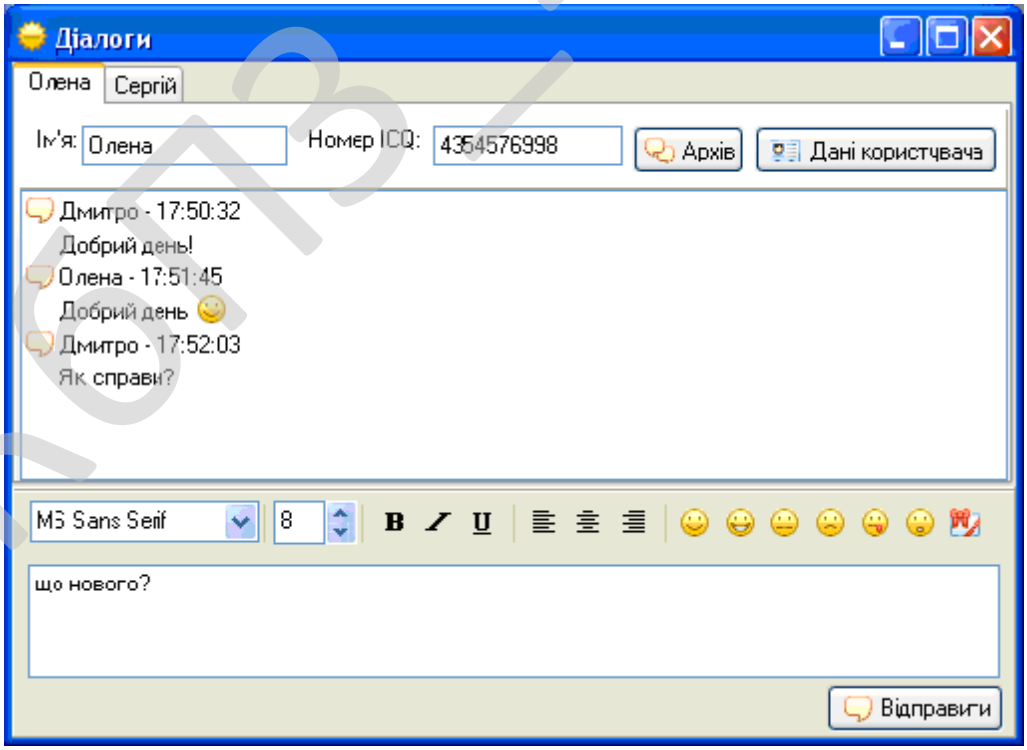

Рисунок 5.2 – Вікно діалогів

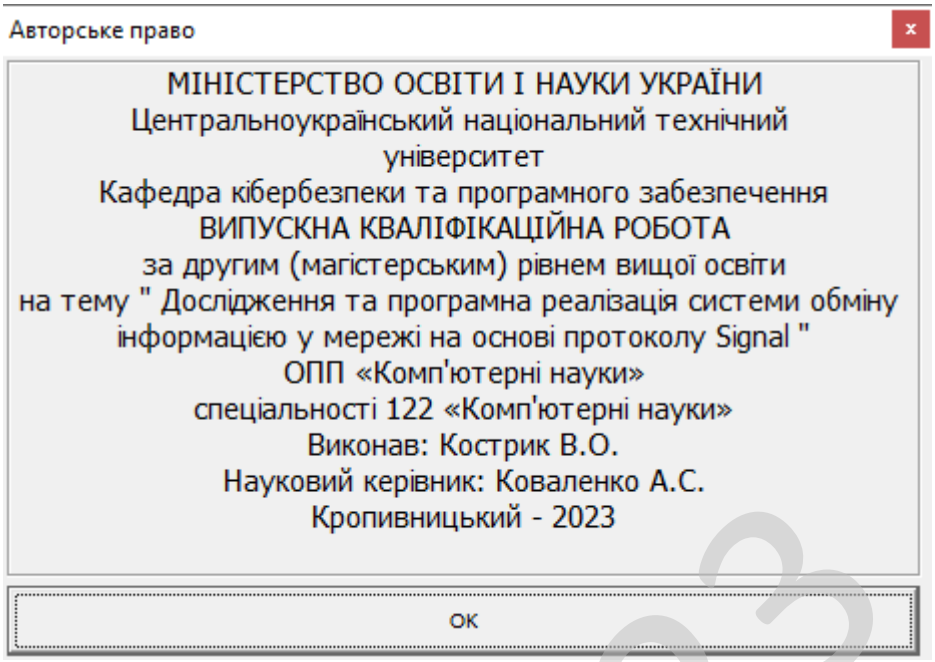

Рисунок 5.3 – Довідка

На рисунку 5.3 наведено довідку, з якої можливо отримати наступну інформацію:

– Автор проекту.

– Тема проекту.

– Керівник проекту.

– Місце розробки проекту.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи обміну інформацією у мережі на основі протоколу Signal.

Метою розробки є дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal.

Об'єктом дослідження є процес обміну інформацією у мережі на основі протоколу Signal.

Предметом дослідження є методи обміну інформацією у мережі на основі протоколу Signal.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод обміну інформацією у мережі на основі протоколу Signal.

– Розроблено вітчизняний продукт обміну інформацією у мережі на основі протоколу Signal, 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 було проведене дослідження та розроблене програмне забезпечення системи обміну інформацією у мережі на основі протоколу Signal. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) 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	130
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 табл. 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 табл. 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	–	13000
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 A – коефіцієнт Боема, A=2,45;

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

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де W_i – 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.:

$$T_{уточн}=T_{ном}\, ПV_j, \qquad (7.3)$$

де $ПVj$ – 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:

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

де С – 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)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot80=135 \text{ люд/день}$$

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	135	Ф 7.1-7.4
Впровадження	13	Д13
Всього	176	–

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,

$$Ч = \frac{176 \cdot 1}{48\text{-}5} = 4 \text{ ставки}$$

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	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
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	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	42,99

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{43 \cdot 2}{1,2} = 72 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=72/(48·8)=0,2 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	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу АДСЛ (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	0,2	0,1
	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	0,2	
Всього		0,8	

Продовження табл. 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,4
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	1	
	Контроль взаєморозрахунків з постачальниками	0,2	
Всього		3,2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,2	0,1
	Створення графічних і стилістичних елементів сайту	0,2	
	Оформлення банерів і промо-сторінок	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		0,8	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	0,2	0,1
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		0,8	

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	11296	22592
Продакт-менеджер	0,4	9000	7200
Інженер-програміст	4	10200	81600
Інженер-електронщик	0,2	8500	3400
Інженер-системотехнік	0,1	8500	1700
Адміністратор мережі	0,1	8500	1700
Системний програміст	0,1	8500	1700
Дизайнер WEB	0,1	8500	1700
Інженер-верстальник	0,1	8500	1700
Бухгалтер-економіст	0,1	8500	1700
Всього за період розробки	$R_{сп}$=6,2	-	$Ф_{роб}$=124992

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.

$$З_{сд} = \frac{124992}{6,2 \cdot 48} = 420 \; грн.$$

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

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

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

S_y – 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

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

де $Ц_м$ – ціна меблів для одного робочого місця, грн.

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

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 за прайсом фірми Комп'ютерторг за 29.10.23 – джерело http://computorg.ua/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		12721
Системний блок		7721
Процесор	Intel Core i5-4570 (4 ядра по 3.2 – 3.6 GHz); 6 MB Cache Memory	-
Системна плата	ASUS B85M-K Socket 1150 Intel B85 OEM Refurbished (SATA II – 2 шт, SATA-3 – 4 шт, 4x USB 3.0, 6x USB 2.0, 4x Audio Ports, LAN (RJ-45), 2x PS/2, Com Port, 2x DP, VGA)	-
Відеокарта	Інтегрована Intel HD Graphics 4600	-
Жорсткий диск	SSD 120Gb + HDD 500 Gb SAMSUNG Barracuda HD502HJ	-
Оперативна пам'ять	8Gb (2x4Gb) DDR3 PC3-12800 Kingston, 1600MHz, 512M x 64, CL9-9-9-27, 1.65V, w/heatsink, HyperX	-
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	Logicpower 8702 – 550w 12cm, 440x180x445, 2 USB 3.0+2 USB 2.0 на передній панелі	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	Подарунок

Продовження таблиці 7.5

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Монітор BenQ GL2450HM Black	3600
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	12721	10176,8	111944,8
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	–	–	–	133576,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	137536	-	-
Всього по групі	137536	30	41260,8
Нематеріальні активи			
4. Нематеріальні активи	13000	25	3250
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	28000	25	7000
Всього по групі	158906	-	33632,75
Разом	$К_р$ = 1717442		$А_р$ = 148543,55

Примітка: вартість автомобіля Citroen Berlingo пасс. 2008 взята за даними електронного ресурсу https://auto.ria.com/uk/auto_citroen_berlingo_pass_33583056.html і складає 121875 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.

$$З_о = 420 \cdot 176/130 = 569 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та 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, %

$$З_д = 569 \cdot 10 \cdot 0{,}01 = 57 \text{ грн}$$

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

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

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(569+57) = 138 \text{ грн}$$

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, %

$$Г_{осп} = 569 \cdot 15 \cdot 0{,}01 = 85 \text{ грн}$$

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, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,33 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 210 \cdot 0{,}33 = 70 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 38):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 24 грн./шт., DVD-R box – 34 грн./шт.

$$З_{М2} = 37 \cdot 24 + 34 = 922 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі 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.

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

$$З_М = (70 + 922 + 1702)/130 = 21 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних 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, %

$$О_п = 569 \cdot 15 \cdot 0{,}01 = 85 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 130 прим.)

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

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 148544 \cdot 2/(130 \cdot 12) = 190 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 569+57+138+85+21+85+190 = 1145 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де $P_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 1145 = 572{,}5 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	569
2. Додаткова зарплата виконавців	$З_д$	57
3. Відрахування на соціальні потреби	$C_{оц}$	138
4. Загальногосподарські витрати	$Г_{осп}$	85
5. Витрати на матеріали	$З_М$	21
6. Освоєння нових операційних систем, мов програмування	$О_п$	85

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	190
8. Повна собівартість програмного забезпечення	$С_п$	1145
9. Плановий прибуток	$П_р$	572,5
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	1717,5
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	343,5
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	2061

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо 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 розрахунків зводимо у таблицю 7.10.

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

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	2061
Всього капітальних витрат	–	2061

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування системи	$З_р$	13420	5368
2. Витрати на амортизацію	$З_{ам}$	0	515,25
Всього витрат за рік	$І$	13420	5883,25

Витрати на технічне обслуговування і підтримку працездатності системи:

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

де $Т_р$ – кількість годин обслуговування за рік, *год.*,

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 100 годин на рік до 40 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 100 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 13420\ грн.$$

до

$$З_{р\ нов} = 40 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 5368\ грн.$$

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	25	–	2061	–	515,25
Всього відрахувань	-	–	2061	–	515,25

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

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

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

$$Ев = (1717,5 - 1145) \cdot 130 - (0,05 \cdot 1408000 + 0,3 \cdot 137536 +$$
$$+0,2 \cdot 121875 + 0,25 \cdot 37031 + 0,25 \cdot 13000) \cdot 2/12 = 49668 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

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.

$$T_в = \frac{310442}{(1717,5 - 1145) \cdot 130 \cdot 12 / 2} = 0,7 \text{ років}$$

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.	130
2. Повна собівартість розробленої програми	Грн.	1145
3. Ціна розробленої програми	Грн.	1717,5
4. Плановий прибуток від реалізації розробленої програми	Грн.	572,5
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1717442
7. Загальний прибуток від реалізації програмної продукції	Грн.	74425
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	49668
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,7
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	2061
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7022
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Роки	0,27

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.26)$$

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

$$E_{сп} = (13420\text{-}5883)\text{-}0{,}25 \cdot 2061 = 7022 \text{ грн.}$$

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.27)$$

$$T_{сп} = \frac{2061}{13420 - 5883} = 0{,}27 \ роки$$

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 праці – це: система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності;

Охорона праці є складовою частиною безпеки життєдіяльності [3,4].

Законом України "Про охорону праці" [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

Загальний нагляд за додержанням норм охорони праці покладено на прокуратуру, спеціальний покладено на професійні спілки. За безпекою контроль праці здійснюють державні й відомчі спеціалізовані інспекції.

У Законодавстві про працю міститься вимоги і норми з виробничої санітарії, техніки безпеки та норми, що регулюють робочий час, час відпочинку, звільнення та переведення на іншу роботу, а також норми праці щодо жінок, молоді, гігієнічні норми і правила, тощо.

8.2 Аналіз умов праці

Фірма дотримується всіх правил з охорони праці і слідкує за їх дотриманням.

При виконанні робіт на комп'ютерах працівникам необхідно дотримуватись вимог загальної інструкції з охорони праці.

До роботи на комп'ютерах допускаються особи, які пройшли: медичний огляд, навчання по професії, вступний інструктаж з охорони праці та первинний інструктаж з охорони праці на робочому місці. В подальшому вони проходять

повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Основним обладнанням робочого місця є монітор, системний блок, миша та клавіатура.

Робочі місця розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1м та між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Монітор розташований на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Елементи робочого місця розміщуються так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Джерела освітлення розташовані з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, також використовують антиблікові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Використовуються скляні поляризаційні фільтри вони забезпечують найкращу якість зображення. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Працівники мають пам’ятки, що раціональною робочою позою вважається положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктєвого суглоба коливається в межах 70 – 90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15 – 20°.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп'ютерах, в тому числі на лазерних та світлодіодних

принтерах, збільшується вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Для попередження травм усе електричне обладнання заземлене. Приступаючи до роботи працівникам необхідно перевірити справність обладнання. В разі виявлення порушень їм треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.3 Розробка заходів з охорони праці

Працівники повинні дотримуватися статті 18 Закону України ”Про охорону праці” згідно цій статті працівники зобов'язані:

– знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;

– співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров'ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу;

– дотримуватись зобов'язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства.

Також повинні виконуватися вимоги безпеки перед початком роботи, працівникам потрібно:

– увімкнути систему кондиціювання в приміщенні;

– перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;

– перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;

– відрегулювати освітленість робочого місця;

– відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;

– приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

– ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

– відрегулювати яскравість монітора, мінімальний розмір світної точки, фокусування, контрастність.

Працівники повинні дотримуватися рекомендацій:

– яскравість монітору – не менше 100Kg/M2;

– відношення яскравості монітора до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;

– мінімальний розмір точки свічення не більше 0,4 мм для монохромного монітора і не менше 0,6 мм для кольорового, контрастність зображення знаку – не менше 0,8.

При виявленні будь-яких неполадок роботу не розпочинати, повідомити про це керівника.

Працівникам потрібно дотримуватися вимог безпеки під час виконання роботи:

– необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;

– для забезпечення несприятливого впливу на користувача пристроїв типу ”миша” належить забезпечувати вільну велику поверхню столу для переміщення ”миші” і зручного упору ліктьового суглоба;

– не дозволяються сторонні розмови, подразнюючі шуми;

– періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран ВДТ та захисний екран протирають ганчіркою, змоченою у спирті. Не дозволяється використовувати рідинні або аерозольні засоби чищення поверхонь комп'ютера.

Працівники не повинні порушувати правил з охорони праці та їм забороняється:

– самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера, 1 раз на півроку повинні відкривати процесор і вилучати пилососом пил і бруд, що накопичилися;

– класти будь-яку предмети на апаратуру комп'ютера;

– закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

Для зняття статичної електрики рекомендується час від часу доторкатися до металевих поверхонь.

Розташувати принтер необхідно поруч з системним блоком таким чином, щоб з’єднувальний шнур не був натягнутий. Забороняється ставити принтери на системний блок.

Для досягнення найбільш чистих, з високою роздільністю зображень і щоб не зіпсувати апарат, має використовуватися папір, вказаний в інструкції до принтера. При зминанні паперу потрібно відкрити кришку і обережно витягнути лоток з папером.

Працівника потрібно дотримуватися вимоги безпеки після закінчення роботи:

– закінчити та записати у пам'ять комп'ютера файл, що знаходиться в роботі;

– вимкнути принтер та інші периферійні пристрої. Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою запобігання попаданню в неї пилу;

– прибрати робоче місце;

– ретельно вимити руки теплою водою з милом;

– вимкнути кондиціонер, освітлення і загальне електроживлення;

– пройти в спеціально обладнаному приміщенні сеанс психофізіологічного розвантаження і зняття втоми з виконанням спеціальних вправ аутогенного тренування.

8.4 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові електропровідники, здійснюється паяння. Виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежного захисту. До засобів гасіння пожежі,

призначених для локалізації невеликих займань, належать вогнегасники, сухий пісок, азбестові ковдри. Приміщення, в який встановлено комп'ютери і де немає необхідності влаштування систем автоматичного пожежогасіння, необхідно оснащувати переносними вуглекислотними з розрахунку 2 шт. на кожні 20 $м^2$ в приміщеннях. Звуковбирне облицювання стін, стель приміщень треба виконувати з негорючих та важко горючих матеріалів.

Електроустановки (можливість їх застосування, монтаж, накладка експлуатація) повинні відповідати вимогам чинних правил улаштування електроустановок, правил технічної експлуатації, електроустановок та інших нормативних документів.

Ймовірність виникнення пожежі від електротехнічного та іншого одиничного виробу не повинна перевищувати 10^{-6} на рік. При короткому замиканні в місцях з'єднання проводів опір практично дорівнює нулю, звідси величина струму досягає дуже великих значень.

Персональні комп'ютери після закінчення роботи повинні відключатися від мережі не рідше 1 разу на квартал, необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами. Не дозволяється розміщувати комп'ютерні зали ЕОМ у підвалах; проводити ремонт вузлів (блоків) ЕОМ безпосередньо у залах, де знаходяться ПК (персональні комп'ютери), залишати без нагляду ввімкнену в мережу електронну апаратуру, яка використовується для контролю ЕОМ.

Електричний струм силою 0,1 А є небезпечним для людини. Для попередження травм усе електричне обладнання повинне бути заземлене. Приступаючи до роботи необхідно перевірити справність обладнання, ізоляцію проводів і надійність заземлення. Доторкання до оголених струмоведучих і незахищених частин в електроустаткуванні забороняється. В разі виявлення порушень ізоляції електропроводів, відкритих струмоведучих частин електроустаткування або порушення заземлення треба негайно повідомити про це

свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.5 Дослідження інформаційного навантаження на програміста

Програміст, у залежності від підготовки і досвіду, вирішує задачі різної складності, але в загальному випадку робота програміста будується по наступному алгоритму:

Таблиця 8.1 – Алгоритм

Етап	Зміст	Витрата часу, %
III	Постановка задачі Вивчення матеріалу за поставленою задачею	6.25
III	Визначення методу рішення задачі	6.25
IV	Складання алгоритму рішення задачі	12.5
V	Програмування	25
VI	Налагодження програми, складання звіту	50

Даний алгоритм відображає загальні дії програміста при рішенні поставленої задачі незалежно від її складності.

Таблиця 8.2 – Дії програміста

Етап	Член алгоритму	Зміст роботи	Літерне позначення
1	2	3	4
I	1	Одержання першого варіанта технічного	A
	7	Складання і уточнення технічного завдання	Ri
		Одержання остаточного варіанта технічного завдання	Сіл
	4	Складання переліку матеріалів, що існують	НІІ2
	5	Вивчення матеріалів по тематиці задачі	Аг

Продовження таблиці 8.2

1	2	3	4
	6	Вибір метолу рішення	C2J3
	7	Уточнення й узгодження обраного методу	B2
	8	Остаточний вибір методу рішення	СчТ4
	9	Аналіз вхідної і вихідної інформації	B2
	10	Вибір мови програмування	C4TS
	11	Визначення структури програми	H3C5q
	12	Складання блок-схеми програми	C6q2
	13	Логічний аналіз програми і коректування її	F1H4W2
	14	Компіляція програми	F2
	15	Виправлення помилок	D1W2
	16	Редагування програми	F2H5B3W
	17	Виконання програми	F3
	18	Аналіз результатів виконання	H6W5
	19	Тестування	C7W6
	20	Підготовка звіту про роботу	F4

Підрахуємо кількість членів алгоритму і їхню частоту (імовірність) щодо загального числа, прийнятого за одиницю. Імовірність повторення і-ої ситуації визначається по формулі:

$$Pi = к/п, \quad (8.1)$$

де к – кількість повторень кожного елемента одного типу, п – сумарна кількість повторень від джерела інформації, одного типу.

Результати розрахунку зведемо в таблицю:

Таблиця 8.3 – Результати розрахунку

Джерело	Члени алгоритму	Символ	Кількість	Частота повторень
1	2	3	4	5
1	Аферентні – усього		6	1,00
	Вивчення технічної	A	2	0,33
	Спостереження	P	4	0,67

Продовження таблиці 8.3

1	2	3	4	5
2	Еферентні – усього		18	1,00
	Уточнення	В	3	0,17
	Вибір найкращого	С	8	0,44
	Виправлення	0	1	0.06
	Аналіз отриманих	н	6	0,33
	Виконання	к	0	0
3	Логічні умови		13	1,00
	Прийняття рішень на основі вивчення	І	5	0,39
	Грабічні матеріали	Ч	2	0.15
	Отриманого тексту	V	6	0.46
	Усього:		37	

Кількісні характеристики (Табл. 8.6) дозволяють розрахувати інформаційне навантаження програміста [8]. Ентропія інформації елементів кожного джерела інформації розраховується по формулі, біт/сигн:

$$H_j = \sum_{i=1}^{m} p_i \log_2 p_i, \qquad (8.2)$$

де т – число однотипних членів алгоритму розглянутого джерела інформації.

$$H1=2 \times 2+2 \times 4 = 12$$

$$H2 = 3 \times 1,585 +8 \times 3+0+6 \times 2,585 = 44,265$$

$$H3 = 5 \times 2,323 + 2 \times 1+6 + 2,585 = 29,125$$

Потім визначається загальна ентропія інформації, біт/сигн:

$$Hs=Hl+H2+H3, \qquad (8.3)$$

де HI, H2, H3 – ентропія аферентних, еферентних елементів і логічних умов відповідно.

Hs = 10 + 44,265 + 29,125 = 83,39. Далі визначається потік інформаційного навантаження біт/хв,

$$F = \frac{H_{\sum} \cdot N}{t}, \quad (8.4)$$

де N – сумарне число всіх членів алгоритму; I – тривалість виконання всієї роботи, хв.

Від кожного джерела в інформації (члена алгоритму) у середньому надходить 3 інформаційних сигнали в годину, час роботи – 225 годин,

$$\Phi = \frac{83,39 \cdot 37 \cdot 3 \cdot 225}{13500} = 2,6 \text{ біт/с}$$

Розраховане інформаційне навантаження порівнюється з припустимою. При необхідності приймається рішення про зміни в трудовому процесі.

Умови нормальної роботи виконуються при дотриманні співвідношення:

$$\Phi_{доп.мин} < \Phi_{расч} < \Phi_{доп.макс}, \quad (8.5)$$

де $\Phi_{доп.мін}$, і $\Phi_{доп.макс}$ мінімальний і максимальний припустимі рівні інформаційних навантажень (0,8 і 3,2 біт/з відповідно);

$\Phi_{расч}$ – розрахункове інформаційне навантаження 0,8 < 2,6 <3,2

8.6 Висновки до розділу

У цій частині магістерської проекту були розглянуті вплив факторів трудового і виробничого середовища програмістів, дослідження інформаційного навантаження на програміста. Дотримання умов, що визначають оптимальну організацію робочого місця програміста і навантаження, отримані ним в процесі роботи, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить, як у кількісному, так і в якісному відносинах продуктивність праці програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи обміну інформацією у мережі на основі протоколу Signal.

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 обміну інформацією у мережі на основі протоколу Signal.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем обміну інформацією у мережі на основі протоколу Signal.

– Досліджена система обміну інформацією у мережі на основі протоколу Signal.

– На основі отриманих результатів досліджень створена програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання обміну інформацією у мережі на основі протоколу Signal.

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 Embarcadero Delphi. 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 CRYPTON.

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 7022 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,27 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кострик В.О. Дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Will Grant. 101 UX Principles. Packt Publishing. 2022. 432 p.

3. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

4. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

5. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

6. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

7. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

8. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

9. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

10. 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.

11. 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.

12. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph.. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

13. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

14. 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.

15. 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,

16. 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.

17. 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

18. 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.

19. 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.

20. 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.

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

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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).

47. 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

48. 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.

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

50. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи обміну інформацією у мережі на основі протоколу Signal.

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. № 32-13 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 обміну інформацією у мережі на основі протоколу Signal.

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 обміну інформацією у мережі на основі протоколу Signal;

– 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 Embarcadero Delphi.

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 пожежна безпека.

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 – 108 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 11.12.2023 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

_____ Коваленко А.С.

Дослідження та програмна реалізація системи обміну інформацією у мережі на основі протоколу Signal

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2023 року

Додаток Б

(обов’язковий)

Міністерство освіти і науки, молоді та спорту України

Кіровоградський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник дипломного проекту

_____ Коваленко А.С.

Програмне забезпечення обміну інформацією у мережі на основі протоколу Signal

Лістинг програми

Код документу 12

Носій: дискета

Загальна кількість аркушів: 53

Літера: РП

Кропивницький 2023

Клієнтська частина
Unit_Signal_OSCAR_RL.pas – основна частина клієнтського додатку

```
unit Unit_Signal_OSCAR_RL;

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_Signal_OSCAR_RL = class(TForm)
    ClientSocket: TClientSocket;
    Memo_Signal_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;
  _Signal_OSCAR_RL: T_Signal_OSCAR_RL;
  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, _Signal_OSCAR_HauteurDeDebut, _Signal_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_Signal_OSCAR_RL.RendreVisiblePremierPlan;
begin
  if (ChoixCouleur.RadioGroupVisible.ItemIndex = 0) then
  begin
    Application.Restore;
    Application.BringToFront;
  end;
end;

function T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.Minimize;
begin
  mvtFenetre(1);
  _Signal_OSCAR_RL.Visible := False;
end;


//створення форми

 procedure T_Signal_OSCAR_RL.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');
  _Signal_OSCAR_RL.Height := Round(500 * (Screen.Width / 1024) / 1.3);
  _Signal_OSCAR_RL.Width := Round(300 * (Screen.height / 768) / 1.3);
  _Signal_OSCAR_HauteurDeDebut := _Signal_OSCAR_RL.Height;
  _Signal_OSCAR_LargeurDeDebut := _Signal_OSCAR_RL.Width;
  ChoixCouleur := TChoixCouleur.Create(_Signal_OSCAR_RL);
  CAudioVideo := TCAudioVideo.Create(_Signal_OSCAR_RL);
  Randomize;
  NombreSecret := RandomRange(1000, 9999);
  _Signal_OSCAR_RL.Left := screen.Width - _Signal_OSCAR_RL.Width;
  _Signal_OSCAR_RL.top := screen.height - _Signal_OSCAR_RL.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);
  _Signal_OSCAR_RL.Visible := true;
  mvtFenetre(-1);
   _Signal_OSCAR_RL.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_Signal_OSCAR_RL.WM_CALLBACKPRO(var msg: TMessage);
begin

  case msg.LParam of WM_LBUTTONDOWN:
      begin
        if _Signal_OSCAR_RL.Visible = true then
```

```
        begin
          mvtFenetre(1);
          _Signal_OSCAR_RL.Visible := False;
        end
        else
        begin
          mvt := 1;
          _Signal_OSCAR_RL.Visible := True;
          mvtFenetre(-1);
        end;
      end;
    WM_RBUTTONDOWN: traymenu.Popup(mouse.CursorPos.X, mouse.CursorPos.y);
  end;
end;

procedure T_Signal_OSCAR_RL.mvtFenetre(i: Integer);
var
  k, fin, offset: integer;
begin
  offset := screen.Height - GetSystemMetrics(SM_CYFULLSCREEN);
  mvt := 1;
  _Signal_OSCAR_RL.Left := screen.Width - _Signal_OSCAR_RL.Width;
  if (i < 0) then
    fin := screen.Height - _Signal_OSCAR_RL.Height - offset
  else
    fin := screen.Height;
  while (_Signal_OSCAR_RL.Top <> fin) do
  begin
    _Signal_OSCAR_RL.Top := _Signal_OSCAR_RL.Top + i;
    for k := 1 to 1000000 do
      mvt := 1;
  end;
  if (i < 0) then
    _Signal_OSCAR_RL.Top := screen.Height - _Signal_OSCAR_RL.Height - offset
  else
    _Signal_OSCAR_RL.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_Signal_OSCAR_RL.TimerNombresClientsActuelsTimer(Sender: TObject);
begin
  application.ProcessMessages;
end;

//Час перез'єднання

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  ActionFermerExecute(Sender);
end;

//З'єднання з сокетом клієнта

 procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Application.ProcessMessages;
end;

//Помилка у сокеті клієнта

 procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.Action10Execute(Sender: TObject);
begin
  CAudioVideo.Show;
end;

procedure T_Signal_OSCAR_RL.Action13Execute(Sender: TObject);
begin
  ChoixCouleur.Show;
end;

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.ActionFermerExecute(Sender: TObject);
begin
  Shell_NotifyIcon(Nim_DELETE, @trayicon);
  Fermer1Click(_Signal_OSCAR_RL);
end;

procedure T_Signal_OSCAR_RL.ActionAjouterGroupeExecute(Sender: TObject);
begin
  ClientDataSet3.Insert;
  GroupeNB := GroupeNB + 1;
  ClientDataSet3N_ID.Value := GroupeNB;
  ClientDataSet3GROUPE.Value := 'Groupe Temp';
  ClientDataSet3.Post;
end;

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.ActionQuitterExecute(Sender: TObject);
begin
  Shell_NotifyIcon(Nim_DELETE, @trayicon);
  Fermer1Click(_Signal_OSCAR_RL);
end;

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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#';
        _Signal_OSCAR_RL.ClientSocket.Socket.SendText(#13 + MessageStatut);
      end;
end;

procedure T_Signal_OSCAR_RL.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(_Signal_OSCAR_RL);
        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_Signal_OSCAR_RL.MessageInfo: string;
var
  i: integer;
  MessageInfoListe_Signal_OSCAR_eur, MessageInfoListe_Signal_OSCAR_eur1: string;
begin
  Application.ProcessMessages;
  MessageInfoListe_Signal_OSCAR_eur1 := '';
  MessageInfoListe_Signal_OSCAR_eur := '';
  MessageInfoListe_Signal_OSCAR_eur := '@щ*@';
  for i := 1 to NumeroArriveConnexion + 3 do
  begin
    if StructureOrdinateur[i].LoginConnecte <> '' then
    begin
      MessageInfoListe_Signal_OSCAR_eur1 := MessageInfoListe_Signal_OSCAR_eur1 +
'µ' + StructureOrdinateur[i].LoginConnecte
        + '#IMG#' + StructureOrdinateur[i].img + '#ORDI#' +
StructureOrdinateur[i].NomOrdinateur;
    end;
  end;

  if MessageInfoListe_Signal_OSCAR_eur1 <> '' then
  begin
    MessageInfoListe_Signal_OSCAR_eur := MessageInfoListe_Signal_OSCAR_eur +
MessageInfoListe_Signal_OSCAR_eur1 + 'µ¤@*!@¤';
    result := MessageInfoListe_Signal_OSCAR_eur;
  end
  else
  begin
    result := '??';
  end;
end;

//Читання з сокету клієнта

 procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_.Lines.Add(MessageRecuClient + '   ' +
QuelHeureEstIl);
      end
      else
      begin
        AlertMsgBox('Роз'єднання', MessageRecuClient, 0, false, 1000, 10, nil);
        Memo_Signal_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_Signal_OSCAR_.Lines.Count > 10) then
        Memo_Signal_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(_Signal_OSCAR_RL);
            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(_Signal_OSCAR_RL);
```

```
          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_Signal_OSCAR_.Lines.Count > 10) then
        Memo_Signal_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(_Signal_OSCAR_RL);
            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(_Signal_OSCAR_RL);
          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_Signal_OSCAR_.Lines.Count > 10) then
        Memo_Signal_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(_Signal_OSCAR_RL);
            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(_Signal_OSCAR_RL);
          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_Signal_OSCAR_.Lines.Add('Сервер роз'єднаний...  ' + QuelHeureEstIl);
    Application.ProcessMessages;
    cxGrid3DBTableView1.ClearItems;

    ClientSocket.Active := FALSE;
    ClientSocket.Close;
    ClientConnecter := FALSE;
  end;
end;

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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#';
        _Signal_OSCAR_RL.ClientSocket.Socket.SendText(#13 + MessageStatut);
      end;
    end;
    DS_SOURCE.DataSet.Next;
  end;

end;

procedure T_Signal_OSCAR_RL.BitBtnDeconnexionClick(Sender: TObject);
begin
  DireQueOnSeDeconnecte;
end;

procedure T_Signal_OSCAR_RL.FormResize(Sender: TObject);
begin
  Application.OnMinimize := Minimize;
end;



initialization
  NumeroArriveConnexion := 0;

end.
```

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_Signal_OSCAR_RL, 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',  _Signal_OSCAR_RL.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);
  _Signal_OSCAR_RL.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
   _Signal_OSCAR_RL.sd1.SkinFile:=path+Couleur.Text;
   if not _Signal_OSCAR_RL.sd1.Active then _Signal_OSCAR_RL.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.
```

Серверна частина
Unit_Signal_OSCAR_RL.pas – основна частина серверного додатку

```
unit Unit_Signal_OSCAR_RL;

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_Signal_OSCAR_RL = class(TForm)
    PageControl: TPageControl;
    TabSheetServeur: TTabSheet;
    GroupBox1: TGroupBox;
    LabelIP: TLabel;
    ServerSocket: TServerSocket;
    TimerInformations_Signal_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_Signal_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
   _Signal_OSCAR_RL: T_Signal_OSCAR_RL;
  EditPortServeur : string;
  StructureOrdinateur,StructureOrdinateur002 :array[1..52] of
TStructureListeConnecte;

ComboBoxNombresClients,NumeroArriveConnexion,NombresMaximumClients,NombreSecret
: integer;
  ServeurActif,ClientConnecter : boolean;
  Present: TDateTime;
  Hour, Min, Sec,MSec: Word;

_Signal_OSCAR_HauteurDeDebut,_Signal_OSCAR_LargeurDeDebut,PageHauteurDebut,PageL
argeurDebut : 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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.TabSheetServeurShow(Sender: TObject);
begin
        LabelIP.Caption := TrouverIP(NomPcActuel);
        GroupBox1.Caption := 'Вибрати сервер: '+ NomPcActuel;
        ComboBoxNombresClients := 999;
end;

//Створення форми чату


//створення форми

 procedure T_Signal_OSCAR_RL.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);

  _Signal_OSCAR_RL.Left := screen.Width - _Signal_OSCAR_RL.Width ;
  _Signal_OSCAR_RL.top := screen.height - _Signal_OSCAR_RL.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);

  _Signal_OSCAR_RL.Visible := true;

  _Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.ServerSocketClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
     inc(NumeroArriveConnexion);
     TimerInformations_Signal_OSCAR_eur.Enabled := TRUE;
end;

//Закриття форми


//Закриття форми

 procedure T_Signal_OSCAR_RL.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
   Fermer1Click(_Signal_OSCAR_RL);
end;

//Запуск та відключення серверу

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_eur.Enabled := FALSE;
          TimerInformations_Signal_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_Signal_OSCAR_eur.Enabled := FALSE;
          TimerInformations_Signal_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_Signal_OSCAR_RL.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
      dec(NumeroArriveConnexion);
      if NumeroArriveConnexion = 0 then
      begin
        TimerInformations_Signal_OSCAR_eur.Enabled := FALSE;
        TimerInformations_Signal_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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.TimerInformations_Signal_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_Signal_OSCAR_RL.MessageInfo : string;
var
    i : integer;
    MessageInfoListe_Signal_OSCAR_eur,MessageInfoListe_Signal_OSCAR_eur1 :
string;
```

```
begin
     Application.ProcessMessages;
     MessageInfoListe_Signal_OSCAR_eur1 := '';
     MessageInfoListe_Signal_OSCAR_eur := '';
     MessageInfoListe_Signal_OSCAR_eur := '@щ*@';
     for i:=1 to NumeroArriveConnexion + 3 do
     begin
        if StructureOrdinateur[i].LoginConnecte <> '' then
        begin
            MessageInfoListe_Signal_OSCAR_eur1 :=
MessageInfoListe_Signal_OSCAR_eur1 + 'µ'
            + StructureOrdinateur[i].LoginConnecte + '#IMG#' +
StructureOrdinateur[i].img +
             '#ORDI#'+StructureOrdinateur[i].NomOrdinateur;
        end;
     end;

     if MessageInfoListe_Signal_OSCAR_eur1 <> '' then
     begin
       MessageInfoListe_Signal_OSCAR_eur := MessageInfoListe_Signal_OSCAR_eur +
MessageInfoListe_Signal_OSCAR_eur1 + 'µ¤@*!@¤';
       result:= MessageInfoListe_Signal_OSCAR_eur;
     end
     else
     begin
         result:= '??';
     end;

end;

//Виведення імені користувача, з яким спілкуємся

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.PageControlChanging(Sender: TObject;
  var AllowChange: Boolean);
begin
  if ((Sender as TPageControl).ActivePage = TabSheetServeur) and (ServeurActif =
TRUE) then
     AllowChange := FALSE
end;

procedure T_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL.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_Signal_OSCAR_RL;


{$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;
     _Signal_OSCAR_RL.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_Signal_OSCAR_RL;

{$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#';
    _Signal_OSCAR_RL.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#';
  _Signal_OSCAR_RL.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#';
      _Signal_OSCAR_RL.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_Signal_OSCAR_RL, 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 обміну інформацією
у мережі на основі протоколу Signal');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Коваленко А.С.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент  Кострик Владислав Олександрович');
Memo1.Lines.Add('                   гр. КН-22М-1');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Кропивницький 2023');
Memo1.Lines.Add('');

end;

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```