Технічні науки

УДК 004.62:006.86

Коваленко Олександр Сергійович
доктор медичних наук, завідувач відділу
Міжнародний науково-навчальний центр
інформаційних технологій та систем НАН України

Завражний Дмитро Костянтинович
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»

Гулін Іван Леонідович
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»

Наджафиан Тумаджани Махаммадали
молодший науковий співробітник
Міжнародний науково-навчальний центр
інформаційних технологій та систем НАН України

ПРОТОТИП РОЗПОДІЛЕНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ ДЛЯ ЕЛЕКТРОННОЇ ОХОРОНИ ЗДОРОВ'Я

Анотація. *В статті розглянуті приклади реалізації децентралізованих баз даних в медицині, де описано проектування, прототипування та розроблену інформаційну інфраструктуру.*

Ключові слова: *блокчейн, система охорони здоров'я, безпека, мобільний додаток.*

Вступ. Побудова розподіленої інформаційної системи для електронної охорони здоров'я може вирішити ряд питань, які наразі існують в e-Health. Найбільш актуальними є питання децентралізації, захисту інформації та адаптивності інформаційних систем. Нажаль, в сучасному світі, переважна кількість компаній та виконавців проектують системи лише в розрізі своїх цілей, на базі вже існуючих розробок або патернів. З часом дані розробки мають проблеми, які не були вирішені в процесі проектування. Наприклад, не було передбачено проблему передачі документів в інший медичний заклад, що може бути вирішене застосуванням стандарту Health Level 7 (далі HL7).

В даній статті розглянуто прототип додатку, який має вирішувати основні проблеми інфраструктури для електронної охорони здоров'я, а саме: децентралізація та захист даних користувачів, стандартизація та адаптивність системи, простота користування, комунікація.

Проблеми, які має вирішувати система. Перед пропонованою інфраструктурою поставлено ряд завдань, які мають бути вирішеними завдяки сучасним підходам та патернам.

1. Децентралізація та захист інформації. Додаток має відповідати стандартам захисту інформації та запобігати розповсюдженню персональної інформації.

2. Стандартизація збереження та обміну даними. За допомогою стандартизації інформації в додатку, в подальшому можна впроваджувати обмін між іншими сервісами, які не відносяться до основного.

3. Кросплатформеність системи. Кросплатформеність додатку може надати можливість використання одного коду на різних пристроях, що зменшить обсяги роботи над проектом.

4. Комунікація користувачів. Користувачі системи повинні мати можливість комунікувати один з одним.

Інструменти розробки. Після дослідження сучасних підходів та методів побудови інфраструктури для електронної охорони здоров'я, було знайдено декілька методів реалізації даної платформи:

1. Розподілена інформаційна інфраструктура. В даному підході є перевага щодо захисту інформації. Доступ до особистих даних користувача відбувається за допомогою єдиного ключа. Тобто користувач особисто регулює усі можливі операції над своїм гаманцем. Можливим мінусом даної системи є зменшення можливостей, які можна буде реалізувати.

2. Клієнт-серверний застосунок. Типовий підхід для реалізації будь-якого додатку. В даному підході існують проблеми, які не можуть бути вирішеними, а саме повноцінний захист інформації. Так, дані можуть знаходитися на захищеному сервері, але вони більш вразливі аніж збережені дані через розподілену інфраструктуру.

3. Розподілена інфраструктура із зберіганням даних на клієнтському застосунку. Досить неоднозначний підхід, але досить захищений. В даному підході розглядається система, в якій інформація зберігається на кінцевому пристрої користувача. Останній може поділитися ними із іншими користувачами. Проблемою є менша гнучкість системи, незахищеність від втрати даних. Але перевагою є захищеність даних, які не зберігаються ніде крім кінцевого пристрою.

Нами розглядається побудова розподіленої інфраструктури для електронної охорони здоров'я. Система має включати в себе обмін документами, інформацією про стан здоров'я пацієнта, довірену сутність, яка зможе видавати документи державного рівня для нових користувачів, комунікацію між кінцевими користувачами. Для побудови такої системи можна використати досить новий інструмент Hyperledger Indy. Даний інструмент наразі в процесі розробки, але його вже можна використовувати для обміну документами між сутностями або установами за допомогою розподіленої інфраструктури блокчейну.

Основним стандартом комунікації документів між системою та користувачем потрібно використати стандарт HL7. Це досить розповсюджений стандарт та має постійні оновлення та інтеграції. За його допомогою можна вирішувати питання адаптивності інформації під інші сервіси. Також система має використовувати стандарт DICOM для обміну зображеннями та їх інформацією.

Для побудови візуальної частини можна використати інструмент React Native. Даний інструмент побудований на базі JavaScript, та в кінцевому вигляді перетворюється на мобільний застосунок. Завдяки React Native можна побудувати мобільні додатки IOS та Android, а також веб-додаток, в рамках одного проекту, що досить пришвидшує роботу та вирішує питання кросплатформенності.

Опис основних сутностей системи. В будь-якій системі мають бути сутності, які між собою спілкуються та виконують обмін інформацією. В прототипі даної розподіленої інфраструктури це обов'язково. Далі наведено приклади основних сутностей, які будуть використані в системі:

1. Сутність, якій можна довіряти:

Мабуть, найважливіша сутність, на якій буде будуватися система. Даний гравець має видавати документи державного значення для інших користувачів, а саме:

- паспорт;
- ідентифікаційний код;
- диплом;
- сертифікат.

Кожний користувач може перевірити іншого за допомогою цих даних. Дана сутність по суті є державою в розрізі даної системі, і їй можна завжди довіряти, тому що це єдиний вхід для документів державного значення.

2. Гравці. Гравцями розподіленої системи є лікарі та пацієнти. Вони між собою спілкуються та виконують обмін. Кожен пацієнт завдяки сутності

«держава» може перевірити свого лікаря, запитавши в нього номер диплому або сертифікати для перевірки. Якщо лікар не буде в змозі це підтвердити, то пацієнт буде в праві перервати контакт з лікарем.

В даній системі досить важливим є момент того, що користувач самостійно розпоряджається своїми даними. Також він самостійно знайомиться з лікарем і вибирає його в якості свого лікаря.

В свою чергу лікар може виписувати документи пацієнтам (наприклад, результати аналізів, обстеження тощо). Пацієнт ці документи отримує в свій гаманець, та так само може поділитися з іншим лікарем, який, наприклад, буде проводити аналіз обстеження та робити висновки про лікування.

3. Елементи обміну. Основними елементами обміну даними є документи користувачів. Вони поділяються на 2 рівні – державні (ті що видає сутність держава), та локальні (ті що видають лікарі). Державні може редагувати лише сутність держава, тому вони завжди валідні та правильні, і їм можна довіряти іншими користувачами.

Локальні документи виписують лікарі, які мають контакт із пацієнтами. На основі цих документів будується інша частина системи обміну. Саме на цьому етапі обміну використовуються стандарти HL7 та DICOM. Дані можуть бути як експортовані так і імпортовані в систему. Саме стандарт HL7 дозволяє інтегрувати дану інфраструктуру з іншими системами, які використовують той же стандарт. Стандарт DICOM дозволяє виконувати обмін медичними зображеннями між пацієнтом та лікарем.

4. Додаткова інформація, або інші дані. До цього типу даних можна віднести сповіщення, які будуть з'являтися в кабінеті користувача та медичні записи пацієнта. За допомогою сповіщень будується контакт між користувачами. В своїх сповіщеннях користувач може побачити запрошення на створення контакту, або запит на додавання документів. Також за допомогою цих сповіщень побудований чат, в якому користувачі можуть спілкуватися.

Медичні записи пацієнта виконують роль записника. Цими даними можна буде в результаті поділитися з доктором, який в свою чергу їх отримає, та зможе проаналізувати.

Прототип системи. Система будується на основі розподіленої бази даних (далі – блокчейн). В контексті інформаційної медичної інфраструктури використовується інструмент Hyperledger Indy.

В Hyperledger Indy наявна можливість будувати схеми майбутніх документів (Scheme), в якій вказуються можливі поля документів та їх типи даних. На основі цих схем буде будуватися обмін документами між користувачами.

На базі розподіленої інфраструктури будується взаємозв'язок між сутностями. На рис. 1 зображено спроектовану взаємодію між сутностями лікар, пацієнт та держава.

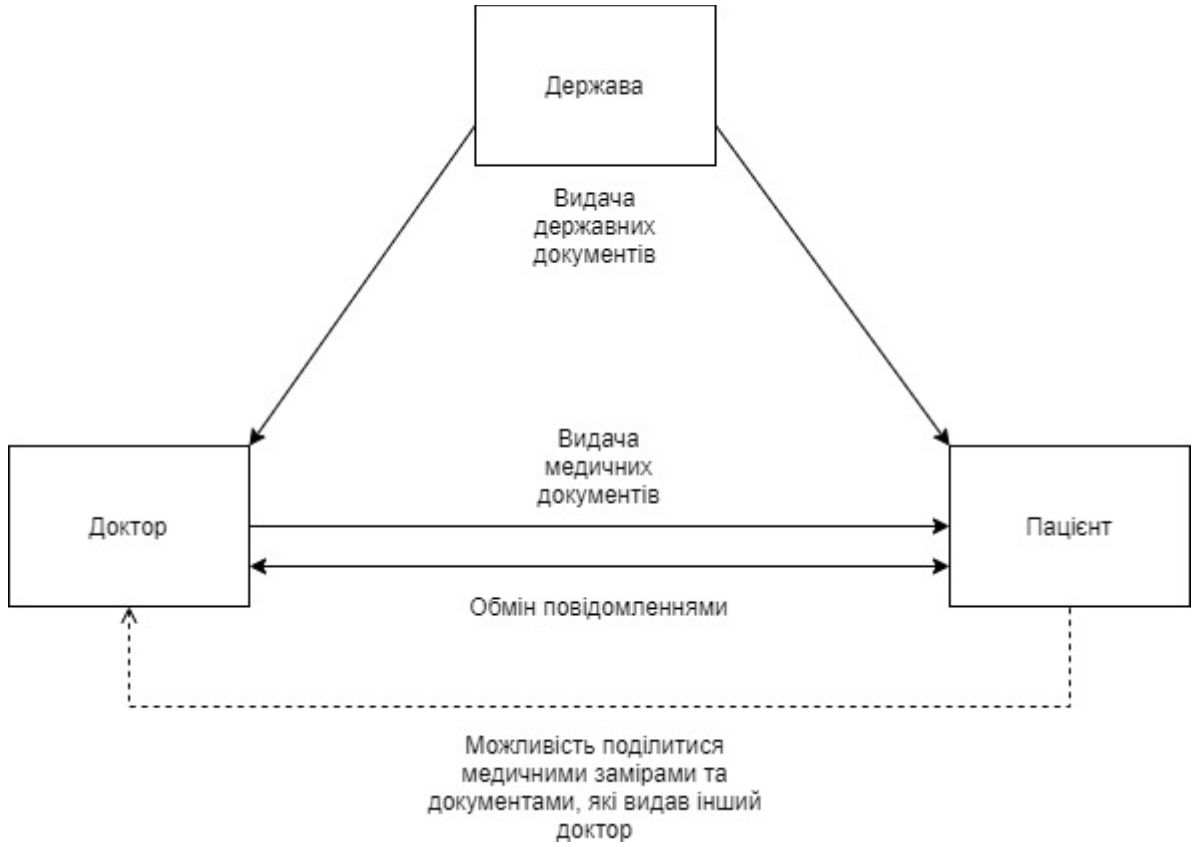

Рис. 1. Приклад взаємодії між сутностями

Держава виступає джерелом державних документів, які при реєстрації видаються користувачам.

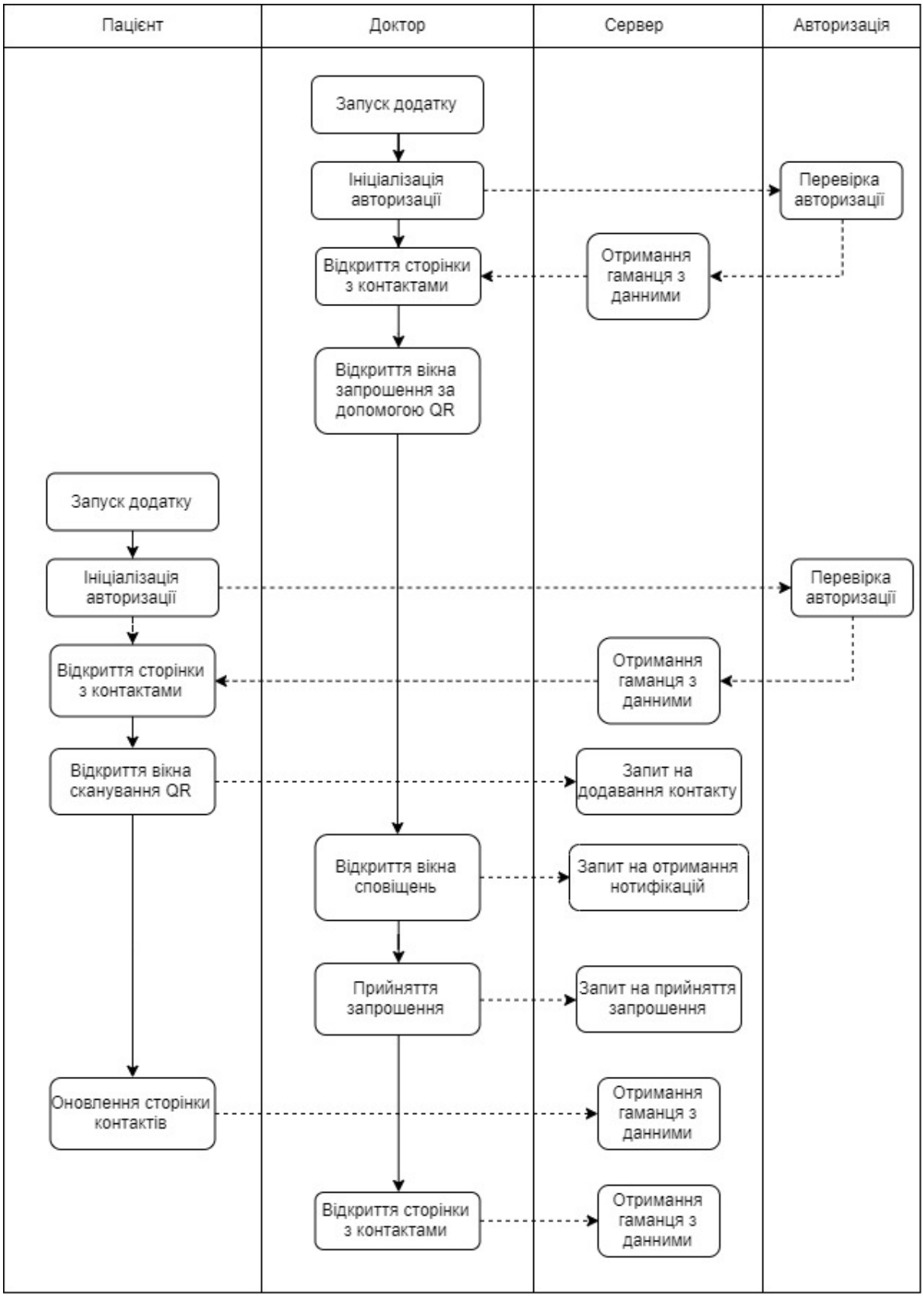

Рис. 2. UML Activity Diagram бізнес-процесу встановлення контакту

У лікаря та пацієнта наявні такі загальні документи:

- паспорт;
- ідентифікаційний код.

Для лікаря пакет документів розширюється, та також додається:

- сертифікат;
- диплом.

Ці документи дають змогу верифікувати один одного пацієнтом та доктором.

Джерелом медичних документів виступає доктор. В даній інфраструктурі тільки доктор може додавати до пацієнта документи. Планується використовувати два типи документів:

- HL7 процедура;
- DICOM зображення з метаданими.

Спосіб збереження усіх даних користувачів – використання розподіленої інфраструктури блокчейн. У кожного користувача буде свій гаманець, в якому зберігаються усі важливі дані, а саме: документи, чат, медичні записи, дані про користувача.

Основний процес взаємодії буде побудований як раз на основі документів (як державних так і медичних). Базовим методом буде процес встановлення зв'язку між пацієнтом та лікарем.

Спосіб комунікації та підтвердження тих чи інших дій – сповіщення. За допомогою сповіщень користувачі зможуть підтверджувати записи сутностей чи їх видалення в гаманці блочейн.

Вивід інформації буде побудований на основі інструменту React Native. На його базі буде відбуватися реалізація усіх бізнес-процесів, пов'язаних із додатком.

Можливі бізнес-процеси:

- реєстрація;
- авторизація;

- створення зв'язку між користувачами (приклад на рис. 1);
- запит на отримання документів для перевірки;
- процес комунікації (чат);
- процес додавання медичного документу (лікарем);
- взаємодія із сповіщеннями;
- процеси експорту та імпорту документів;
- процес додавання медичного запису (пацієнтом);
- видалення гаманця;
- завершення сесії.

Опис бек-енд частини. Серверна частина система реалізована за допомогою мови програмування Java. Архітектура показана на рис. 3.

Фронт-енд частина спілкується із серверною через Gateway Service, в якому реалізовано основну частину API. Цей сервіс надалі розподіляє запит в інші сервіси, які видають потрібну інформацію, або виконують запис даних.

Запити поділяються також на ті, що потребують авторизацію, або що не потребують. Перед виконанням цих запитів потрібно виконувати Oauth авторизацію з отриманням токену (або виконувати оновлення токену за допомогою refresh_token).

Основним способом спілкування з гаманцем користувача та з розподіленою системою є запити до User service. Даний сервіс відповідає за усі дії, пов'язані із користувачем в блокчейн системі (методи отримання гаманця або запису в гаманець).

Сервіс Government відповідає за сутність «Держава», яка видає документи державного рівня. В ньому реалізоване отримання та додавання документів системи. Державні дані зберігаються в звичайній SQL базі даних, та виконують роль державного API.

Documentation сервіс відповідає за етап розробки додатку. Фронт-енд розробник з легкістю може отримати інформацію про запити, які існують в серверній частині.

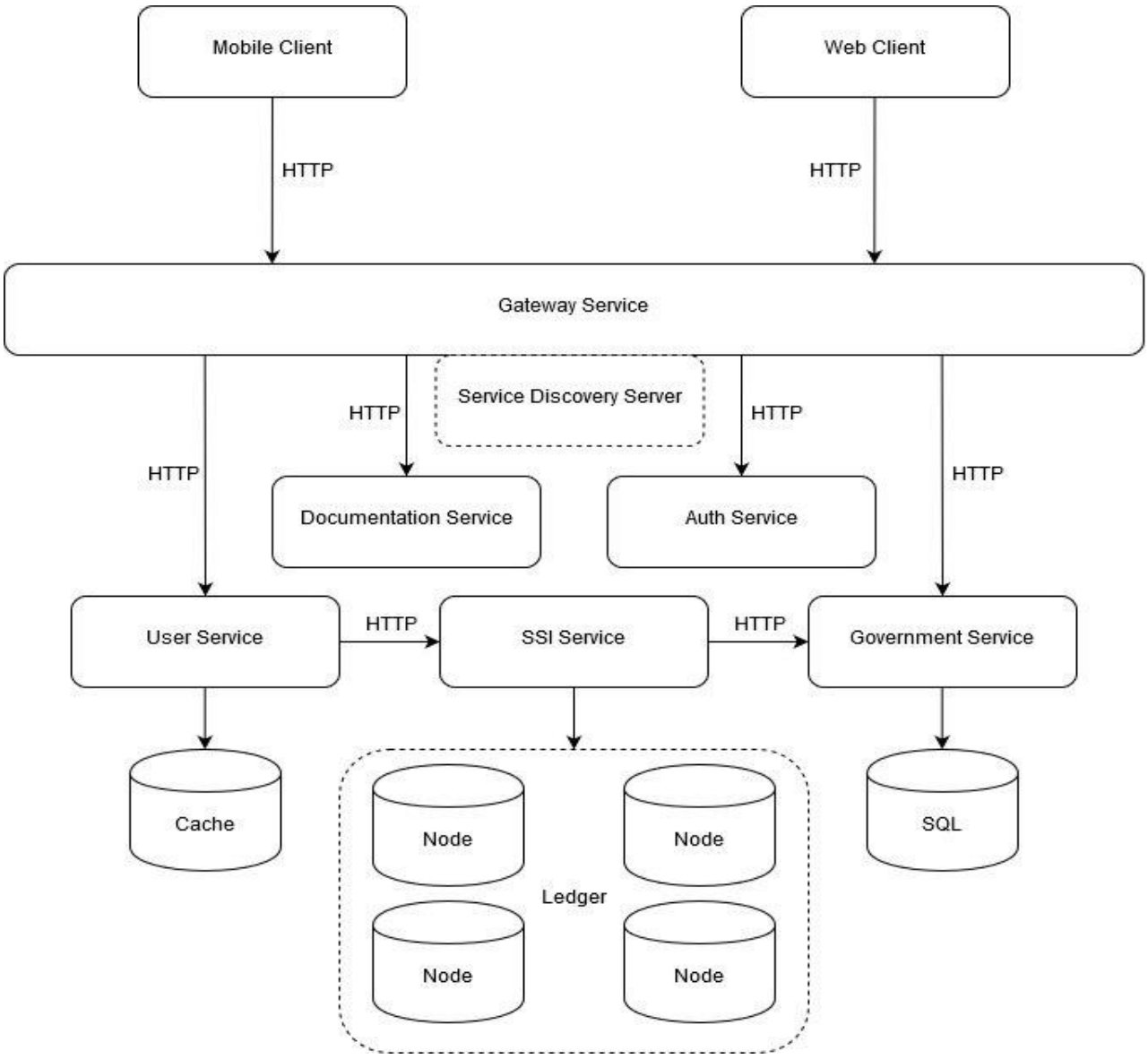

Рис. 3. Архітектура серверної частини

Усі взаємодії відбуваються за допомогою HTTP запитів. Кожен сервіс побудовано на основі Docker контейнерів, та спілкуються між собою. Кожен з контейнерів захищено, доступ до них має тільки Gateway сервіс.

SSI сервіс побудований на базі Hyperledger Indy, який в свою чергу має розподілену інфраструктуру блоків даних. Кожен блок являє собою гаманець із даними, доступ до якого можна отримати лише за допомогою ключа, який видається при створенні гаманця.

Опис фронт-енд частини. Фронт-енд частина використовує серверний Gateway сервіс, та побудована на базі React Native фреймворку. Дана бібліотека

являє собою середовище, за допомогою якого можна побудувати Android, IOS та Web додатки, використовуючи при цьому єдиний код та середовище, що вирішує проблему кросплатформеності.

Інструмент побудований на базі NodeJS, що дозволяє використовувати сучасні технології при побудові застосунку. NodeJS має досить велике співтовариство завдяки якому розвивається к React Native так і NodeJS.

Побудований проект використовує HTTP запити до серверної частини, за допомогою яких вільно спілкується з нею, може виконувати основні бізнес-процеси та візуально відображати потрібну інформацію.

React Native має компоненту структуру, тобто кожен елемент можна запрограмувати як компонент, та вбудовувати логіку для кожного. Фреймворк використовує мову програмування JavaScript, та системі контроллери JavaScriptCore, за допомогою яких в побудованому IOS та Android додатку відбувається відображення інтерфейсу, виконання логіки та інші операції.

В якості локальної бази даних використовується фреймворк Redux. З його допомогою можна легко виконувати зберігання стану системи, та його постійного оновлення. Даний фреймворк використовує структуру моделей, в яких описані основні операції, що існують в додатку. Для простоти використання фреймворку Redux, в фронт-енд частині також використовується обгортка Rematch. Вона робить використання Redux простішим та більш зрозумілим.

Чат реалізовано за допомогою модуля GiftedChat. Його було запрограмовано під потреби проекти, що в результаті надало готовий функціонуючий чат.

Для побудови функціоналу сканування QR коду та його генерації, було використано бібліотеки react-native-qr-scanner та react-native-qr-builder. Це надало можливість виконувати зв'язок між пацієнтом та лікарем за допомогою QR коду.

Побудова меню та пагінації була розроблена за допомогою фреймворку react-native-flux. Даний фреймворк містить в собі велику кількість інструментів, що реалізують передачу потрібних параметрів між екранами, побудову структури застосунку, зв'язки між екранами та структурою.

В проекті використовуються стилізовані кнопки та структури із бібліотеки react-native-elements. За допомогою неї будується основний дизайн проекту та його зовнішній вигляд.

Використання кешу даних та можливості офлайн роботи продукту реалізовано за допомогою фреймворку Redux-Persist.

Основними інструментами розробки є редактор коду Visual Studio Code, будівник проекту, який дозволяє в реальному часі перевіряти зміни в готовому продукті – Expo-Cli та Node.js консоль.

Проект використовує декілька моделей взаємодії Redux, а саме:

- user;
- notifications;
- chat;
- ledger;
- oauth;
- documents.

Також проект має декілька особливостей, які покращують його роботу:

- використання сутностей Middleware. В фреймворку Redux присутня можливість запрограмувати виклик методів, які виконуються перед зазначеним методами. Налаштування відбуваються в конфігураційному файлі.

- до методів та будь-яких моделей присутній доступ із будь-якої частини додатку (якщо ця частина входить в основний та стартовий (root) компонент застосунку).

Кожна модель фреймворку Redux може спілкуватися із іншими моделями, які присутні в основному компоненті додатку.

Модель oauth відповідає за авторизацію користувача в системі, та отримання ключа доступу до інших моделей. Також в цій моделі присутній метод оновлення ключа доступу. В основному, дана модель спрацьовує явно на етапі авторизації користувача, та неявно в процесі перевірки авторизації перед використанням інших методів за допомогою middleware методів.

Модель ledger призначена для зберігання ключів доступу до гаманця розподіленої системи. Вона використовується явно для виконання процесу зберігання, та неявно для перевірки наявності цих ключів. Це потрібно для попередньої перевірки реквізитів запиту інших моделей.

Модель user відповідає за всі методи та запити, пов'язані із користувачем. Наприклад: створення гаманця, отримання даних про користувача, локальні методи отримання цільових документів із масиву користувача. Дана модель використовує також моделі ledger та oauth для перевірки реквізитів запиту.

Модель notifications реалізовує процес отримання сповіщень на проведення операцій із гаманцем кінцевого користувача. Дана модель також використовує моделі ledger та oauth.

Модель chat використовується для реалізації чату між користувачами. Дана модель також використовує моделі user, notifications, oauth та ledger. Модель user використовується для отримання даних контакту, та для отримання особистих даних відправника. Через модель notifications виконується отримання повідомлень, які надалі кінцевий користувач заповнює до себе в гаманець в фоновому режимі.

Модель documents використовується для реалізації обміну документами між користувачами. Вона також використовує моделі user, notifications, ledger та oauth.

Реалізація фронт-енд частини. При завантаженні додатку, виконується перевірка авторизації користувача, запит на отримання гаманця, якщо ще діє авторизація та відображення інформації. У випадку якщо авторизації немає, або

запит на отримання гаманця повернув помилку, йде повернення фронт-енд частиною екрану із вибором одного з двох: реєстрації або авторизації.

Якщо користувач вибрав реєстрацію, він повинен заповнити дві форми. Перша форма відповідає за створення гаманця на стороні сервера в розподіленій інфраструктурі. Користувач повинен заповнити електронну пошту та пароль (рис. 4).

При успішному створенні гаманця, користувача буде відправлено на другий етап реєстрації.

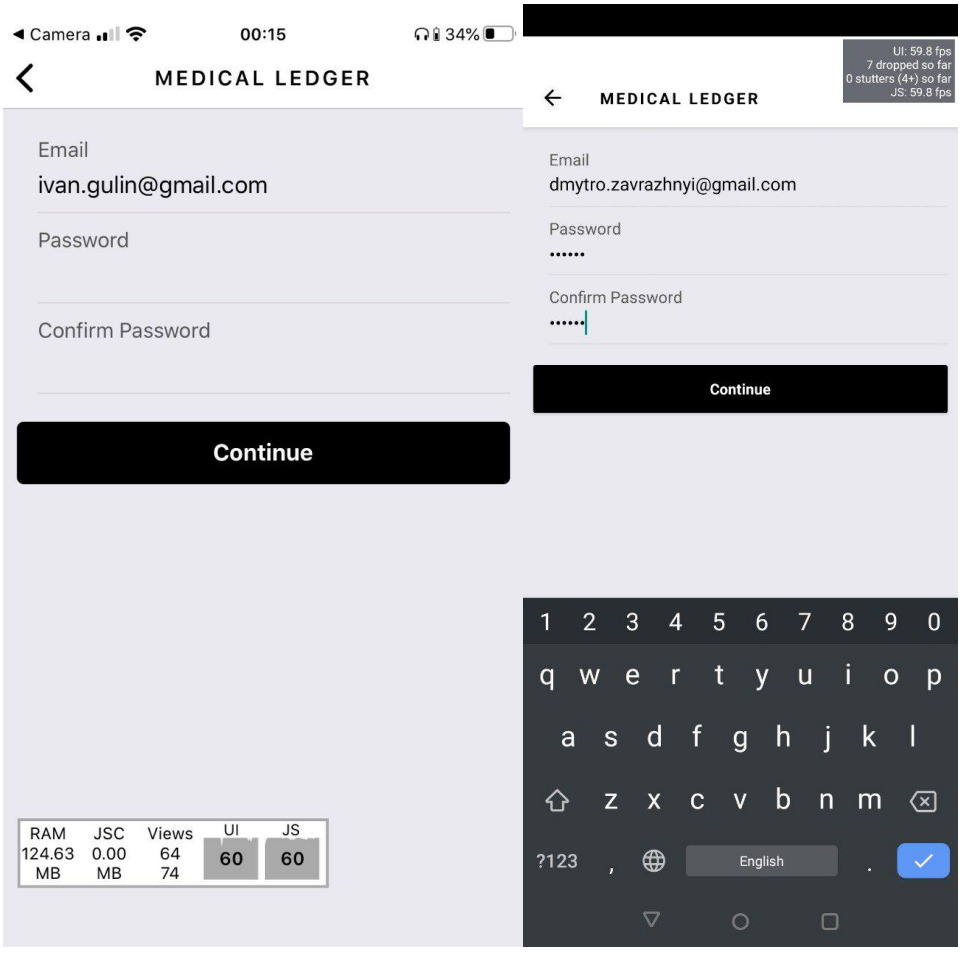

Рис. 4. Перший етап реєстрації

На другому етапі реєстрації (рис. 5) користувачу потрібно заповнити його державні дані. Це потрібно для пошуку користувача в державному реєстрі (сервіс Government).

Після успішного завершення другого етапу реєстрації, в розподіленій інфраструктурі заповнюються початкові дані користувача. Його буде

відправлено на перший екран додатку – «Зв'язки». На ньому можна побачити зв'язок лише із сутністю Government.

Для відтворення процесу додавання зв'язку (рис. 6) лікарю потрібно згенерувати QR код - натиснути «+» біля заголовку, та вибрати Create invitation, а для пацієнта вибрати Scan code.

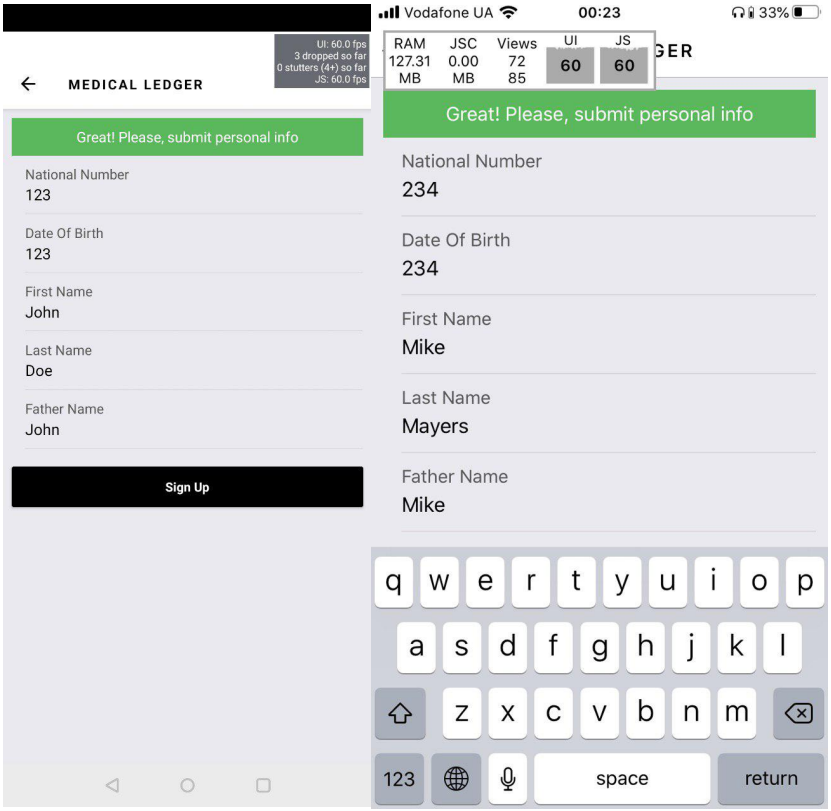

Рис. 5. Другий етап реєстрації

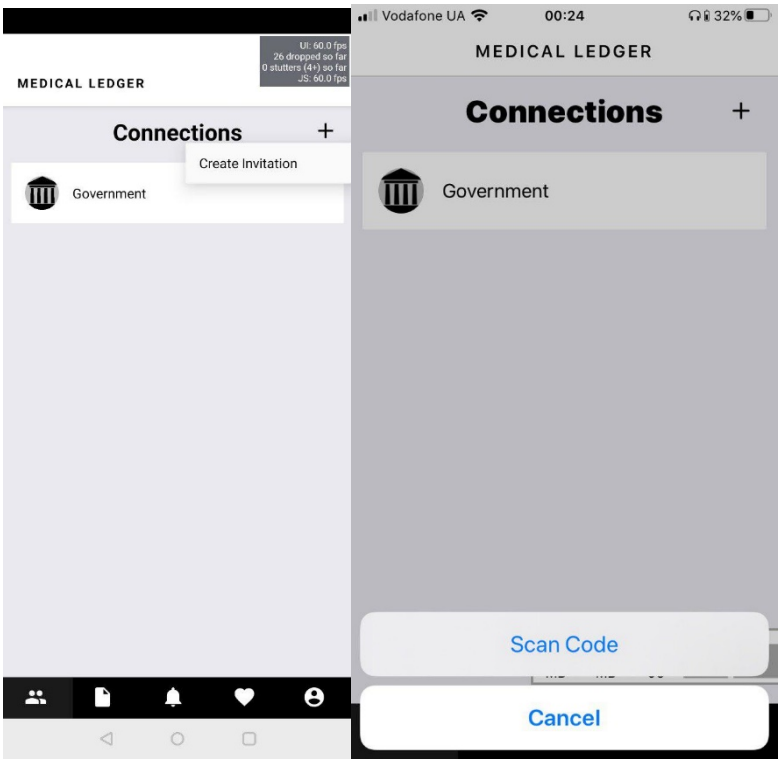

Рис. 6. Вибір методу взаємодії

Коли доктор натискає потрібну кнопку, генерується QR код, який пацієнт має засканувати за допомогою камери (рис.7).

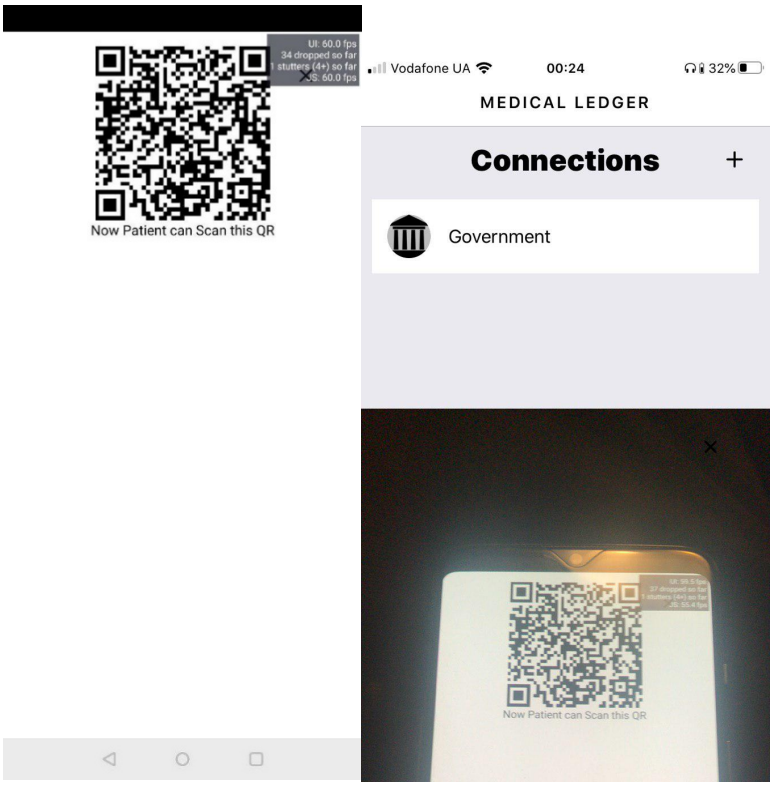

Рис. 7. Сканування та генерація QR коду

Після успішного розпізнавання QR коду, пацієнту буде висвітлена форма для підтвердження створення зв'язку (рис. 8).

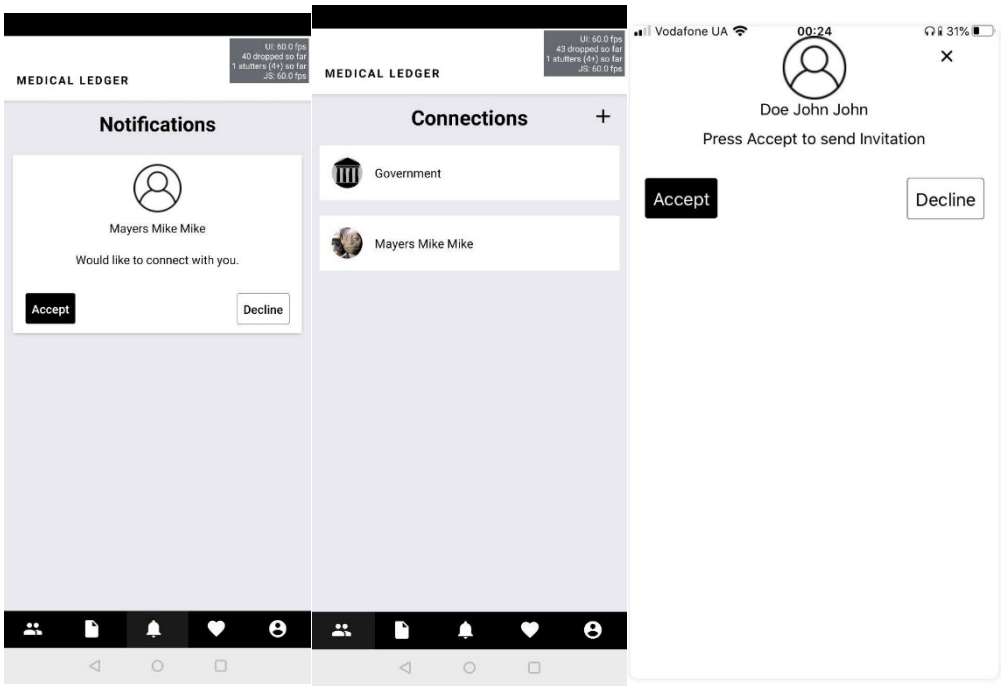

Рис. 8. Екран підтвердження створення зв'язку

При натисканні клавіші Accept, буде відправлено сповіщення до лікаря, щоб він підтвердив створення зв'язку, в той же час у пацієнта теж з'явиться сповіщення, про те, що в даний період часу існує один не прийнятий зв'язок доктором. Пацієнт може також відмінити створення зв'язку, а доктор має прийняти або відмовити даний зв'язок (рис. 9).

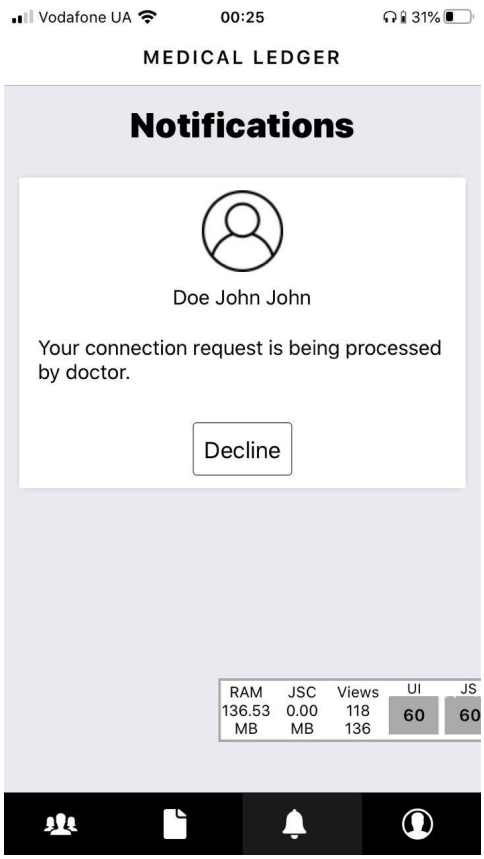

Рис. 9. Екран нотифікацій під час створення зв'язків

Якщо доктор натисне Accept, то до гаманця доктора та пацієнта буде успішно додано контакт (зв'язок) (рис. 10).

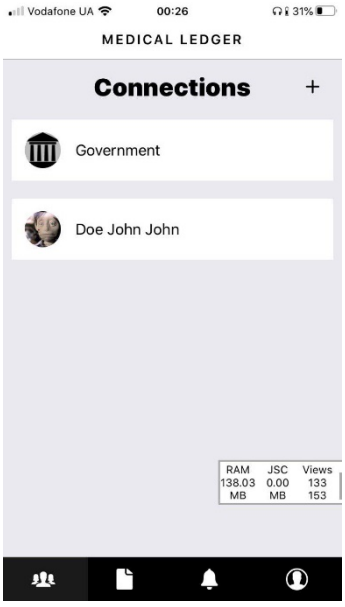

Рис. 10. Екран зв'язків

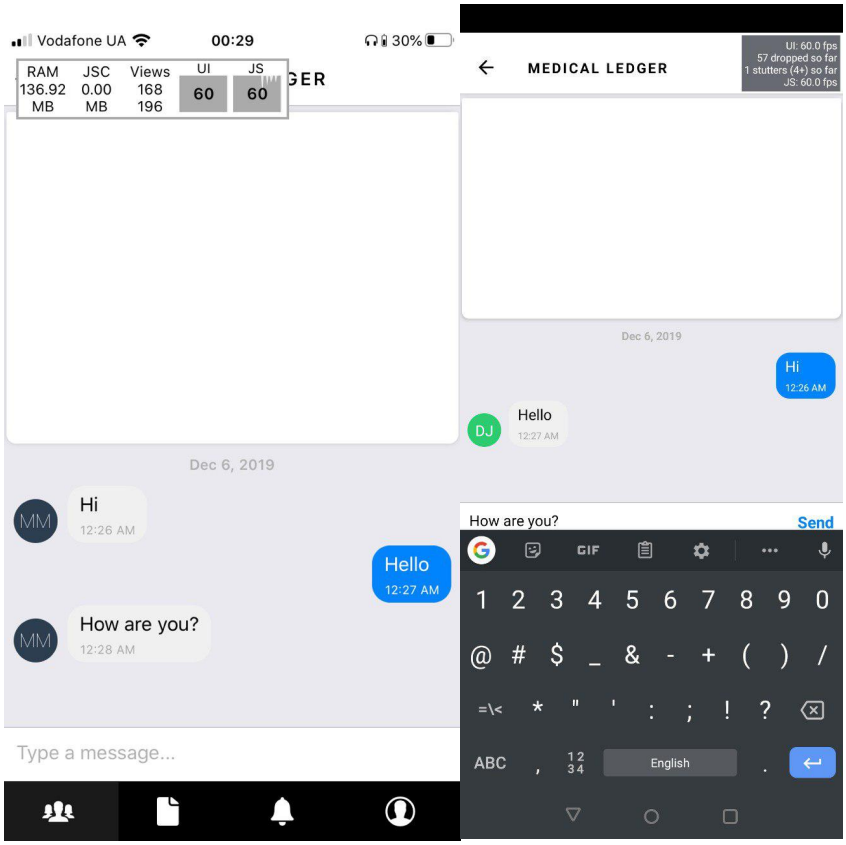

Рис. 11. Екран чату

Також пацієнт та доктор можуть відкрити чат та спілкуватися між собою. Для цього потрібно натиснути на контакт.

Висновки. Нами розроблено прототип розподіленої інформаційної інфраструктури для електронної охорони здоров'я з застосуванням платформи блокчейну, яка дає можливість підвищити ефективність обміну даними між учасниками процесу надання медичної допомоги (лікарі та пацієнти) та реалізувати захист інформації.