uChat Project:

Повна Технічна Енциклопедія Проекту

Версія: 2.1

Технологічний стек: C# .NET 8, WinUI 3, SQLite, TCP/IP.

By Alexey

Зміст

🧬 Частина 1: Архітектура та Протокол (Core)
Це "фундамент" системи. Файли ідентичні на клієнті та сервері.

1. Протокол Передачі (TCP Layer)
Використовується кастомний протокол поверх TCP для вирішення проблеми "склеювання" пакетів (TCP Stream).

- **Формат пакету:** [Header: 4 bytes (Length)] + [Body: JSON Bytes].
- **Логіка читання:**
    1. Зчитати перші 4 байти -> конвертувати в int (це розмір повідомлення).
    2. Зчитати рівно стільки байт, скільки вказано в розмірі.
    3. Десеріалізувати байти в об'єкт ProtocolMessage.

2. Структура Даних (ProtocolMessage.cs)
Головний клас-контейнер для обміну даними.

- **Type** (MessageType enum): Тип команди.
    - *Приклади:* Login, Register, SendMessage, SendImage, CreateGroup, Heartbeat.
- **Data** (string?): Основне навантаження.
    - *Текст:* "Привіт, як справи?"
    - *Файл/Зображення:* Base64 рядок.
    - *Списки:* JSON масив (наприклад, список юзерів [{"Id":1, "Username":"Alex"}]).
- **Parameters** (Dictionary<string, string>?): Метадані.
    - *Ключі:* sender, targetUsername, messageId, filename, timestamp, groupId, isPrivate.

💻 Частина 2: Серверна Частина (uchat_server)

1. Program.cs & Server.cs

- **Program.cs**: Точка входу. Зчитує порт (1337). Запускає основний сервер та фоновий потік планувальника.
- **Server.cs**:
  - **_clients**: List<ClientHandler> — список активних підключень.
  - **TcpListener**: Слухає порт. При новому підключенні створює ClientHandler і запускає його в Task.Run.
  - **BroadcastToAll**: Серіалізує повідомлення один раз і розсилає всім у списку _clients.
  - **CheckAndSendScheduledMessagesAsync**: Фоновий процес (Loop). Кожні 10 с перевіряє БД на наявність повідомлень, час яких настав.

2. ClientHandler.cs (Логіка обробки)

Один екземпляр на кожного підключеного користувача.

- **HandleClientAsync()**: Нескінченний цикл читання потоку.
- **ProcessMessageAsync(message)**: Головний switch, що розподіляє логіку (Login -> HandleLoginAsync, SendMessage -> HandleSendMessageAsync).

3. DatabaseContext.cs (Дані)

Робота з **SQLite** (uchat.db). Використовує режим WAL (Write-Ahead Logging) для швидкодії.

- **Основні таблиці:**
  - Users: Профілі (Id, Username, PasswordHash, DisplayName, AvatarData).
  - Messages: Історія глобального чату.
  - DirectMessages: Приватні повідомлення (SenderId, RecipientId).
  - Groups / GroupMembers / GroupMessages: Структура груп.
  - ScheduledMessages: Відкладені повідомлення.
  - **FileBlobs**: Зберігає "важкі" дані (файли) окремо, щоб прискорити завантаження історії чату.
- **DeleteUser(userId)**: Реалізує "м'яке видалення" (затирає дані, залишає ID) та виконує VACUUM для стиснення файлу бази.

4. AuthenticationService.cs

- **HashPassword**: Хешування через **SHA256**.
- **Authenticate**: Перевірка пари логін/хеш.

💻 Частина 3: Клієнтська Частина (uchat)

1. MainPage.xaml.cs (UI Controller)

Керує відображенням та станом.

- **Змінні стану:**
  - `_currentChatTarget`: `string?` (`null` = Глобальний чат, інакше = `username` контакту).
  - `_currentUsername`: Логін поточного користувача.
  - `_currentDisplayName`: Відображуване ім'я.
- **Колекції (Data Binding):**
  - `_messages`: `ObservableCollection<MessageViewModel>` — список повідомлень на екрані.
  - `_usersList`: Список контактів у сайдбарі.
  - `_allUsersCache`: Повний кешований список усіх користувачів (для пошуку).

2. NetworkClient.cs

- **ConnectAsync**: Ініціалізує `TcpClient`, встановлює з'єднання та запускає два фонових процеси: для читання вхідних даних та обробки черги вихідних.
- **ReceiveMessagesAsync**: Фоновий цикл читання. Спочатку зчитує 4 байти заголовка (розмір пакету), потім зчитує тіло, десеріалізує його та викликає подію `MessageReceived`.
- **SendMessageAsync / ProcessSendQueueAsync**: Реалізує безпечну відправку через **чергу** (`ConcurrentQueue`). Повідомлення не пишуться в потік напряму, а додаються в чергу, яку послідовно обробляє окремий потік (`ProcessSendQueueAsync`), щоб уникнути конфліктів та перевантаження сокета.
- **ReconnectAsync**: Закриває поточні ресурси та автоматично пробує встановити нове з'єднання.

3. Моделі та Сервіси

- **`MessageViewModel`**: Містить логіку UI (`IsVoice`, `IsImage`, `IsOwnMessage`). Керує видимістю елементів у XAML.
- **`VoiceMessageManager`**: Записує звук через `MediaCapture`, повертає Base64.
- **`SettingsService`**: Зберігає JSON з налаштуваннями (тема, шлях завантаження).

⚙️ Частина 4: Детальний Опис Функцій (Input -> Logic -> Output)

Тут розписано, як саме працюють конкретні функції, які змінні вони приймають і що роблять.

🔐 1. Авторизація (Login)

- **Функція (Клієнт):** `AttemptConnection(bool isLogin)`
  - **Вхідні змінні:** `UsernameTextBox.Text`, `PasswordBox.Password`.
  - **Логіка:**
    1. Створює з'єднання через `_networkClient.ConnectAsync`.
    2. Формує пакет: `Type = Login`, `Parameters = { username, password }`.
    3. Відправляє на сервер.
- **Функція (Сервер):** `HandleLoginAsync`
  - **Вхідні дані:** `message.Parameters["username"]`, `message.Parameters["password"]`.
  - **Логіка:**
    4. `_authService.Authenticate(...)` хешує пароль і шукає в БД.
    5. Якщо знайдено: заповнює `_currentUser`.
  - **Вихід:** Пакет `LoginResponse` з даними профілю (`userId`, `displayName`, `avatar`).

📨 2. Відправка Повідомлення (Send)

- **Функція (Клієнт):** `SendMessage()`
  - **Вхідні змінні:** `MessageTextBox.Text`, `_currentChatTarget` (стан).
  - **Логіка:**
    1. Створює локальну `MessageViewModel` і додає в `_messages` (щоб юзер бачив миттєво).
    2. Перевіряє `_currentChatTarget`:
       - Якщо `null` (Global) -> `Type = SendMessage`.
       - Якщо група -> `Type = SendGroupMessage`, додає `groupId`.
       - Якщо юзер -> `Type = SendPrivateMessage`, додає `targetUsername`.
    3. Відправляє пакет.
- **Функція (Сервер):** `HandleSendMessageAsync` / `HandleSendPrivateMessageAsync`
  - **Вхідні дані:** `message.Data` (текст), `_currentUser`.
  - **Логіка:**

1. _db.CreateMessage(...): Записує в таблицю Messages або DirectMessages.

- **Вихід:**
  1. Global: _server.BroadcastToAll.
  2. Private: Знаходить ClientHandler отримувача і викликає SendMessageAsync тільки йому.

🗀 3. Відправка Файлів (Media)

- **Функція (Клієнт):** AttachButton_Click
  - **Вхідні дані:** Вибраний файл через FileOpenPicker.
  - **Логіка:**
    1. Читає файл у потік -> Конвертує масив байтів у Base64 string.
    2. Формує пакет SendFile (або SendImage). Тіло (Data) = Base64.
    3. Додає параметр filename.
- **Функція (Сервер):** HandleSendMediaAsync
  - **Логіка:**
    1. _db.SaveFileBlob(base64) -> Зберігає "важкі" дані в FileBlobs, повертає int BlobId (наприклад, 55).
    2. Формує текстове посилання (Ref): <FILE_REF:55|doc.pdf|1024>.
    3. Зберігає це посилання як текст повідомлення.
    4. Розсилає посилання клієнтам (сам файл не шлеться, щоб не забити канал).
- **Функція (Клієнт-Отримувач):** AddMessageToView
  - **Вхідні дані:** Текст <FILE_REF:55...>, BlobId з параметра.
  - **Логіка:**
    1. Бачить тег. Розуміє, що це файл.
    2. Якщо це картинка -> автоматично шле запит GetFileContent(55).
    3. Якщо файл -> показує кнопку "Download".

👥 4. Групи (Group Chat)

- **Функція (Клієнт):** CreateGroup_Click
  - **Вхідні дані:** Список вибраних userId з діалогу CreateGroupDialog.
  - **Логіка:** Шле пакет CreateGroup, де Data = JSON списку ID.
- **Функція (Сервер):** HandleCreateGroupAsync
  - **Логіка:** _db.CreateGroup (пише в таблиці Groups та GroupMembers).
- **Функція (Сервер):** SendToGroupMembersAsync
  - **Використання:** При SendGroupMessage.

- **Логіка:**
    1. _db.GetGroupMemberIds(groupId) -> отримує список ID учасників.
    2. Фільтрує список активних клієнтів _clients.
    3. Відправляє повідомлення тільки тим, хто в групі.

🔍 5. Пошук (Search)

- **Функція (Клієнт):** UserSearchBox_TextChanged
    - **Вхідні дані:** Текст пошуку.
    - **Логіка (Гібридна):**
        1. Спочатку фільтрує локальний кеш _allUsersCache.
        2. Якщо локально мало результатів -> запускає таймер (Debounce) і шле запит SearchUsers на сервер.
- **Функція (Сервер):** HandleSearchUsersAsync
    - **Логіка:** SQL SELECT ... WHERE Username LIKE %query%. Повертає список знайдених.

👥 6. Сповіщення (Notifications)

- **Функція (Клієнт):** ShowNotification
    - **Вхідні дані:** title (ім'я відправника), content (текст), isPrivate.
    - **Логіка:**
        1. Перевіряє налаштування SettingsService (All, Mentions Only, None).
        2. Перевіряє фокус вікна через Win32 API (IsAppInForeground). Якщо користувач зараз дивиться у цей чат — сповіщення блокується.
        3. Обрізає довгий текст (>100 символів).
        4. Використовує ToastContentBuilder для показу системного Toast-повідомлення Windows.

📨 7. Відповідь на повідомлення (Reply)

- **Функція (Клієнт - UI):** ReplyContext_Click -> EnterReplyMode
    - **Логіка:** Встановлює змінну _replyingToMessage. Показує панель ReplyPanel над полем вводу з ім'ям автора та текстом оригінального повідомлення.
- **Функція (Клієнт - Мережа):** SendMessage
    - **Логіка:** Якщо _replyingToMessage != null, додає параметр "replyToId" у ProtocolMessage.
- **Функція (Сервер - БД):** GetMessages / GetGroupMessages
    - **Логіка:** Виконує SQL LEFT JOIN таблиці повідомлень самої на себе, щоб дістати ReplySender та ReplyContent для повідомлення, яке є

відповіддю. Це дозволяє клієнту відобразити контекст без додаткових запитів.

8. Безпека та Зміна Пароля

Захист реалізовано через хешування. Коли ви реєструєтесь з паролем 12345, у базу не пишеться 12345.

1. Сервер бере 12345.
2. Проганяє через алгоритм SHA256
3. Отримує щось типу a8f5f167f44f4964e6c998dee827110c.
4. Зберігає цей набір букв.

Коли ви міняєте пароль:

1. Ви вводите старий пароль.
2. Сервер хешує його і порівнює з тим, що в базі. Якщо співпало — значить це ви.
3. Тоді він хешує новий пароль і перезаписує старий запис у базі.

9. База Даних (Скелет системи)

Ваша база даних SQLite — це набір таблиць, пов'язаних між собою:

- **Users**: Тут лежать логіни, хеші паролів (зашифровані), посилання на аватарки.
- **Messages**: Тут лежить вся історія "Глобального чату".
- **DirectMessages**: Тут лежить історія особистих переписок (хто кому писав).
- **FileBlobs**: Це "склад". Тут лежать самі файли (картинки, документи), щоб не засмічувати таблиці з повідомленнями.
- **ScheduledMessages**: Це "записник". Тут лежать повідомлення, які треба відправити в майбутньому. Сервер постійно перевіряє цю таблицю.

10. Автозбереження Файлів (Auto-Save)

- **Функція (Клієнт):** `TryAutoSaveFileAsync`
    - **Логіка:**
        1. Перевіряє налаштування `SettingsService.IsAutoDownloadEnabled`.
        2. Використовує `StorageApplicationPermissions.FutureAccessList`, щоб отримати доступ до папки без повторного відкриття діалогу (навіть після перезапуску програми).
        3. Якщо токен валідний -> зберігає файл.
        4. Якщо ні -> ігнорує (юзер має натиснути кнопку "Download" вручну).

🎨 11. Зміна Теми та UI

- **Функція (Клієнт):** ThemeToggleButton_Click
  - **Логіка:** Перемикає _isLightTheme. Зберігає в SettingsService.
  - **Застосування:** Викликає SetAppTheme, яка змінює RequestedTheme кореневого елемента (ElementTheme.Dark/Light) та оновлює ресурси кольорів (Brushes) "на льоту".
- **Функція (Клієнт - Масштабування):** OnZoom (Ctrl + Wheel)
  - **Логіка:** Перехоплює подію прокрутки. Якщо натиснуто Ctrl, змінює коефіцієнт _currentZoom (від 0.7 до 1.2).
  - **Результат:** Змінює Width і Height кореневого RootGrid, що емулює масштабування всього інтерфейсу.

👤 12. Профіль та Акаунт

- **Редагування профілю:**
  - **Клієнт:** SaveProfile_Click. Відправляє пакет UpdateProfile з новим іменем, біо та Base64 аватарки.
  - **Сервер:** Оновлює Users table. Розсилає ProfileUpdated всім клієнтам, щоб вони оновили кеш контактів.
- **Видалення акаунту:**
  - **Клієнт:** DeleteAccountButton_Click -> DeleteAccount packet.
  - **Сервер:** HandleDeleteAccountAsync -> _db.SoftDeleteUser.
    - Замінює Username на deleted_GUID.
    - Очищає пароль та аватар.
    - Виконує VACUUM для очищення місця.
  - **Клієнт:** Отримує підтвердження, розлогінюється, очищає локальний кеш.

🧩 13. Стікери та GIF

- **Завантаження Стікерів:**
  - **Клієнт:** При відкритті меню шле GetStickerPacks.
  - **Сервер:** Сканує папку Stickers, повертає список паків.
  - **Клієнт:** При виборі паку шле GetStickerPackContent. Отримавши список файлів, показує їх. Самі картинки завантажуються лениво (lazy loading) через GetSticker і кешуються в _stickerCache.
- **Відправка (Механізм):**

- Стікер відправляється як текст: <STICKER:PackName|FileName>.
- GIF відправляється як текст: <GIF:FileName>.
- Сервер не зберігає саме зображення в БД повідомлень, тільки цей тег.

🔍 14. Пошук (Search)

- **Пошук чатів/юзерів (**UserSearchBox**):**
  - Працює в гібридному режимі. Спочатку фільтрує локальний список _allUsersCache. Через 500мс (debounce) шле запит SearchUsers на сервер для пошуку глобальних юзерів.
- **Пошук повідомлень (**ChatSearchBox**):**
  - **Клієнт:** Шле пакет SearchHistory з текстом запиту і targetUsername (або groupId).
  - **Сервер:** Виконує SQL LIKE %query% в таблицях повідомлень. Повертає список знайдених повідомлень.
  - **Клієнт:** При кліку на результат використовує GetHistoryAroundId (Jump to message), щоб завантажити контекст повідомлення.

🗑️ 15. Управління Чатами (Видалення та Вихід)

- **Видалення Чату (Private):**
  - **Клієнт:** DeleteChat_Click. Параметр forEveryone (checkbox).
  - **Сервер (**HandleDeleteChatAsync**):**
    - Якщо forEveryone = true: Видаляє записи з DirectMessages.
    - Якщо false: Оновлює таблицю ChatSettings, встановлюючи LastClearedMessageId на ID останнього повідомлення. Це "ховає" старі повідомлення для цього юзера, але залишає їх у базі.
- **Вихід з Групи:**
  - **Клієнт:** Якщо юзер не адмін, кнопка "Leave Group". Шле пакет LeaveGroup.
  - **Сервер:** Видаляє запис з таблиці GroupMembers.
- **Видалення Групи (Admin):**
  - **Клієнт:** Якщо юзер адмін -> DeleteGroup.
  - **Сервер:** Видаляє групу з Groups, всіх учасників з GroupMembers та історію з GroupMessages. Розсилає сповіщення GroupDeleted.

16. Голосові Повідомлення (Voice Messages)

- **Функція (Клієнт - Запис):** VoiceButton_Click

    - **Логіка:** Використовує VoiceMessageManager та MediaCapture. Записує аудіо в пам'ять -> конвертує в Base64 -> шле повідомлення з тегом <VOICE:base64_data>.

- **Функція (Клієнт - Відтворення):** PlayVoiceMessage_Click.
    - **Логіка:** Використовує один глобальний MediaPlayer. Декодує Base64 назад у потік. Таймер оновлює прогрес-бар.

17. Відповідь на повідомлення (Reply)

- **UI:** ReplyContext_Click активує режим відповіді (_replyingToMessage).
- **Мережа:** При відправці додається параметр replyToId.
- **Відображення:** Сервер заповнює поля ReplyToSender та ReplyContent (через SQL Join), тому клієнт одразу показує, на що це відповідь, без додаткових запитів.

🏷️ Частина 5: Довідник Спец-Тегів (Rich Content)

Система розпізнає тип контенту за текстовим тегом на початку повідомлення.

Тип	Формат у БД / Протоколі	Як обробляє Клієнт (AddMessageToView)
Голосове	<VOICE:base64_data>	Декодує Base64, показує Slider і кнопку Play.
GIF	<GIF:filename>	Показує анімацію. Якщо немає в кеші -> запитує GetGif.
Стікер	<STICKER:pack|name>	Показує картинку з паку стікерів.
Картинка (Ref)	<IMG_REF:blobId|name>	Показує плейсхолдер. Фоном вантажить GetFileContent.
Файл (Ref)	<FILE_REF:blobId|name|size>	Показує іконку файлу, назву і розмір. Клік -> Завантаження.
Картинка (Legacy)	<IMAGE:base64>	*Старий формат.* Картинка вшита в текст.

⚠️ Частина 6: Важливі Технічні Нюанси

1. Потоки (Threading UI):
   Всі події від NetworkClient приходять із фонового потоку. У MainPage.xaml.cs обов'язково використовувати _dispatcherQueue.TryEnqueue(() => { ... }) для будь-якого оновлення інтерфейсу (додавання в ObservableCollection, зміна тексту), інакше програма "впаде".

2. Очищення Пам'яті (DB Vacuum):
   При видаленні повідомлень або юзерів (DeleteChat, DeleteAccount), SQLite не звільняє місце на диску автоматично. Тому в DatabaseContext після видалення викликається команда VACUUM;, яка перебудовує файл бази даних і зменшує його розмір.

3. Безпека Паролів:
   Паролі ніколи не передаються у відкритому вигляді після реєстрації (крім моменту входу). В базі зберігається лише хеш (AuthenticationService). При зміні пароля сервер вимагає спочатку підтвердити старий пароль (звіряє його хеш).

Частина 7. Короткий словник файлів

- **Program.cs**: Кнопка "Пуск" для сервера.
- **Server.cs**: "Диспетчер". Приймає дзвінки (підключення) і роздає їх операторам (ClientHandler).
- **ClientHandler.cs**: "Оператор". Працює з конкретним клієнтом. Знає, як відповісти на "Login", "SendFile" і т.д.
- **DatabaseContext.cs**: "Бібліотекар". Тільки він вміє читати і писати в книгу (файл uchat.db).
- **MainPage.xaml.cs**: "Обличчя". Керує всім, що ви бачите: кнопками, списками, анімаціями.
- **NetworkClient.cs**: "Кур'єр". Його робота — просто взяти пакет і доставити його на сервер, або принести пакет з сервера. Він не думає, що всередині.