МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Проєкт

з курсу «Комп'ютерні мережі»

Тема: надійний месенджер для локальної мережі

Виконав: студент групи КІ-32

Ільїн С. А.

Викладач: Кухарєв С. О.

Київ – 2025р.

ЗМІСТ

Загальний опис проєкту

Назва: Local mail

Посилання на репозиторій: github.com/njnjfnj/Local_Mail

Призначення: Месенджер з відкритим вихідним кодом для спілкування з іншими пристроями в локальній дротовій/бездротовій мережі за допомогою захищених peer-to-peer повідомлень.

Предметна область: Великі корпоративні компанії, виробництва та інші, хто потребує надійне безкоштовне віддалене спілкування на великій території та можливість переробити месенджер під себе.

1. Перелік основних впроваджених функцій

Представимо основні впроваджені функції у формі блок-схеми на рисунку 1.1.

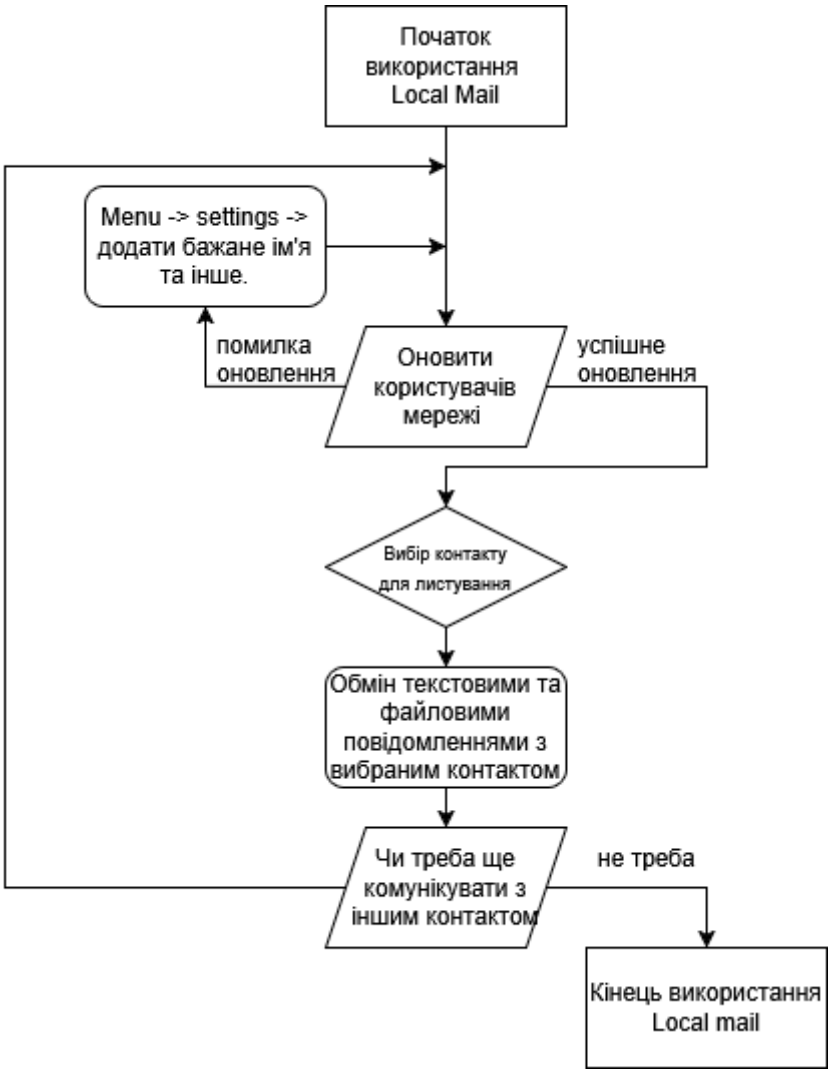

Рисунок 1.1 - Блок схема використання месенджера.

Треба зауважити, що на рисунку 1.1 для початку листування з іншим контактом обов’язково треба оновити користувачів мережі з діючим месенджером. Це через те, що якщо ви не оновите користувачів, після чого написати контакту, в якого месенджер вже не діє, то ваші повідомлення нікуди не надійдуть. Це пов’язано з відсутністю централізованої системи в вигляді сервера та бази даних, кожне повідомлення зберігається локально. На момент написання звіту, повідомлення зберігаються в оперативній пам’яті.

2. Загальний огляд інтерфейсу

На рисунках 2.1 - 2.5 представлені вигляд списку користувачів (початкової сторінки), меню, налаштувань, а також приклад листування.

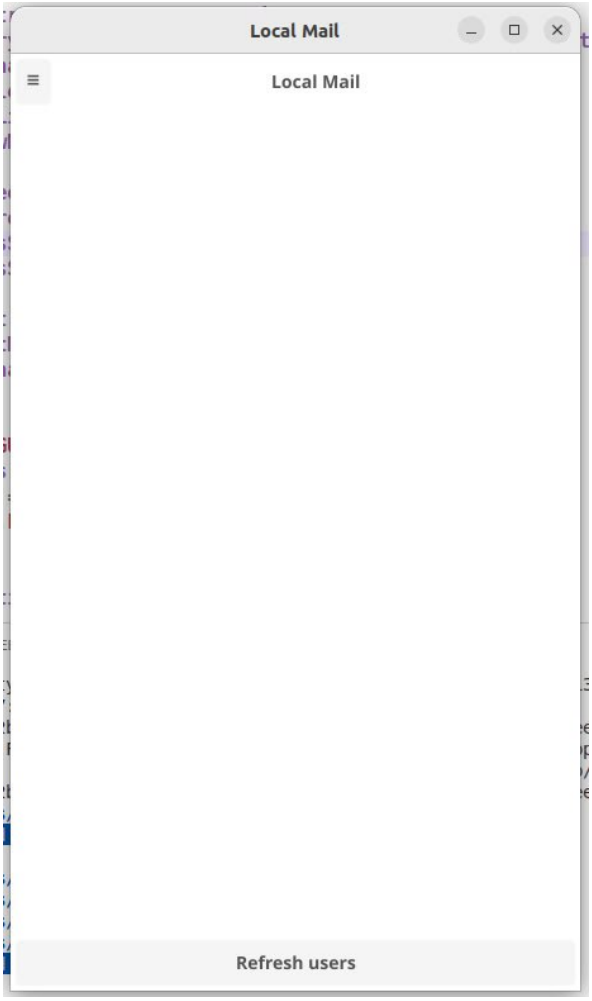

Рисунок 2.1 – пустий список контактів.

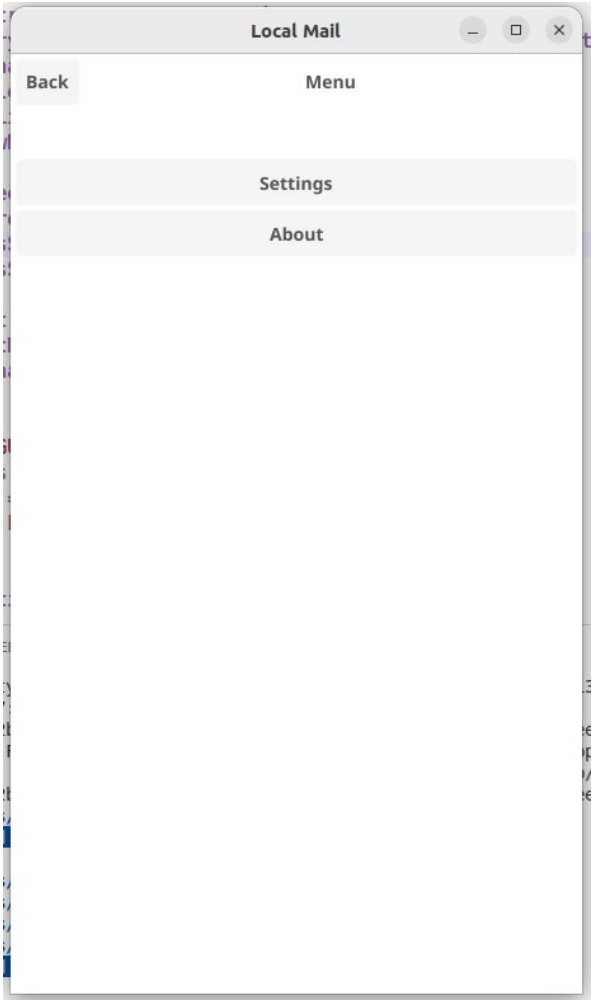

Рисунок 2.2 – меню.

Рисунок 2.3 – налаштування.

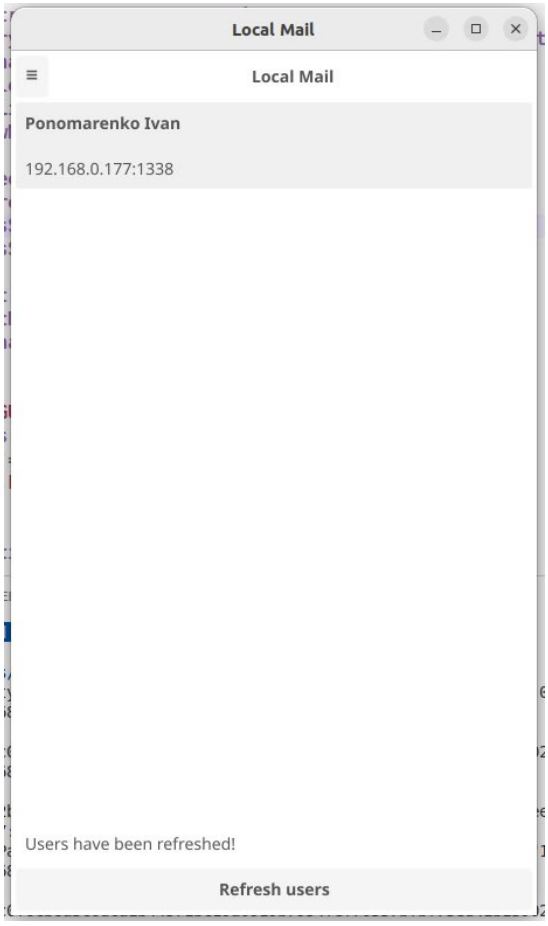

Рисунок 2.4 – оновлений список користувачів.

Рисунок 2.5 – приклад листування за допомогою текстових та файлових повідомлень.

3. Технічні аспекти

Програма була написана на мові програмування golang з використанням вбудованих бібліотек та фреймвороку fyne для графічного інтерфейсу.

Для мережевого спілкування були використані протоколи udp та tls, які потрібні для оновлення контактів через udp broadcast та захищені peer-to-peer повідомлення відповідно. Схема оновлення контактів представлена на рисунку 3.1.

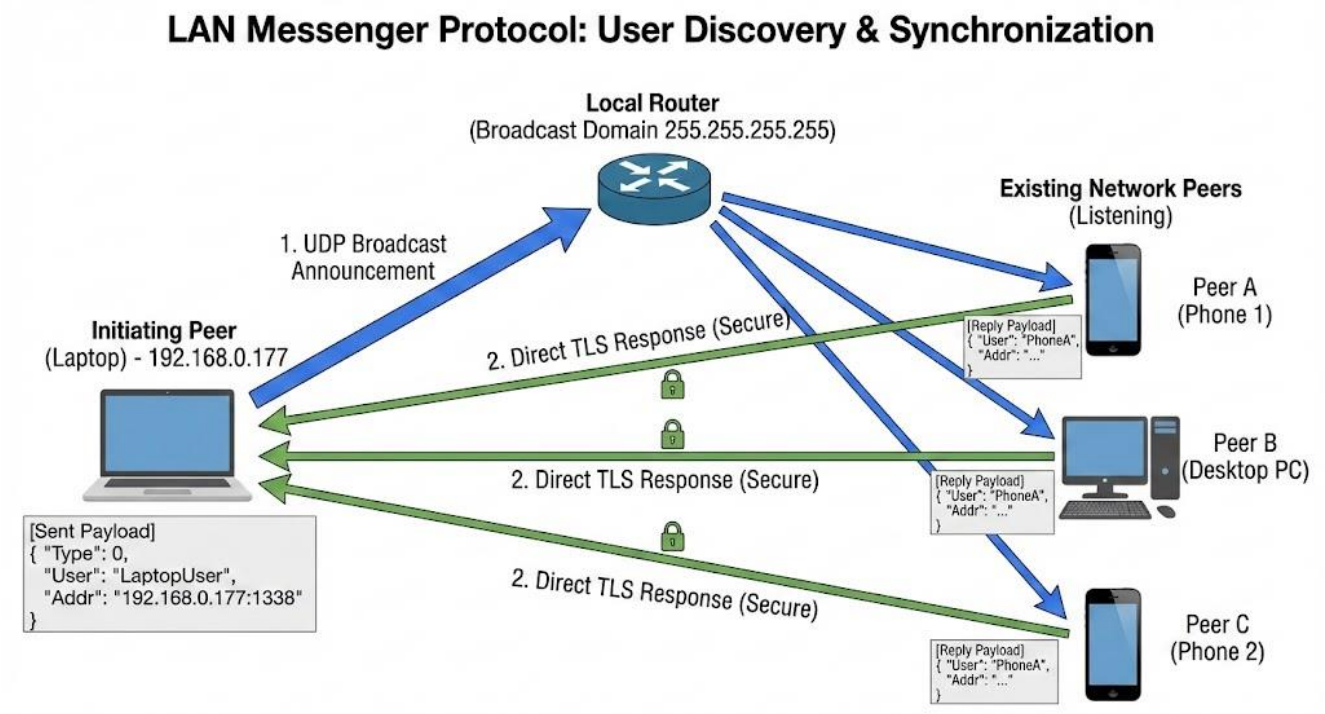

Рисунок 3.1 – схема оновлення контактів.

Як ми бачимо з рисунку 3.1, для того, щоб оновити контакти в мережі, месенджер на кожному пристрої запускає udp та tls сервери на різних портах, які можна змінити в налаштуваннях (див. рис. 2.3). При натискані на кнопку оновлення, програма відправляє пакет з json, в якому написані дані про тип пакету (для оновлення пристроїв тип пакету приймає значення нуль, або ж PackageTypeHandshake), ім’я користувача та його повний адрес з локальним IP та портом tls сервера. Інші користувачі приймають цей пакет, додають до себе цього користувача з його контактною інформацією та надсилають цьому користувачеві tls повідомлення вже з своїми контактними даними також через подібний json.

Користувач, який оновлює користувачів, приймає ці tls повідомлення та також зберігає в себе цих користувачів.

Після отримання активних користувачів, користувач вже може переходити до самого листування. Схема листування текстовими та файловими повідомленнями представлена на рисунку 3.2.

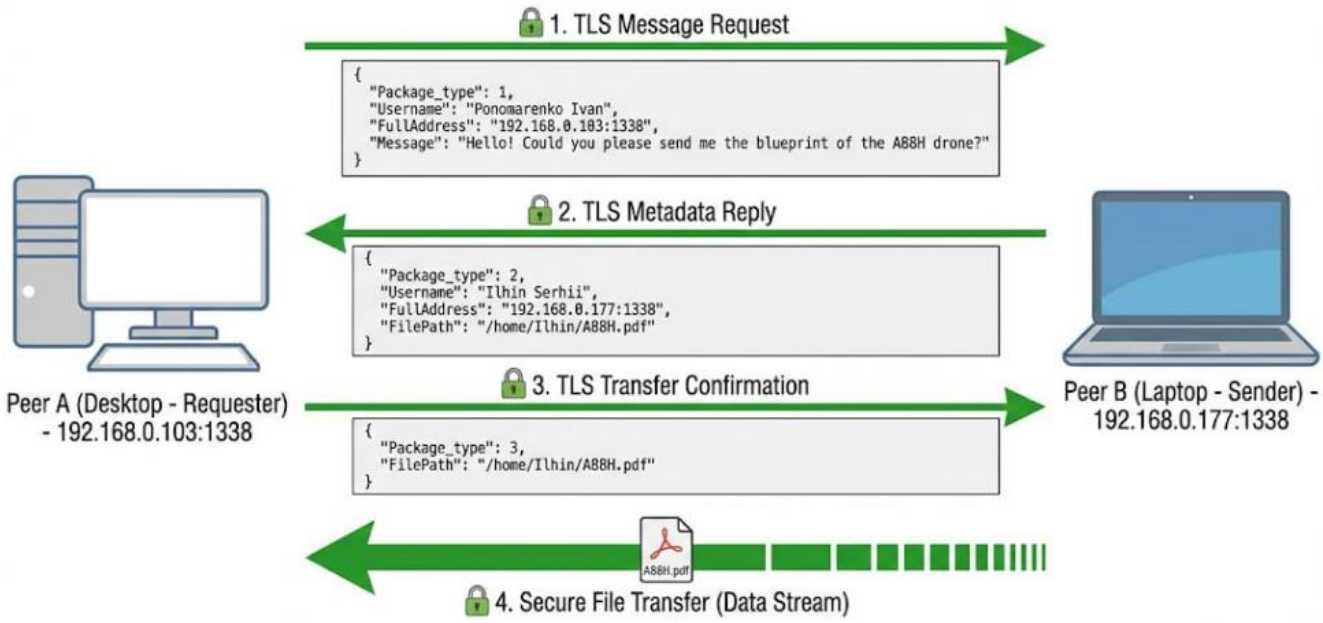

Рисунок 3.2 – схема листування захищеними текстовими та файловими повідомленнями через peer-to-peer tls.

Розберемно схему на рисунку 3.2:

- Для відправки текстового повідомлення користувачеві Б, користувач А має надіслати p2p tls пакет з json, в якому буде інформація про повну адресу користувача А, його ім'я, саме текстове повідомлення та тип пакету, яке рівне 1 (PackageTypeMessage). Користувач Б зчитає цей json та збереже це повідомлення локально, висвітлить його в чаті.
- Для відправки файлового повідомлення користувачеві А, кристувач Б надсилає контактні дані самого файлу (шлях до файлу на пристрої користувача Б). Це відбувається через надсилання p2p tls пакету з json, в якому буде інформація про повну адресу користувача Б, його ім'я, контактна інформація цільового файлу та тип пакету, яке рівне 2 (PackageTypeSendFileInfo). Користувач А приймає це повідомлення, зберігає його в спеціальному типі даних File_type, яке є кнопкою

графічного інтерфейсу з функцією завантаження самого файлу. Ця кнопка зберігає в собі повну адресу користувача Б, шлях цільового файлу на пристрої користувача Б та службові поля.

- При натисканні на кнопку завантаження файлу користувачем А, він відправляє користувачеві Б наступне p2p tls пакет з json, в якому зберігається інформація про контактні дані бажаного файлу користувача Б та тип пакету, значення якого рівне 3 (PackageTypeFileReq). Після відправки повідомлення, користувач А очікує відправку файлу користувачем Б в новому потоці. Користувач Б, в свою чергу, відповідає на пакет користувача А відправкою самого файлу користувачеві А через теж саме p2p tls з'єднання.

В коді відправка та зчитування файлу реалізована через вбудований метод io.Copy(fileObject, tlsConnection) в мові програмування golang. Для відправки спочатку він розбиває файл на TLS-рекорди (блоки), шифрує ці блоки узгодженим сертифікатом, додає код аутентифікації повідомлення MAC для цілісності файлу, щоб умовний хакер не міг змінити значення хешу блоків файлу, та відправляє ці зашифровані блоки. Отримувач чекає зашифровані блоки, розшифровує отримані TLS-рекорди узгодженим сертифікатом, перевіряє файл на цілісність через MAC, записує файл лоакльно на пристрої отримувача.

4. Зауваження до технічних аспектів

Через децентралізовану специфіку додатку слід роз’яснити процес передачі tls пакетів. На початку треба зазначити, що в області кібербезпеки фігурує так звана модель CIA. Вона забезпечує наступні три принципа:

- Confidentiality. Ніхто сторонній не може прочитати повідомлення.
- Integrity. Дані не були змінені в дорозі.
- Availability. Сервіс доступний для легітимних користувачів.

Розберемо як реалізується кожен принцип.

Confidentiality. Під час проходження даних через conn.Write або json.NewEncoder(conn) (golang методів передачі пакетів), вони шифруються узгодженим обома сторонами симетричним ключем X509 (є стандартом для цифрових сертифікатів). Саме з’єднання розділяється на наступні етапи:

- TCP Handshake (Транспортний рівень). Перш ніж TLS почне роботу, операційна система встановлює TCP-з'єднання. Спочатку клієнт надсилає на порт отримувача запит SYN, сервер надає SYN-ACK відповідь, клієнт надсилає ACK та відкриває поки незахищений канал з сервером.
- TLS Handshake (Рівень сеансу). Клієнт надає серверу інформацію про присутні типи шифрування, сервер обирає алгоритм та відправляє свій самопідписаний сертифікат клієнту. Після отримання клієнт відправляє свій самопідписаний сертифікат серверу. Сторони використовують асиметричну криптографію (RSA з сертифіката), щоб безпечно створити спільний сесійний ключ. Сторони перемикаються на симетричне шифрування. Тунель готовий.

Integrity. Цілісність пакетів гарантується механізмом MAC (про який писалось вище), що вбудований у TLS. Якщо біт зміниться, conn.Read поверне помилку. Також наші передаваємі пакети використовують хешування (sha256.Sum256(peerCert.Raw)), щоб перевірити незмінність сертифіката піра (fingerprinting).

Availability. Через децентралізовану систему при надсиланні пакетів в коді відсутня явна перевірка на справжній сервер, що є вразливістю до активної атаки «Людина посередині» (MitM). Це виправляє сама структура листування та графічне представлення користувачів: відправник явно бачить якому адресу він відправляє повідомлення, тому при подібній атаці відправник маловірогідно відправить своє повідомлення не туди. На момент написання звіту в tls сервері месенджеру відсутній захист ddos атак, що провокує переповнення оперативної пам'яті (зависання та екстрене завершення програми) та відсутні таймаути надсилання пакетів, через що зловмисник може надіслати неймовірну кількість пакетів, які будуть надсилатись, наприклад, 1 байт на хвилину, що також провокує

переповнення пам’яті. Звісно, зловмисник не зможе застосувати такі методи атаки, якщо сама мережа буде захищена, але в майбутньому ці вразливості будуть обов’язково виправлені, тому в межах цієї роботи для зручності оформлення висновків будемо вважати, що наведені вище вразливості є полагодженими.

Отже, наш месенджер Local Mail повністю реалізує CIA модель, а отже є повністю захищеним, надійним, відповідає бажаним стандартам цільової аудиторії, яка наведена в предметній області загального опису проєкту.

5. Код проекту

Структура проєкту в вигляді файлового каталогу представлена на рисунках 5.1 та 5.2.

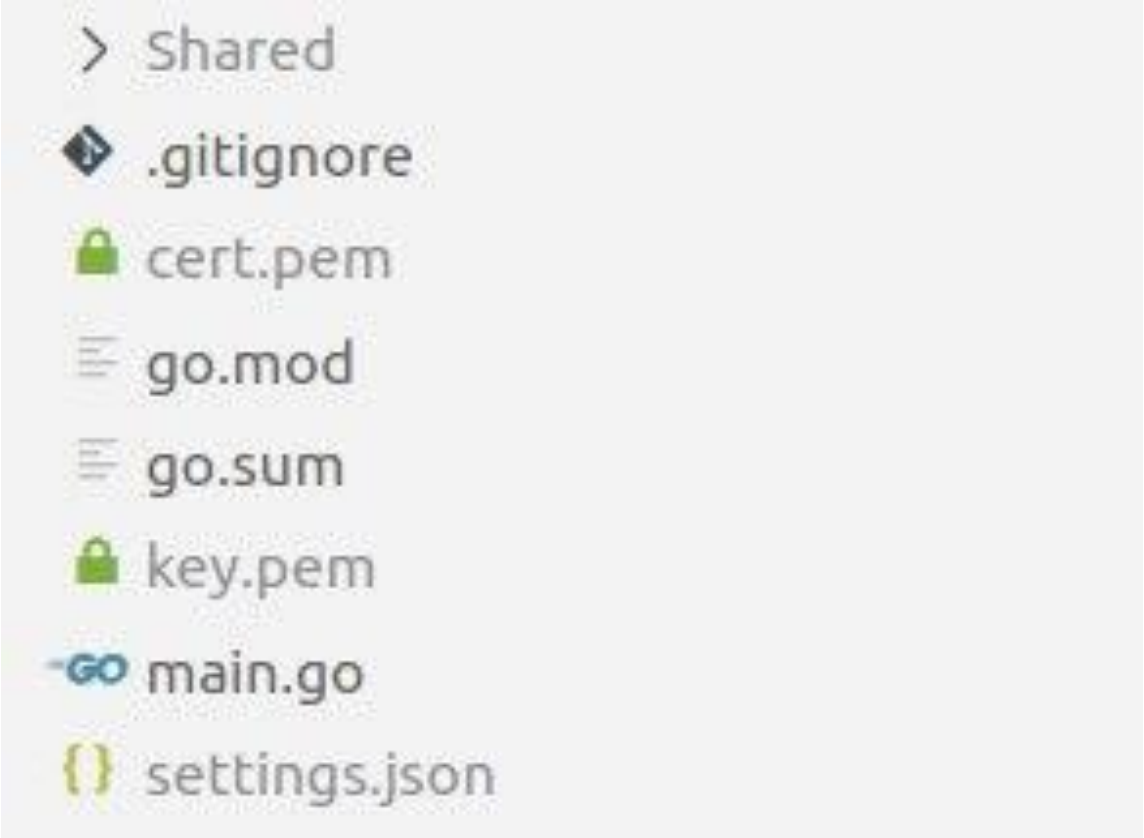

Рисунок 5.1 – представлення положення головного файлу та решти службових файлів проєкту в кореневому каталозі.

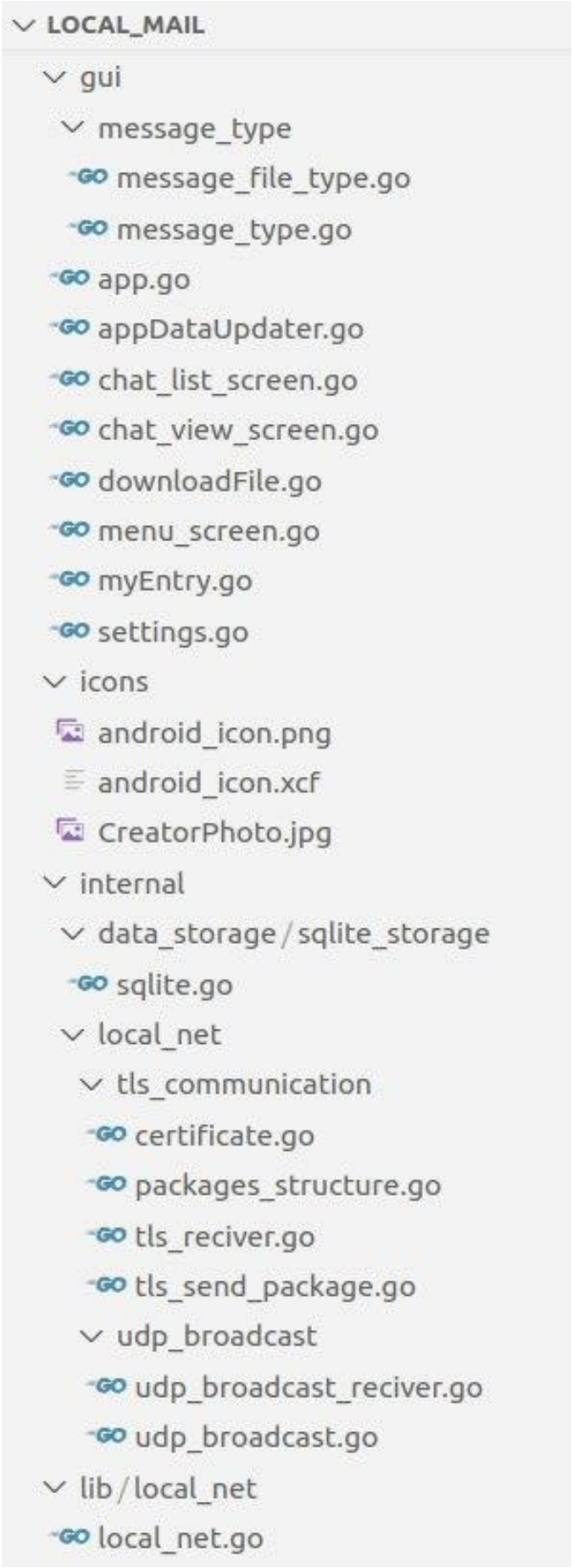

Рисунок 5.2 – представлення головного коду проєкту в вигляді файлового каталогу.

Проєкт структурований за класичним розподіленням по пакетах в мові програмування golang. В цій мові кожна папка є окремим пакетом.

Розпишемо по пунктам яка папка за що відповідає:

- Корінна папка відповідає за збереження всього проєкту, тут можна знайти точку запуску проєкту файл main.go, конфігураційні golang файли go.sum та go.mod, самопідписані сертифікати key.pem та cert.pem, файл налаштувань settings.json.
- Папка icons зберігає в собі іконки та фотографії.
- Папка share зберігає в собі файли, які надсилались другим користувачам.
- Пакет gui реалізує всю логіку та представлення графічного інтерфейсу.
- Пакет message_type описує структуру одного повідомлення для подальшого його використання в списках чатів.
- Пакет local_net зберігає в собі метод знаходження локальної адреси користувача.
- Пакет tls_communication відповідає за реалізацію та логіку всього tls спілкування.
- Пакет udp_broadcast відповідає за реалізацію та логіку оновлення користувачів.
- Пакет sqlite_storage є незавершеною реалізацією довгострокового зберігання повідомлень.

Повний код проєкту представлений в публічному репозиторії за посиланням [1].

Пропозиції щодо подальших покращень

Ось кілька пропозицій покращень для месенджеру Local Mail:

1. Виправлення аспекту Availability додатку, а саме реалізувати захист від ddos та Slow Read кібератак.
2. Покращити ui/ux додатку, а саме додати можливість встановлення аватарів, пункту «про себе», інші характеристики для додаткової ідентифікації користувача. Також слід адаптувати інтерфейс під телефони, а саме збільшити розмір кнопок, додати керування жестами і так далі.
3. Додати більше можливостей при передачі повідомлень: картинки, гіфки, стікери, тощо.
4. Додавання більшого функціоналу: створення груп, тимчасові повідомлення, локальне розвернення ботів на пристрої користувача для інших користувачів, довгострокове збереження повідомлень, тощо.
5. Рефакторинг коду.
6. Оформлення гітхаб репозиторію, а саме додати інструкції по експлуатації, локальній збірці проєкту, додати документацію по коду та висвітлити принцип роботи додатку.
7. Реліз проєкту на платформах windows, linux, android, можливо mac.

Висновок:

В результаті виконання проєкту був створений надійний месенджер для локальної мережі під назвою Local Mail за допомогою мови програмування golang та фреймворку Fyne, який виконує роль передачі захищених повідомлень.

У процесі розробки було проведено безліч тренувальних сесій за допомогою трьох пристроїв, у результаті яких була продемонстрована коректна робота додатку.

Отже, «Local Mail» – це підходяще безкоштовне рішення з відкритим вихідним кодом для співробітників компаній, підприємств, угрупувань різного рівня секретності та просто для ентузіастів в сфері кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Посилання на репозиторій проєкту:
https://github.com/njnjfnj/Local_Mail
2) Посилання на мій гітхаб акаунт:
https://github.com/njnjfnj