Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Реалізація мобільного додатку EU Digital Identity Wallet

Текстова частина до курсової роботи
за спеціальністю «Комп'ютерні науки» - 122

Керівник курсової роботи
старший викладач
Гороховський К.С.
_____ (Підпис)
"___" _____ 2024 року
Виконав студент
КН-3 Кучеренко Д.О.
"___" _____ 2024 року

Київ – 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

_____Жежерун О.П

(підпис)

„____”_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту____Кучеренко Д.О____ факультету _інформатики_ ____3-го _курсу
Тема _____ Реалізація мобільного додатку EU Digital Identity Wallet____
Вихідні дані:
Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
1 Огляд існуючих ідентифікаторів
2 Огляд механізму створення DID
3 Розробка власного децентралізованого цифрового гаманця
Висновки
Список літератури
Додатки (за необхідністю)

Дата видачі „___” _____ 2024 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: _____Реалізація мобільного додатку EU Digital Identity Wallet_____

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проєкту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	02.11.2023	
2.	Огляд технічної літератури за темою роботи	15.11.2023	
3.	Виконання аналізу сучасних методів	25.11.2023	
4.	Розробка алгоритму	30.12.2023	
5.	Програмування розробленого алгоритму	15.01.2024	
6.	Застосування розробленого алгоритму до	30.03.2024	
7.	Написання пояснювальної роботи		
8.	Створення слайдів для доповіді та написання доповіді	29.04.2024	
9.	Остаточне оформлення пояснювальної роботи та слайдів.	10.05.2024	
10.	Захист курсової роботи (проєкту)	22.05.2024	

Студент ____Кучеренко Д.О._____

Керівник ___Гороховський К.С_____

“_____”_____

Зміст›

Анотація

Метою роботи є дослідження галузі Digital Identity Wallets. Конкретно розглянута її робота, формати представлення даних та існуючі застосунки, які пропонують свої рішення, щодо використання цієї технології. Також був розроблений свій демо-аналог цифрового гаманця.

Ключові слова: ЦИФРОВА ІДЕНТИЧНІСТЬ, ЦИФРОВИЙ ГАМАНЕЦЬ, VERIFIABLE CREDENTIALS, EUROPEAN IDENTITY DIGITAL WALLET, DIGITAL IDENTIFIERS, SWIFT, SWIFTUI.

Скорочення та умовні позначення

DID – decentralized identifiers. Унікальний децентралізований цифровий ідентифікатор для цифрових гаманців.

XSS атаки – cross-site scripting. Вид атаки з можливістю додавання до вже існуючої веб-системи шкідливого коду.

eIDAS - electronic IDentification, Authentication and trust Services.

VC – verifiable credentials. Документи, що можуть бути підтвердженими та відповідають стандарту Verifiable Credential Data Model 1.0.

JWS – JSON Web Signature. Це стандарт для підписання даних.

JWT – JSON Web Tokens. Стандарт для безпечного представлення вимог між двома організаціями.

RDF - Resource Description Framework. Це загальна структура для представлення взаємопов'язаних даних в Інтернеті.

Вступ

Кожен з нас використовує смартфон або ноутбук в повсякденні не дивлячись на професію: бізнес, медицина, навчання, тощо. Нові застосунки дозволяють мати неоціненний вклад в наші життя. Вони дозволяють використовувати велику кількість сервісів: відстеження стану здоров'я, месенджери, навігацію. Через деякий час, з'явиться потреба в зберіганні десь даних користувача, адже багато застосунків зав'язані саме на збереженні унікальних даних та інформації під кожного.

Зберігання даних є критичним, оскільки кількість файлів та інформації, з якою ми взаємодіємо, не зможе поміститись на сьогоденних носіях даних. Зрозуміло, що наша інформація містить багато приватних моментів, тож не варто забувати про безпеку: витоки цієї інформації та можлива втрата можуть бути критичними.

Подумаємо про збереження якоїсь інформації в застосунку банку. Наприклад, ідентифікаційний код, номер ID-картки, адреса проживання. По суті, при реєстрації ви, безпосередньо, надаєте цю інформацію банку для подальшого збереження її в окремих базах даних. Не буде відкриттям, що зловмисники це розуміють та проводять атаки на системи банків задля отримання конфіденційної інформації клієнтів. Можна навести багато таких прикладів по всьому світу. Один з найбільших банків Америки став об'єктом кібератаки в 2019-му році. Витік даних призвів до втрати персональної інформації понад 100 млн. громадян США.

Також не буде перебільшенням факт того, що задля збереження документів ми можемо використовувати хмарні технології для завантаження інформації. Як ми всі знаємо, хмарні технології належать великим компаніям, в яких теж є ймовірність витоку інформації. Отже покладатися на конфіденційність теж не варто. Аналогічно до банків, зловмисники використовують фішингові методи, щоб зламати акаунти

користувачів та дізнатися цінну інформацію. В 2012-му році сервіс Dropbox зазнав атаки, в якій зловмисникам вдалося дізнатися понад 78 млн. паролів користувачів для доступу до сервісу.

Заходячи на сайт, ми можемо реєструватись або входити через наші, вже існуючі, акаунти Google, Facebook, використовуючи OAuth, щоб полегшити процес «знайомства» з сайтом. Таким чином ми, окрім просто логіну, надаємо ще доступ до додаткової інформації нашого існуючого акаунту, яка може бути використана розробниками відповідного сайту. Тут, користувач також може стикнутися з кіберзлочинцями, які можуть заволодіти інформацією людини через токен, що передається під час логіну.

Постає питання як же вирішити проблему з безпекою в подібних випадках. Одним з рішень може бути використання Decentralized Identity Wallet. Це, так званий, гаманець-застосунок, в якому користувач може зберігати свої персональні дані та документи. Безпека полягає в тому, що, по суті, людина не має «відправляти свої документи на збереження» в інтернет на зберігання компаніям. Документ лишається на носії, але інформація про нього міститься в застосунку, який перевіряє дійсність цього документу та зберігає певний ключ про документ в блокчейні. В Європі Digital Identity Wallets починають набирати обертів та ставати більш популярними між розробниками та загалом компаніями.

При реєстрації будь-де, користувач надає доступ лише до потрібної інформації від компанії та завжди має право відкликати цей доступ. Тобто, це рішення також підходить для аутентифікації. Це унеможливлює зберігання сервісами персональної інформації та, відповідно, її можливу втрату. Ви більше не маєте носити з собою фізичний паспорт, щоб верифікувати його в магазині або водійське посвідчення, щоб взяти автомобіль в оренду під час відпочинку закордоном. Уся інформація доступна у вашому додатку-гаманці, який людина може сканувати та отримати потрібну інформацію про документ.

До кінця 2024-го року, кожен громадянин Євросоюзу повинен мати можливість створити цей гаманець, відповідно до регуляцій консорціуму eIDAS. Україна, скоріше більш розвинена та більш готова за Євросоюз прийняти ці вимоги, але, на жаль, немає великої кількості прикладів гаманців. Тож метою роботи є внесок в освоєння та розвиток ринку Digital Identity Wallets в Україні

1. Digital Identity

1.1 Digital Identity

Digital Identity – це збірка інформації про людину або організацію, яка зберігається онлайн. Щодо конкретних даних, то цифрова ідентичність включає в себе таку інформацію: ім’я користувачів та паролі, історії пошуку, номер соціального страхування та історію покупок. Незважаючи на вже достатній розвиток цієї технології, вона ще не є настільки децентралізованою як цього хотілося б.

Siloed Identity

Найбільш популярним видом Digital Identity все ще є Siloed Identity. На жаль, вона не є настільки безпечною, оскільки не дає такого контролю даними та піддається атакам та, відповідно, втраті даних. Ви все ще лишаєтесь підв’язаними під інші компанії, які зберігають ваше ім’я користувача та пароль, а інколи і не тільки цю інформацію.

З однієї сторони, плюсом Siloed Identity можна вважати унікальність облікових даних. Таким чином покращується безпека та конфіденційність, оскільки ім’я користувачів і паролі не повторюються. Можливість користувача звернутись до підтримки або написати скаргу не буде проблемою, враховуючи, що організація має всю потрібну інформацію вже у себе і їй не прийдеться чекати певний час, щоб отримати дані про особу через третіх осіб.

На противагу цьому, Siloed Identity не є зручною для простих користувачів через кількість сайтів та застосунків, які ми використовуємо. Величезна кількість компаній змушує нас створювати унікальні дані: ім’я користувача, паролі, які потім легко забуваються. Створювати однакові паролі на кожний сервіс теж не є виграшною ідеєю, оскільки призводить до

повної втрати безпеки. Якщо у зловмисника є доступ до одного ресурсу, то він може його отримати і до інших без зайвих ускладнень.

Federated Identity

На відміну від Siloed Identity, Federated Identity покращує враження користувача від використання сервісів. Особа має можливість використовувати, вже існуючі облікові записи, для реєстрації або входу в сервіси. Для прикладу, під це може підпадати OAuth, де ви маєте можливість увійти в онлайн-сервіс за допомогою акаунту у Google або Facebook. Незважаючи на те, що це позитивно позначається на притоку користувачів, оскільки їм легше взаємодіяти з сервісом, це не робить цей підхід децентралізованим, оскільки все ще є компанія, яка має доступ до ваших даних.

Передаючи свої дані, навіть таким чином, ми надаємо доступ для монетизації та реклами. Не будемо забувати й при витоки інформації з того ж Facebook, коли приватна інформація, майже 78млн. користувачів, була використана не відповідаючи політиці конфіденційності компанії.

Можливостей отримати конфіденційну інформацію під час запиту на аутентифікацію досить багато. Починаючи з підробки токену, коли неправильна або недостатня валідація токенів дозволяє зловмиснику отримати доступ до ресурсів сервера, що надає валідні токени, і постачати фейкові токени, які відповідають його цілям. Підроблені токени можуть бути створені шляхом зміни певних характеристик реального токена (crafted token), або ж вони можуть бути створені з нуля. Закінчуючи XSS атаками під час якої, завдяки соціальній інженерії, створивши сайт-дублер, людина може отримати доступ до унікального токену для входу підв’язаного під користувача.

Decentralized Identity

Вирішенням проблеми незалежності користувача Digital Identity є Decentralized Identity. Це вид Digital Identity, при якому людина може повністю контролювати свої дані без надання доступу до них іншим сервісам. У випадку Decentralized Identity, має бути підтвердження дійсності документів, які людина використовує. В наш час це робиться просто за допомогою інтернету та відповідного застосунку в вашому смартфоні. Головною метою використання цієї технології є, якраз, можливість надати потрібні документи з верифікацією в реальному часі.

Користувач більше не має показувати свою дату народження або місце народження для того, щоб підтвердити, що він досяг повноліття або народився саме в цьому місті, відповідно. Ще одним плюсом використання Decentralized Identity є те, що ви можете в будь-який момент відкликати доступ до цієї інформації зі своєї сторони.

Децентралізація є важливою частиною, оскільки всі ми розуміємо, що компанії, маючи доступ до персональних даних, можуть використовувати їх для своєї мети незважаючи на регуляції і закони. Не варто забувати й про ймовірність витоку цієї інформації, як було наведено в прикладах Digital Identities вище.

Можливість децентралізації пропонує користувачу:

- Контроль над своїм digital identity
- Можливість ділитися лише певними даними з доступом, щоб відкликати їх від людини, якій ви довірили ці дані
- Монетизація своєї особистої інформації
- Незалежність від інших великих компаній, щоб уникнути ймовірності втрати інформації
- Можливість безслідно відкликати доступ до особистої інформації

1.2 Важливість Decentralized Identity для організацій

Важливість для організацій криється в безпеці. Враховуючи кількість бухгалтерії в наш час, підтвердження якогось документу може займати тижні або, навіть, місяці. Оскільки організація не зберігає у себе інформацію про користувачів, Decentralized Identity дозволяє верифікувати інформацію за лічені секунди. Компанія не має зв'язуватись з стороною, що видає документ (диплом про закінчення університету або водійське посвідчення), щоб підтвердити, що людина дійсно отримала цей документ і він є дійсним.

Також, майже моментальна верифікація унеможливлює підробку сертифікатів. Ця сфера досить популярна сьогодні. Підробка сертифікатів може відбуватись як з нуля (створення сертифікату з нуля), так і просто у виді заміни особистих даних у, вже існуючому, сертифікаті. Важко уявити як жити без можливості верифікувати такі сертифікати, оскільки професії можуть вимагати від працівників певних навичок для успішної роботи. Окрім верифікації вже готових сертифікатів, є можливість виписувати захищені сертифікати. Фейкові дипломи – досить велика індустрія, де виписати сертифікат людині не є проблемою. Зрозуміло, що це призводить до ризиків для людей через роботу в медичній сфері або ще щось більш ризиковане. За статистикою[1], навіть у 80-тих роках було приблизно 5 тис. несправжніх лікарів в США, а зараз очікується, що ще більше.

Щодо безпеки зберігання даних завдяки зберіганню за допомогою загального криптографічного ключа для шифрування та дешифрування інформація знаходиться під надійним захистом.

Знову ж таки, завдяки тому, що компанія не зберігає такої кількості інформації користувачів, то зловмисники не настільки зацікавлені в отриманні особистої інформації, банально, через витрати часу.

Згідно статистики, середня ціна витоку даних для маленького бізнесу сягає 108 тис. доларів в рік, а на виявлення слабкого місця компанія, в середньому витрачає 206 днів та ще 73 днів, щоб його усунити. Також малі бізнеси потерпають від витоків інформації через слабкі місця у 43% випадків. Для великих корпорацій середня ціна витоку інформації обходиться в 204 долари на працівника, коли для малого та середнього бізнеса вона обходиться в 3.5 тис. доларів для працівника.[1]

Важливість Decentralized Identity для розробників

Через використання Decentralized Identity полегшується створення проєктів зі зручним логіном та реєстрацією користувачів через digital identity. Тим самим, ми полегшуємо вхід користувача без використання паролів та імен користувачів. Не будемо забувати про можливість безпечно надсилати запит на отримання даних напряму від користувачів не забуваючи про їх безпеку і приватність.

1.3 Різниця між Decentralized Identity та Self-Sovereign Identity

По суті, це дві нероздільні технології, де SSI – це можливість користувача взаємодіяти з Digital Identity. Self-Sovereign Identity будується з 3-х частин:

- **Блокчейн**: база даних, яка розділена між декількома комп’ютерами в мережі, де користувач має децентралізований доступ. Інформація туди записується з шифрування і коли записується вже не підлягає редагуванню.
- **Verifiable Credentials:** документи користувача, які він може легально використовувати в повсякденному житті для аутентифікації чи підтвердження особистості, як офіційний документ. Через криптографічне шифрування дістати їх з цифрового гаманця майже неможливо.

- **Децентралізовані Ідентифікатори:** створені користувачем, криптографічно зашифровані дані, які не залежать від будь-якої організації. Ці ідентифікатори належать користувачу, але не містять в собі персональної інформації користувача, за якими можна ідентифікувати особу.

1.4 Decentralized Identity Management vs Centralized Identity Management

Найбільшою перевагою Decentralized Identity Management є незалежність від інших сервісів та компаній та контроль над особистою інформацією. На відміну від Centralized Identity Management користувачу не потрібно довірятися компаніям, щоб надавати свою персональні дані на зберігання і розраховувати, що вони не будуть використані у власних цілях компанії. В децентралізованій моделі, завжди є можливість відкликати доступ до даних.

Також, децентралізована модель менеджменту є більш безпечною, оскільки, навіть, при взломі даних одного користувача, зловмисник не зможе отримати доступ до інших завдяки використанню блокчейну.

У випадку централізованої системи менеджменту даними, користувач має надати всі дані і може стати об’єктом таргетованої реклами, що погіршує досвід використання застосунку або веб-сайту. З іншої сторони децентралізований спосіб дозволяє користувачу самому обирати якою інформацією ділитись, а яку залишити приватною.

Підсумовуючи, можемо побачити різницю між цими підходами в даній таблиці:

Centralized Identity Management	**Decentralized Identity Management**
Increased risk of data breaches from storing data in a centralized system	Data is decentralized and stored by users in their wallets, which reduces the risk of large scale data breaches

Centralized Identity Management	Decentralized Identity Management
Data may be collected, stored, and shared with other parties without your knowledge	Data is only shared when you give authorization
Data is owned and controlled by organizations, apps, and services	Data is fully owned and controlled by the user

Табл.1 Різниця між зберіганням централізованої та децентралізованої ідентичності

1.5 Як працює Decentralized Identity?

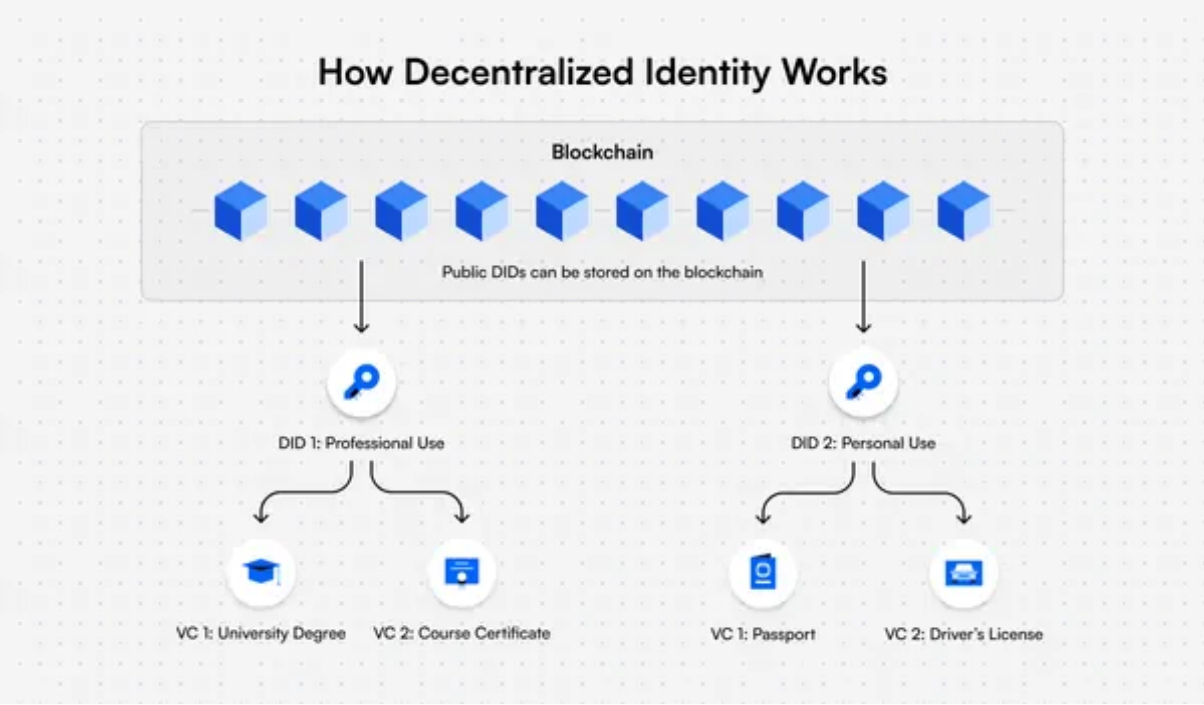


Рис 1. Огляд роботи Decentralized Identity

Створення decentralized identity складається з декількох пунктів, які є обов'язковими для успішної верифікації і зберігання даних.

Для зберігання DID (Decentralized Identifier) використовується блокчейн у вигляді децентралізованої бази даних. Незмінні записи лише допомагають зберегти інформацію незмінною і надійно вберегти її від зловмисників через спосіб зберігання.

За допомогою Decentralized Identity Wallet користувач має право створювати decentralized identifiers і мати доступ до своїх, вже підтверджених даними (verified credentials). Це є основним зв’язком між кінцевим користувачем та записами в блокчейні.

Сам по собі DID зберігається на блокчейні, як було згадано раніше. Це унікальний ідентифікатор або ключ, який створюється під час запиту користувача на верифікацію документу, складається з літер та чисел та криптографічно шифрується перед записом на блок. Ідентифікатор містить публічний ключ та інформацію про верифікацію документу.

І, нарешті, верифіковані облікові дані це криптографічно зашифрована версія документів, які користувач може надавати для верифікації іншими організаціями.

Процес верифікації документів виглядає таким чином, що користувач створює запит на верифікацію, ця інформація передається до компанії, що його верифікує, маючи доступ до DID на блокчейні, та переглядає чи даний документ дійсно був виданий компанією або закладом, що надав цей документ і який має бути підписаний приватним ключем цієї компанії.

Важливим моментом є те, що документи користувача не зберігаються на блокчейні. В блоці лежить лише DID цього документу з усією потрібною інформацією для верифікації.

Кожна верифікація та запис в системі блокчейну є у відкритому доступі тож будь-хто може перевірити чи дійсно була така перевірка, що також допомагає уникнути підробки документів.

2. DIDs (decentralized identifier)

2.1 Що таке Decentralized Identifier

Отже, Decentrilized Identifier це унікальний, незмінний ідентифікатор, який створюється криптографічним шляхом і який є незмінним. Окрім цього, ідентифікатор також не має бути централізовано зареєстрованим, тобто нема потреби в його реєстрації через інші компанії або органи реєстрації. Це спосіб ідентифікувати себе або щось в Інтернеті, не покладаючись на централізовану компанію чи організацію.[1]

Це як особистий номер телефону, який був присвоєний користувачу, який він повністю контролює і може використовувати, щоб довести його ідентичність не покладаючись на третю сторону.

Сам ідентифікатор складається з декількох частин. Це його синтаксис, DID контролер та DID тема.[2]

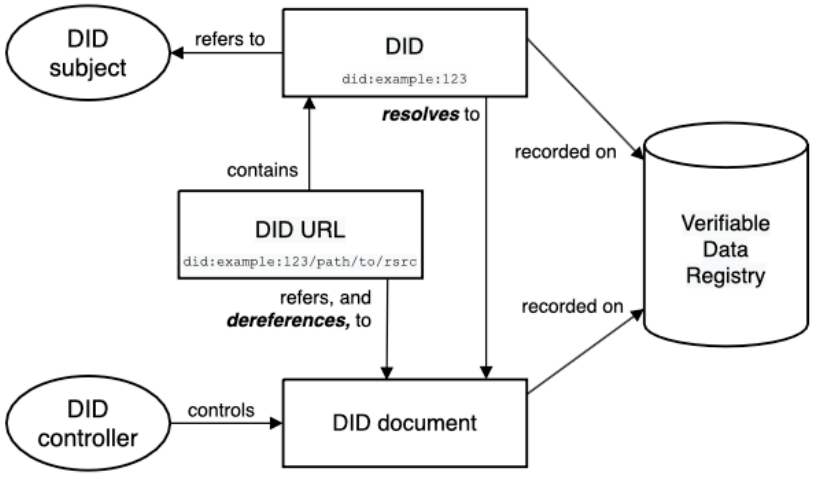


Рис. 2 Архітектура DID та базових зв'язків між компонентами

2.2 Правила синтаксису

Синтаксис DID схеми має відповідати правилам ABNF. Сам ідентифікатор складається з декількох частин.

Нижче наведена таблиця з правилами[2]:

The DID Syntax ABNF Rules

```
did                = "did:" method-name ":" method-specific-id
method-name        = 1*method-char
method-char        = %x61-7A / DIGIT
method-specific-id = *( *idchar ":" ) 1*idchar
idchar             = ALPHA / DIGIT / "." / "-" / "_" / pct-encoded
pct-encoded        = "%" HEXDIG HEXDIG
```

Рис. 3 Правила ABNF синтаксису DID

2.3 DID URL синтаксис

DID URL – це шлях у веб-мережі, який містить всю інформацію про конкретний DID на певному ресурсі. Використовуючи DID URL, користувач може отримати доступ до інформації про тему DID, методи верифікації, посилання на пов’язані ресурси та окремі елементи DID.

Нижче наведений приклад DID URL[2]:

```
did-url = did path-abempty [ "?" query ] [ "#" fragment ]
```

Цей синтаксис також є обов’язковим для дотримання при створенні DID URL.

Компонент path містить дані, зазвичай організовані в ієрархічній формі, які разом з даними в неієрархічному компоненті query, слугують для ідентифікації ресурсу в межах схеми URI і повноважень на присвоєння імен. [3]

Аналогічно працює і компонент query, містячи в собі дані компоненту path для ідентифікації URI.

Частина DID “fragment” фрагмента URI дозволяє ідентифікувати вторинний ресурс через посилання на первинний ресурс і додаткову ідентифікаційну інформацію.[3] Цей вторинний ресурс може бути деякою частиною або підмножиною первинного ресурсу, деяким поглядом на

представлення первинного ресурсу або деяким іншим ресурсом, визначеним або описаним цими представленнями.

2.4 Параметри DID

Параметри DID URL відіграють ключову роль у модифікації чи уточнені запитів до ідентифікаторів. Також, вони можуть бути використаними для DID-resolvers, наприклад, для повернення конкретних даних з DID.

Існують універсальні параметри – які працюють для всіх методів DID однаково, а також які підтримуються не всіма методами.

Параметри не є обов'язковим компонентом DID URL, тож за можливості, коли однакового результату можна досягти без їх додавання, слід уникати їх використання.

Додавання компоненту query робить його частиною ідентифікатору для окремого ресурсу та може містити параметри, які підтримуються DID URL синтаксисом.[2]

Розглянемо універсальні параметри, які користувач може використати в будь-якому DID методі[2].

- service – ідентифікує сервіс з DID-документа за ідентифікатором сервісу.
- relativeRef – відносне посилання на URI, що ідентифікує ресурс на кінцевій точці сервісу, який вибирається з DID-документа за допомогою параметра service.
- versionId – ідентифікує конкретну версію DID-документа, що підлягає обробці.
- versionTime – ідентифікує певну мітку часу версії DID-документа, що підлягає перетворенню. Тобто, документ DID, який був дійсним для DID в певний час.

- hl – ресурсний хеш документа DID для додавання захисту цілісності. Не є обов’язковим до використання.

2.5 DID subject

В кореневій частині DID документу має міститись id типу String для конкретного DID subject та таким чином визначати DID.

```
{ "id": "did:example:123456789abcdefghijk" }
```

Деякі DID методи можуть створювати представлення DID документу, в яких id документу може бути відсутнє: DID resolver, DID resolution. [2] Тим не менш, фінальний DID документ обов’язково має мати це поле.

2.6 DID controller

DID controller – це інструмент, завдяки якому можна вносити зміни до DID документа.[2] DID controller не є обов’язковим, але якщо є, то відповідний DID документ має містити зв’язки з явним вказанням методів верифікації для конкретних випадків. Відповідно, сама авторизація контролера визначається DID методом.

Властивості контролера визначаються в DID документі, а значення виражає як мінімум один DID. Будь-які методи верифікації, що містяться в DID документах для цих DID, повинні бути прийняті як авторитетні, так що докази, які задовольняють цим методам верифікації, повинні вважатися еквівалентними доказам, наданим суб'єктом DID. [2]

Нижче наведений приклад DID документу з властивістю controller: [2]

```
{
"@context": "https://www.w3.org/ns/did/v1",
"id": "did:example:123456789abcdefghi",
"controller": "did:example:bcehfew7h32f32h7af3",
}
```

2.7 Centralized identifiers VS DID

На даний момент централізовані ідентифікатори такі як наші електроні пошти, паролі та ім’я користувача щоб увійти в акаунт на будь-якому сайті, застосунку або сервісі. Але, як ми знаємо, це призводило до втрати персональної інформації, підробці даних або використання нашої особистої інформації без нашого відома для власних цілей. Ну і не варто забувати про незручності користування таким типом логіну, оскільки користувачам варто кожного разу придумувати нові паролі та ім’я користувача, задля того, щоб максимально убезпечити себе від взлому акаунтів.

В той же момент, використання децентралізованих ідентифікаторів позбавляє нас цих проблем, оскільки він є глобально унікальним та випадково згенерованим, зберігається на блокчейні та є незалежним від будь-яких організацій. Також, він є універсальним способом підтвердити свою ідентичність, але, в той же час, не містить приватну інформацію. Оскільки зберігається на блокчейні, то дозволяє безпечно створювати зв’язки між двома організаціями і може бути підтвердженим будь-де та будь-коли.

3. Verifiable Credentials

3.1 Що таке Verifiable Credentials?

Verifiable Credentials – це електронні, криптографічно захищені документи. Під це визначення можуть підпадати як копії електроних так і фізичних документів, які валідні для використання задля підтвердження власної ідентичності. Як зазначалося раніше, це може бути посвідчення водія, закордонний паспорт, диплом про здобуття вищої освіти тощо.

Для того, щоб документ вважався Verifiable Credenial потрібно, щоб він відповідав дата моделі Verifiable Credential Data Model 1.0, яка була затверджена World Wide Web Consortium (W3C).[4]

3.2 Як пов’язані VCs та DID?

Насправді, Verification Credentials та DID не так сильно пов’язані. Використання Verified Credentials з DID покращує різноманітність верифікацій, оскільки DID можна використовувати для всіх видів верифікації, а також можливість використання ключів різних типів з DID.

Незважаючи на це користувач може використовувати VC без DID і це працюватиме як для органа, що видав цей Verifiable Credential, так і для організації, що його верифікує. Вони також працюють з централізованими і локальними ідентифікаторами, адресами блокчейну або смарт-контрактами тощо.

3.3 Види VCs

Як таких видів VC не існує, лише спосіб його інтерпретації. Ці інтерпретації можна трактувати як різні представлення для певної мови або операційної системи. Кожен з них має свої плюси та недоліки, які ми розглянемо згодом.

Нижче наведено список найголовніших та найбільш вживаних типів VCs[5]:

- JSON-LD з LD Signature або з BBS+ Signature
- JSON з JSON Web Signatures у формі веб-токена JSON (JWT)
- ZKP з підписами Каменіша-Лисянської

Також, існує два типи Verification Credentials, які концептуально відрізняються один від одного, способом представлення моделі даних VC і, пов’язані з нею, матеріали.

- VCs представлені у форматі JSON-LD використовуючи Linked Data Proofs
- VCs представлені у форматі JSON використовуючи JWS у формі JWT

Обидва типа інтерпретації VCs підтримують багато різних алгоритмів електронного підпису, що дає зрозуміти наскільки Verified Credential є новітнім та універсальним способом для використання документів.

Третім зазначеним видом VC є формат з підписом Каменіша-Лисянської. Цей формат є спеціальним форматом подачі даних з певним алгоритмом електронного підпису.

Як правило, використовується JSON-LD, оскільки він може легко бути переведеним в інший формат та є зрозумілим, оскільки, по суті, представляє відомий всім формат JSON.

3.4 Тонкощі використання VCs

Незважаючи на те, що формат JSON-LD дуже схожий на JSON, він має певні аспекти, які є критичними при його створенні.

Головним аспектом є використання контексту (@Context), в якому буде створюватись даний JSON-LD об’єкт.[4] Цей контекст задає більш чітке розуміння які змінні та їх типи варто очікувати під час верифікації документу. Це покращує сприйняття об’єктів, оскільки більш чітко дає

зрозуміти, яку семантику варто очікувати і надає можливість легше переходити між форматами представлення Verified Credential.

Отже, JSON-LD – це спосіб представлення Verification Credentials, які дозволяють кодувати складну, інколи, семантично-неоднозначну інформації у, всім відомому форматі, JSON, з яким найчайстіше взаємодіють. Не дивлячись на це, формат JSON-LD працює кардинально інакше. За допомогою RDF можна компонувати базову, графо-структуровану інформацію і переміщатися по ній, як по графу[4]. Насправді, версія LD-документа, яку було перетворено на RDF (або, принаймні, яку можна розглядати як RDF за допомогою стандартного інструментарію RDF), називається "розширеною версією".

3.5 Приклад верифікації документу

Розглянемо приклад верифікації певного документу. Уявімо, що в нас є людина, яка прийшла працювати в таксі і, відповідно, для роботи їй потрібне посвідчення водія. Компанія, яка наймає водія, може миттєво перевірити цей верифікований документ, що посвідчення водія є валідним.

Використання в даному прикладі верифікованого документу є плюсом, оскільки водій не має поширювати приватну інформацію, таку як повне ім’я або адресу проживання.

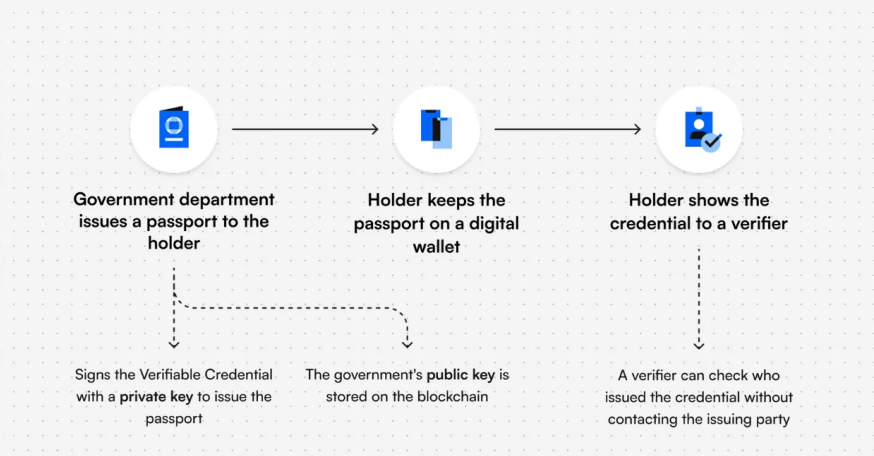


Рис.4 Шлях проходження верифікації документу

Іншим прикладом є підтвердження повноліття в певних закладах. Працівник закладу може сканувати QR-код верифікованого документу гістя, щоб впевнетись, що йому є вісімнадцять років. Децентралізований ідентифікатор ліцензуючої особи зберігається в блокчейні, тож заклад, за умови, що довіряє органу, який видав документ, може перевірити валідність документу. Верифіковані документи, які можна перевірити, створюють довіру між сторонами і гарантують автентичність даних і заяв без фактичного зберігання даних у блокчейні.

4. Decentralized Identity Wallet

Враховуючи розвиток DID, користувачам потрібен сервіс через який буде легко використовувати verified credentials: як процес верифікації, так і подальшого використання.

Ринок цифрових гаманців є досить розвинутим у Європі: Dock.io, Gataca, Altme. Це зумовлено як попитом користувачів, так і вимогами консорціуму, який займається сертифікацією цифрових гаманців у Євросоюзі.

В березні 2024-го року рада Євросоюзу внесла зміни до регуляції European digital identity. Зміни полягали в покращені нормативно-правової бази та покращення для користувачів в різних сферах. Гаманець міститиме інформаційну панель усіх транзакцій, доступних його власнику як онлайн, так і офлайн, надаватиме можливість повідомляти про можливі порушення захисту даних та забезпечуватиме взаємодію між гаманцями. [6] Також, громадяни зможуть використовувати гаманець з вже існуючими національними сервісами для ідентифікації та електроні підписи для непрофесійного використання.

Основні елементи переглянутого закону можна підсумувати наступним чином[6]:

- до 2026 року розробити кожною державою-членом цифровий ідентифікаційний гаманець. Гаманець має бути у відкритому доступі для своїх громадян і у громадян має бути можливість використовувати будь-які інші European Digital Identity Wallets з інших держав-членів з аналогічними можливостями та рівними правами
- користування гаманцем має бути добровільним і особи, які його не використовують, не мають бути будь-яким чином дискриміновані

- усі процеси пов’язані з видачею, використанням та видаленням гаманців мають бути доступні всім – безкоштовними
- створити відповідність між гаманцем, у виді форми електронної ідентифікації та схемою, за якою він видається
- забезпечити державам безкоштовну валідацію гаманців, які використовує користувач і особу, яка його використовує
- код гаманців має бути відкритий, щоб будь-хто міг перевірити чи розробники не використовують надмірно доступи, які їм надає користувач. Натомість, розробники мають право прикривати доступ до певних частин програми, окрім тих, що встановлені на пристроях користувачів, знову ж таки, заради безпеки

Нарешті, переглянутий закон уточнює сферу застосування кваліфікованих сертифікатів автентифікації веб-сайтів (QWAC), що гарантує, що користувачі можуть перевірити, хто стоїть за веб-сайтом, зберігаючи при цьому чинні усталені галузеві правила та стандарти безпеки.[6]

4.1 Імплементація цифрових гаманців

Для того, щоб створювати власний цифровий гаманець компанії мають посилатись на pilot задля впевненості, що гаманець відповідає правилам Євросоюзу та буде цілком відповідати очікуванням користувачів.

Для цього у 2023-му році Європейський Союз випустив чотири пілотні програми, поки що, у закритому доступі для усіх користувачів, для оцінки цифрового гаманця ЄС до його офіційного запуску.[7] Ці проєкти також будуть допомагати розробникам створювати власні імплементації та збирати відгуки щодо вимог цифрового гаманця ЄС.

Комісія буде надавати компаніям вже готовий прототип digital identity wallet (EUDI), який відповідає вимогам регламенту про Європейську

Цифрову Ідентичність. Прототип міститиме потрібні бібліотеки та зразок програми.

Саме створення власного гаманця на основі EDIW допомагає отримати цінну інформацію від користувачів і отримати незалежну оцінку функціональності та зручності використання, оскільки пілотні проєкти позиціонуються для більшості сфер: освіта, фінансові послуги, реєстраційні послуги. Ще однією зручністю є те, що прототипи міститимуть усі потрібні ресурси задля створення власного цифрового гаманця.

4.2 Основні пілоти

У травні 2023 року було запущено чотири масштабні пілотні проєкти з метою тестування цифрового ідентифікаційного гаманця Євросоюзу та забезпечення його безпечного і безперешкодного запуску. У цих пілотних проєктах беруть участь близько 360 організацій, включаючи приватні компанії та органи державної влади з 26 держав-членів ЄС, Норвегії, Ісландії та України.[7] Проєкти розраховані на дію до 2025-го року з фінансуванням грантами від Європейської Комісії та керуються технічним стандартам, розроблені експертною групою eIDAS. Тобто, кожен з цих проєктів має структуру консорціуму, де враховується бачення як державного, так і приватного сектора Євросоюзу.

Пілоти, окрім того, що створені для тестування, будь-яких, юзер-кейсів, розроблені для збору інформації та відгуків від користувачів власних імплементацій цифрових гаманців, які потім можуть бути використані для покращення безпеки, сумісності та, загалом, дизайну імплементацій EU Digital Wallet.

Розглянемо детальніше ці чотири пілотні проєкти:

The EU Digital Identity Wallet Consortium (EWC) – це проєкт запущений з метою покращення ситуації щодо цифрових проїзних в Євросоюзі для всіх

його держав-членів. Як і інші, він сфокусований на використанні переваг EDIW.

POTENTIAL консорціум сфокусований на таких сферах – державні послуги, фінансові послуги, телекомунікація, водійські посвідчення, електронні підписи та медичні послуги

NOBID – це група країн Північної Європи та Балтії, які разом з Італією та Німеччиною створюватимуть власний Digital Identity Wallet з метою авторизації платежів за товари та послуги

DC4EU займається реалізацією гаманців для сфери освіти та соціальних послуг через використання інфраструктури цифрових послуг як для державного так і приватного сектору

Загалом, кожний пілот позиціонується для вирішення стандартних задач для цифрових гаманців.

Це включає в себе такі сфери:

1. Доступ до державних послуг: безпечний доступ до цифрових державних послуг, а саме отримання посвідчення водія, соціальні послуги, отримання паспорта, а також сплата податків
2. Відкриття банківського рахунку: верифікація користувача при відкритті банківських рахунків без потреба повторно надавати приватну інформацію
3. Реєстрація SIM-карти: верифікація особи з метою укладення контрактів на перед- та післяоплачені SIM-картки (реєстрація та активація). Таким чином можна досягти зменшення шахрайства та, відповідно, витрат для операторів мобільного зв'язку
4. Водійське посвідчення: зберігання та пред'явлення цифрового водійського посвідчення як онлайн, так і під час надання посвідчення відповідним особам для верифікації

5. Підписання контрактів: можливість підписувати контракти цифровими підписами без потреби в фізичних документах та підписах
6. Отримання рецептів на ліки: надання цифрових рецептів в аптеку для видачі ліків
7. Подорожі: надання інформації з проїзних документів, наприклад, паспорт або віза, що забезпечує швидкий і легкий доступ при проходженні контролю безпеки на митницях
8. Платежі: перевірка особи користувача при створені платежу
9. Сертифікація освіти: підтвердження наявності документів про освіту: дипломи, наукові ступені та сертифікати
10. Доступ до соціальних виплат: European Digital Identity Wallet можна використовувати для безпечного доступу до інформації про соціальне забезпечення та виплати, наприклад, пенсії, допомоги по інвалідності. Він також може бути використаний для забезпечення свободи пересування шляхом зберігання таких документів, як Європейська картка медичного страхування

5. European Blockchain Services Infrastructure

European Blockchain Services Infrastructure(EBSI) – це ініціатива європейської комісії, в рамках European Blockchain Partnership (EBP), по створенню цифрових гаманців. EBSI вже вирішує питання безпеки, оскільки вже включає в себе використання блокчейну для зберігання інформації. EBP є основною ініціативою розвитку блокчейн стратегії в Євросоюзі.

EBP відповідає за координацію розвитку Європейської інфраструктури блокчейн-послуг (EBSI) між країнами-членами та забезпечення того, щоб вона відповідала потребам державних служб по всьому Євросоюзу. Згідно інформації[8], з боку Європейської Комісії, DG CNECT виступає в якості власника рішення, в той час як технічною реалізацією займається DG DIGIT.

EBSI відстежує чи програма відповідає правилам та процедурам, які дивиться за прозорістю та відкритою взаємодією між зацікавленими сторонами. EBP, в свою чергу, здійснює нагляд за процесом управління, що передбачає консультації між державами-членами та іншими організаціями, які пов’язані з роботою цього сервісу.

Спочатку для дослідження було розроблено European Blockchain Partnership для розуміння чи є потреба у використанні блокчейну для пересічної людини. Зважаючи на досвід, у 2020-му році EBSI запустило перший прототип веб-гаманця і вже у 2021-му році його можуть використовувати 21-а публічна організація у формі ранніх тестувальників. [8]

Вже у 2023-му році EBSI запускається у відкритий доступ з вже доступними пілотами для розробки власних гаманців. Популярність Digital Identity у світі зростає та все більше громадян починаються користуватися цифровими гаманцями у всіх сферах існування.

5.1 Складові EBSI

Усі варіанти EBSI працюють однаково. Базова структура складається з трьох частин: APIs для підключення користувачів, смарт-контракти для взаємодії API та спеціального сховища. Сховище – це децентралізована база даних для налагоджування бізнес процесів.

Розглянемо складові детальніше:

API: власні API від EBSI для зручної взаємодії з користувачем, а також з відповідним блокчейном, де зберігаються транзакції, які є унікальними для кожного варіанту використання EBSI. Для отримання доступу до API, треба подавати заявку в залежності від типу цифрового гаманця та для яких сфер розробник створює власний гаманець.

Смарт-контракти: смарт-контракт це тип «цифрового контракту», який приймається та виконується лише за умови виконання певних правил. У EBSI смарт-контракти строго контролюються та не можуть бути запущені будь-ким, окрім запитам зробленим з EBSI API з відповідних застосунків, які були підтвердженні Європейською Комісією як ті, що верифіковані.[9] Ці смарт контракти дозволяють робити записи на блокчейні.

EBSI журнал: EBSI сховище це децентралізована база даних для запису усіх транзакцій. Інформація зберігається на блоках, які криптографічно зав’язані один на одному утворюючи ланцюжок, що робить інформацію майже неможливою для фальсифікації. Користувач може використовувати реєстри сховища, при використанні зазначених випадків використання, які можуть слугувати джерелом довіри. Наприклад, реєстр довірених емітентів EBSI може використовуватися в освітній сфері для ведення захищених від підробки записів про всі університети, акредитовані національним органом освіти, що підвищує довіру та значно полегшує перевірку дипломів. [9]

6. Імплементація власного цифрового гаманця

Створення власного цифрового гаманця було вирішено робити під операційну систему iPhone Operating System (iOS). Компанія Apple має гарну документацію по своїм мовам програмування та фреймворкам, тож вибір пав саме на цю платформу.

6.1 Мова програмування

В основному для розробки застосунків під iOS використовується або мова програмування Swift, або Objective-C. Objective-C зараз вважається, скоріше, застарілою, оскільки у 2014-му році Apple презентувала Swift на Apple Worldwide Developer Conference (WWDC).

Swift – це сучасна, строго типізована, компільована мова програмування, яка була розроблена для заміни Objective-C. Порівняно з іншими мовами, Swift пропонує лише потрібний інструментарій для розробника.

Безпека з пам’яттю в Swift теж гарно пропрацьована, оскільки Swift використовує Automatic Reference Counting (ARC). ARC автоматично звільняє пам'ять, яку використовують екземпляри класів, коли ці екземпляри більше не потрібні. ARC менш вимогливий до продуктивності процесора, ніж збирач сміття.[10]

Єдиним мінусом використання Swift є те, що він дуже швидко розвивається та деякі технології, які були додані раніше, швидко стають застарілими та в наступних версіях iOS є небажаними для використання.

Через деякий час Apple видала у відкритий доступ вихідний код Swift, зробивши його доступ для усіх розробників, які можуть вносити свої правки та зауваження, тим самим покращуючи Swift для інших розробників. Не варто недооцінювати цей факт, оскільки через це ця мова є одною з тих, що найшвидше розвивається.

6.2 Фреймворки

Зрозуміло, що Swift це лише мова програмування, але для створення, безпосередньо, застосунків потрібні інші інструменти. Такими інструментами є user interface (UI) фреймворки, в які входять SwiftUI та UIKit.

UIKit є старішим фреймворком для Swift, представлений влітку 2013-го року. Розробник взаємодіє з View та відповідною ViewModel для цієї View. View представлена у вигляді графічного інтерфейсу (варіант Storyboard), де користувач може додавати потрібні елементи, просто перетягуючи потрібні, зі списку доступних. Це є досить корисним та зручним для початкових розробників, але тонкощі налаштування кожного елементу теж є важливими, тож не слід недооцінювати потужність цього фреймворку. Оскільки він був випущений раніше, він має багато вирішень помилкових ситуацій та є більш надійним за SwiftUI, а також підтримує старіші версії iOS, а саме з iOS 2.0+.

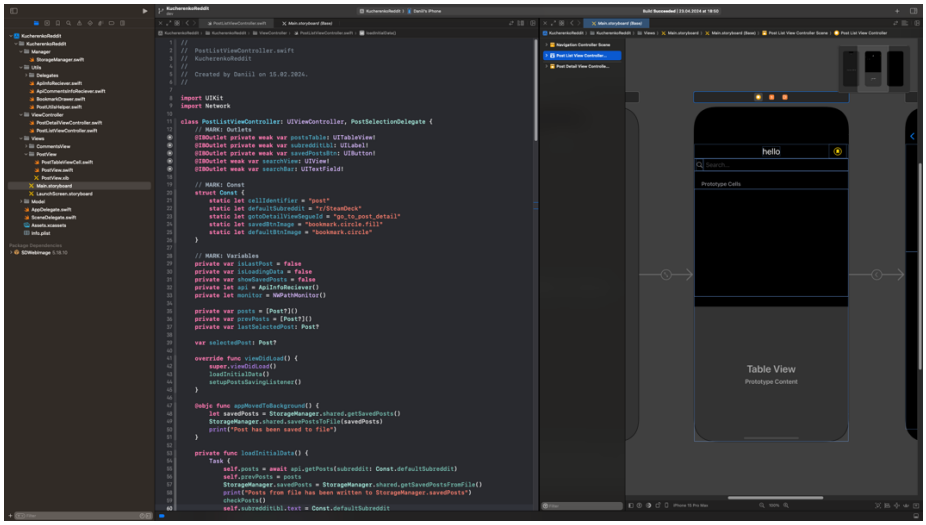


Рис.5 Приклад використання UIKit через Storyboard

Щодо недоліків такого варіанту використання UIKit можна віднести можливі конфлікти злиття (merge conflicts) у разі роботи в команді, оскільки Storyboard це, по суті, XML файл, який може по-різному форматуватись і створювати помилки. Саме через це, розробники можуть використовувати

створення View через їх опис кодом. Усі властивості задаються таким само чином, але такий варіант верстки є більш надійним та більш наочним, оскільки кожну зміну елементу можна відслідкувати в коді.

```swift
import Foundation
import UIKit

class CustomView: UIView {
    let imageView: UIImageView = {
        let view = UIImageView()
        view.translatesAutoresizingMaskIntoConstraints = false
        view.image = UIImage(systemName: "star")
        return view
    }()

    let descriptionLabel: UILabel = {
        let label = UILabel()
        label.translatesAutoresizingMaskIntoConstraints = false
        label.text = "My Favorite Quote"
        label.textColor = .white
        return label
    }()

    override init(frame: CGRect) {
        super.init(frame: frame)
        setupView()
    }

    // If you load view from storyboard
    // If you load view programmatically, this own't be loaded
    required init?(coder: NSCoder) {
        super.init(coder: coder)
        setupView()
    }

    func setupView() {
        backgroundColor = .brown
        addSubview(imageView)
        addSubview(descriptionLabel)

        NSLayoutConstraint.activate([
            imageView.widthAnchor.constraint(equalToConstant: 24),
            imageView.heightAnchor.constraint(equalTo: imageView.widthAnchor),
            imageView.leadingAnchor.constraint(equalTo: leadingAnchor, constant: 12),
            imageView.centerYAnchor.constraint(equalTo: centerYAnchor),
            descriptionLabel.leadingAnchor.constraint(equalTo: imageView.trailingAnchor, constant: 12),
            descriptionLabel.trailingAnchor.constraint(equalTo: trailingAnchor, constant: -12),
            descriptionLabel.topAnchor.constraint(equalTo: topAnchor, constant: 16),
            descriptionLabel.bottomAnchor.constraint(equalTo: bottomAnchor, constant: -16)
        ])
    }
}
```

Рис.6 Приклад використання UIKit зі створенням View кодом

Класичним шаблоном проєктування для UIKit вважається Model-View -Controller (MVC), але через велику сферу відповідальності View Controller'а для кожної View, розробники можуть частіше використовувати патерн Model-View-ViewModel (MVVM). Таким чином View Controller займає менше обсягу і не є головним в обробці логіки дій користувача на View, відповідно при потребі внести якісь зміни код не буде таким крихким, а виконує лише відстеження логіки презентації View.

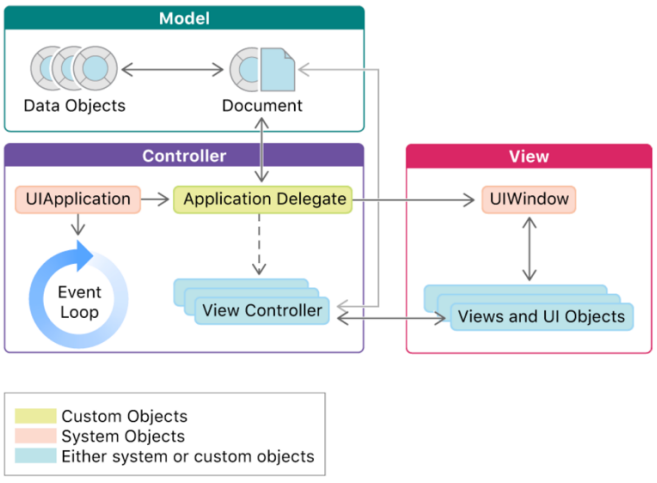


Рис.7 Схема взаємодії елементів патерну MVC у Swift

SwiftUI, в свою чергу, є декларативним фреймворком, який більш зрозуміло дає розробнику використовувати властивості SDK. Використовуючи SwiftUI, у розробника вже немає варіанту використовувати якісь графічні помічники – дозволено писати лише кодом, а також можна використовувати лише один варіант розміщення елементів на екрані, на відміну від UIKit.

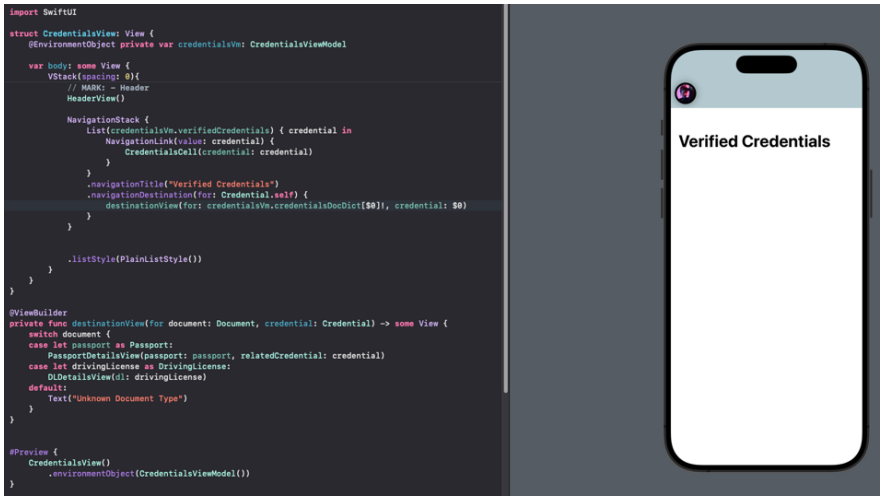


Рис.8 Приклад використання SwiftUI

Здавалось би, що SwiftUI виглядає легше і розробникам варто повністю на нього перейти, але, на жаль, не все так просто. Почнемо з того, що цей фреймворк підтримують операційні системи iOS лише з 13-ї версії. Також, у фреймворку можуть бути проблеми з рівнем модифікації стандартних елементів, а саме налаштування потрібних розробнику тонкощів в конкретних випадках. Це може відноситись як до навігації, так і до банального використання List, який не є гарно модифікованим. Саме тому, в більшості випадків, коли задача не є простою, розробники все ще використовують UIKit.

Можна побачити, що тут розробник вже описує як має виглядати поле body. Це обраховувана змінна, яка має міститись у всіх SwiftUI View. Далі, вкладаючи в цю змінну інші елементи утворюється наш екран, який ми можемо бачити у Preview справа без обов’язкового запуску програми як це було з UIKit.

Головним плюсом при розробці застосунку є те, що розробнику необов’язково обирати конкретний фреймворк, на якому він буде писати, оскільки кожен з цих фреймворків має можливість використання одного в іншому через допоміжні структури.

6.3 Способи взаємодії з користувачем

В реальному цифровому гаманці, який написаний на основі EU Digital Wallet, звернення йде до свого API. Найчастіше використовувався EBSI API для якого потрібен спеціальний токен для доступу. В цьому випадку був використаний власний API, створений на основі сервера на NodeJS з потрібними ендпоінтами. Ендпоінт – це адреса для звернення користувача до сервера задля отримання або надсилання інформації.

Для цього була використана бібліотека Express, яка надає інструменти для простої реалізації такого серверу.

6.4 Процес розробки

Отже, було обрано розробляти застосунок саме на SwiftUI. Як було зазначено раніше, для власного API був використаний NodeJS. Застосунок представляє собою 2 вкладки. Одна з них, це підтверджені документи (Дод. А) , які зберігаються на акаунті користувача. Інша сторінка – можливість додати нові документи (Дод. Б) (в нашому випадку це водійське посвідчення та паспорт).

При верифікації документу, користувачу потрібно перейти на другу вкладку та заповнити формочку з відповідними полями (Дод. В). Після цього на сервер надсилається POST запит, використовуючи вбудовані методи Swift, який при використані EBSI має по-справжньому верифікуватись. (Дод. Г)

Після верифікації, у користувача є можливість презентувати документ у вигляді QR коду (Дод. Д) підтверджуючи потрібну інформацію або переглянути більш детальну інформацію про відповідний документ. (Дод. Ґ)

Відповідно, користувач також може отримувати запити від інших користувачів, щоб той підтвердив певну інформацію про себе надаючи поля документу (Дод. Е). Реалізація спілкування між сервером і клієнтам зроблена через метод long polling. Він полягає в тому, що клієнт слухає сервер і коли на сервері з’являється запит на клієнт, надсилається запит від серверу з потрібною інформацією. Після цього з’являється форма з полями, який користувач може обрати для поширення і в подальшому відправити на сервер.(Дод. Є)

Висновок

Після виконаної роботи було детально розглянуто механізм створення Digital Identity Wallet’a. Також був проведений аналіз щодо подання та зберігання інформації на блокчейні та весь процес підтвердження документів.

Було проаналізовано що таке Decentrilized Identity його варіанти. Розглянуто що таке Decentralized Identifier, структуру зберігання на блокчейні та види представлення Verifiable Credentials.

Проаналізований EBSI і його механізми взаємодії з цифровими гаманцями. На його основі був розроблений власний прототип цифрового гаманця для мобільних пристроїв.

Після аналізу було підтверджено значення використання Digital Identity в наш час та актуальність розвитку цієї галузі, що дозволить користувачам більш безпечно зберігати свою приватну інформацію у мережі. Це також дає, в майбутньому, використовувати DI як аналогічний спосіб підтвердження особи будь-де.

В подальшому планується розвиток взаємодії між власними цифровими гаманцями для користувачів різних ролей: орган, що видає документ, орган, що підтверджує валідність документу та звичайний користувач.

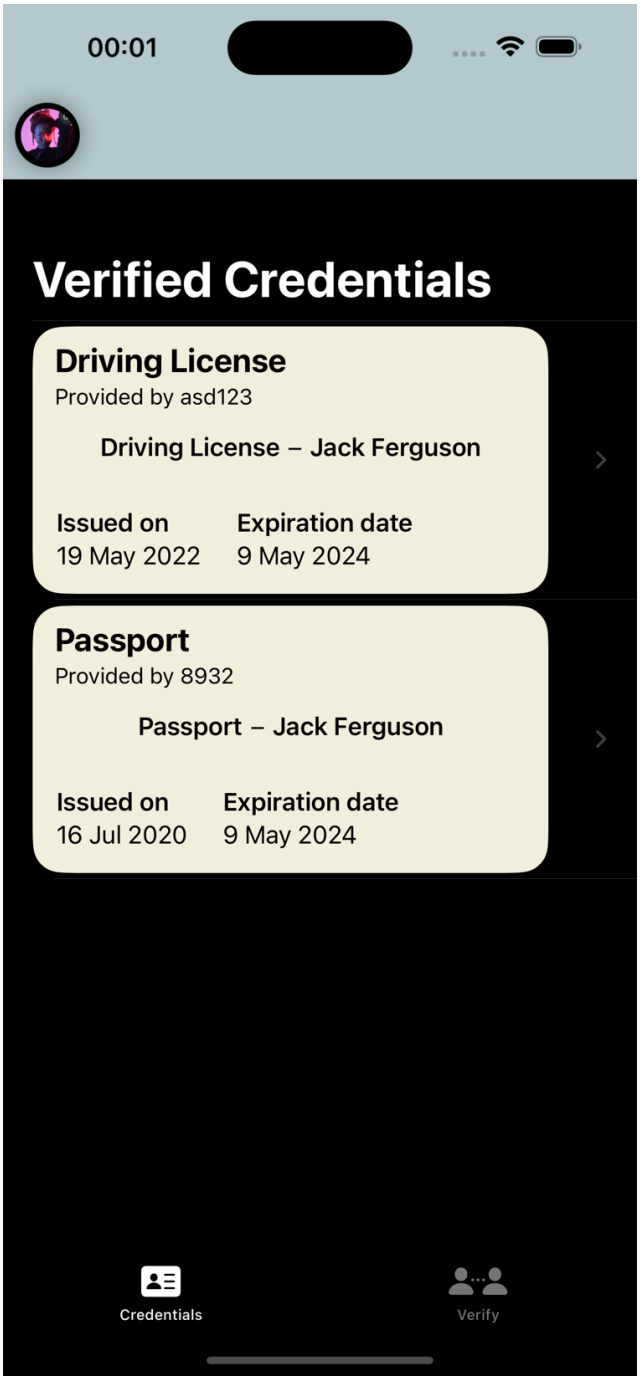
Джерела

1. Dock.io. Decentralized Identity: The Ultimate Guide [Електронний ресурс] / Dock.io. – 2024. – Режим доступу до ресурсу: https://www.dock.io/post/decentralized-identity
2. Decentralized Identifiers(DIDs) v1.0 [Електронний ресурс] / [M. Sporny, D. Longly, M. Sabadello та ін.]. – 2022. – Режим доступу до ресурсу: https://www.w3.org/TR/did-core/#identifiers
3. Berners-Lee T. Unifourm Resource Identifier (URI): Generic Syntax [Електронний ресурс] / T. Berners-Lee, R. Fielding, L. Masinter. – 2005. – Режим доступу до ресурсу: https://www.rfc-editor.org/rfc/rfc3986#section-3.5
4. Caballero J. Decentralized Identity FAQ [Електронний ресурс] / Juan Caballero – Режим доступу до ресурсу: https://identity.foundation/faq/#vc-infrastructure-layer-3
5. Young K. Verifiable Credentials Flavors Explained [Електронний ресурс] / Kaliya Young // LPFH. – 2021. – Режим доступу до ресурсу: https://www.lfph.io/wp-content/uploads/2021/02/Verifiable-Credentials-Flavors-Explained.pdf.
6. European digital identity (eID): Council adopts legal framework on a secure and trustworthy digital wallet for all Europeans [Електронний ресурс] // European Consillium. – 2024. – Режим доступу до ресурсу: https://www.consilium.europa.eu/en/press/press-releases/2024/03/26/european-digital-identity-eid-council-adopts-legal-framework-on-a-secure-and-trustworthy-digital-wallet-for-all-europeans/
7. EU Digital Identity Wallet Pilot Implementation [Електронний ресурс] // European Comission. – 2024. – Режим доступу до ресурсу: https://digital-strategy.ec.europa.eu/en/policies/eudi-wallet-implementation

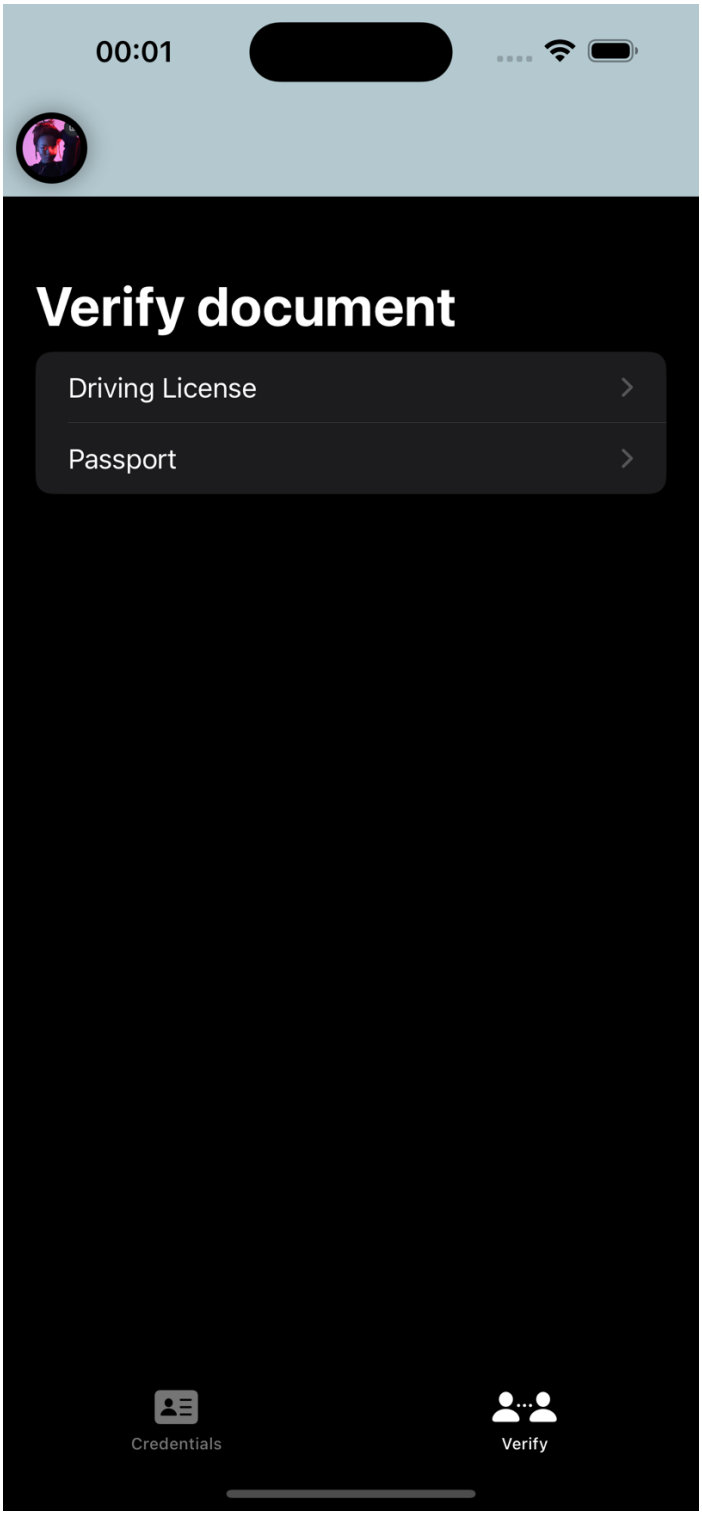
8. EBSI. About Us [Електронний ресурс] // EBSI. – 2024. – Режим доступу до ресурсу: https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/About+us.
9. What is EBSI? [Електронний ресурс] // EBSI. – 2024. – Режим доступу до ресурсу: https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/What+is+ebsi.
10. Fojtik R. Swift a New Programming Language for Development And Education [Електронний ресурс] / Rostislav Fojtik. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/338081271_Swift_a_New_Programming_Language_for_Development_and_Education.

Додатки

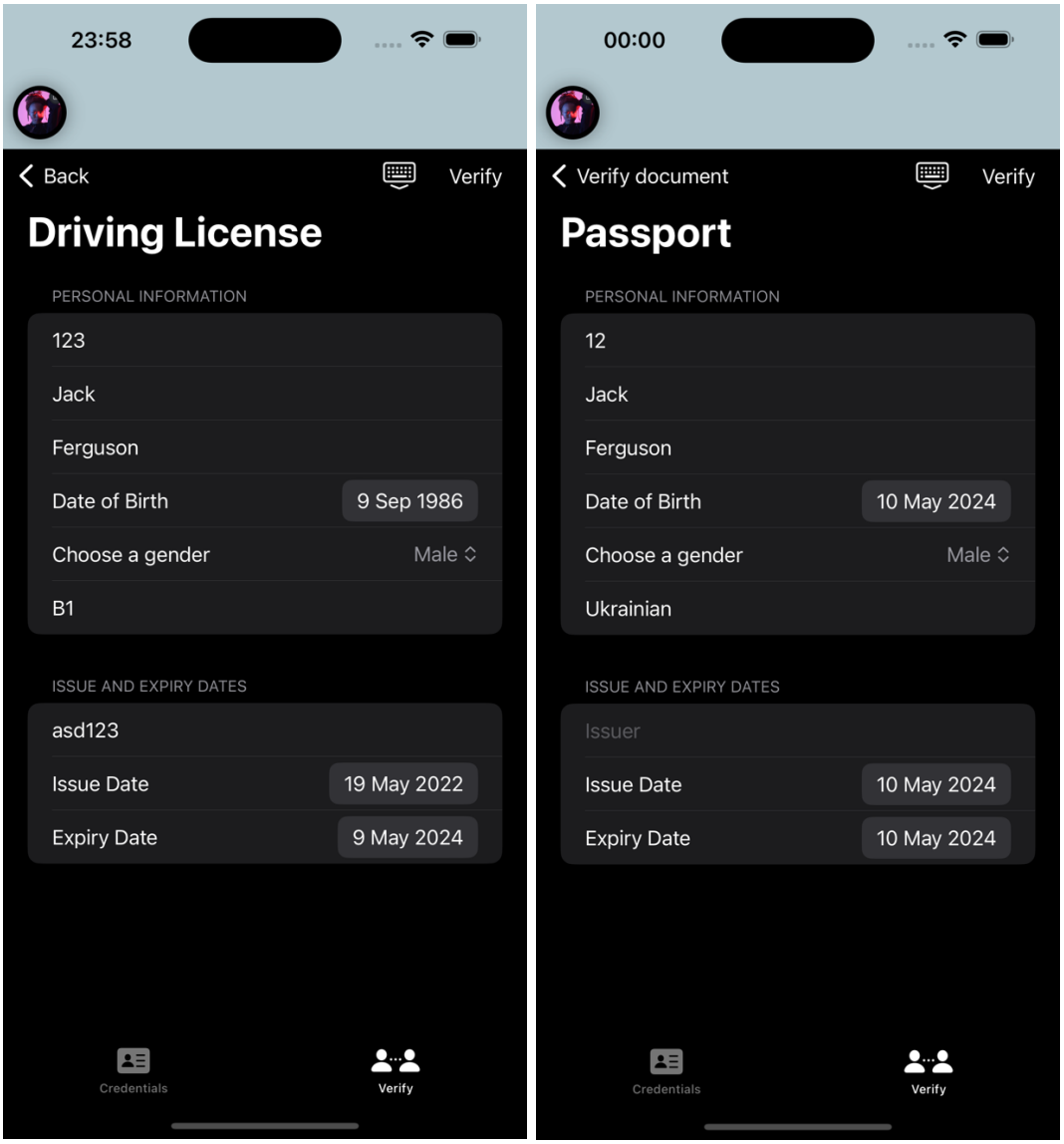
Додаток А. Головний екран Credential View з усіма підтвердженими документами



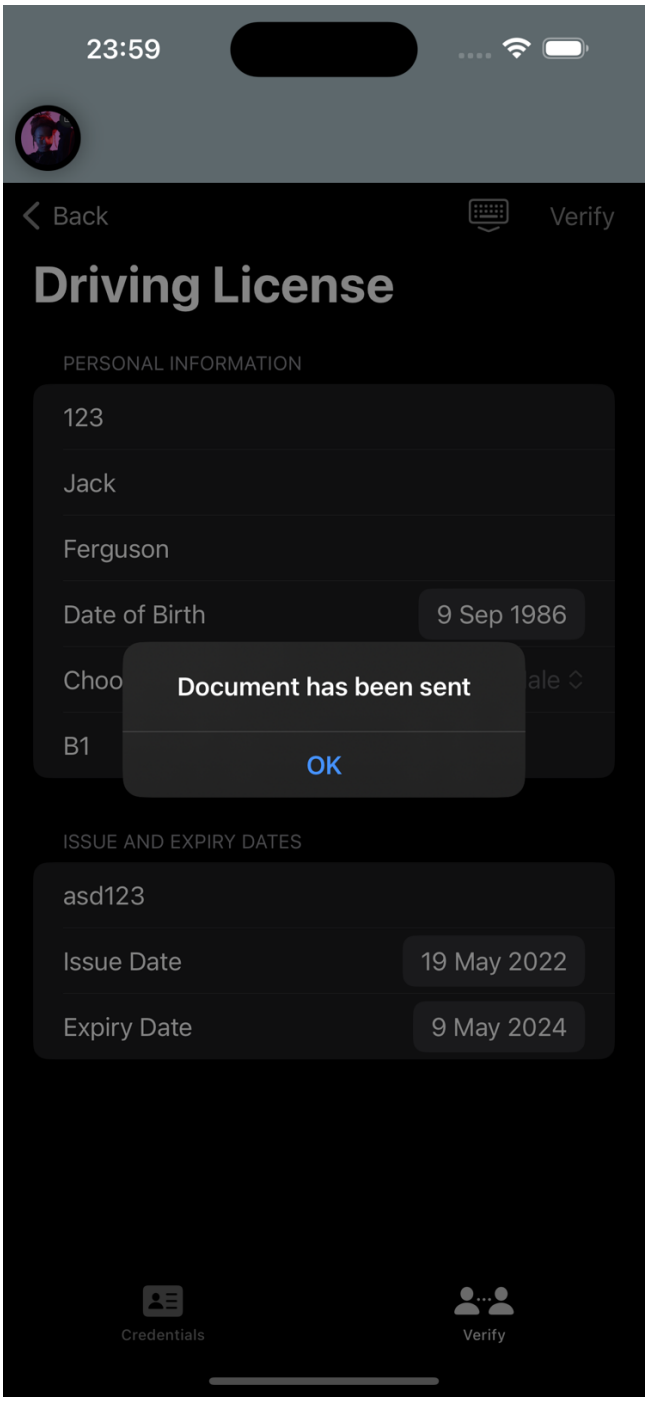
Додаток Б. Екран надсилання документів на верифікацію



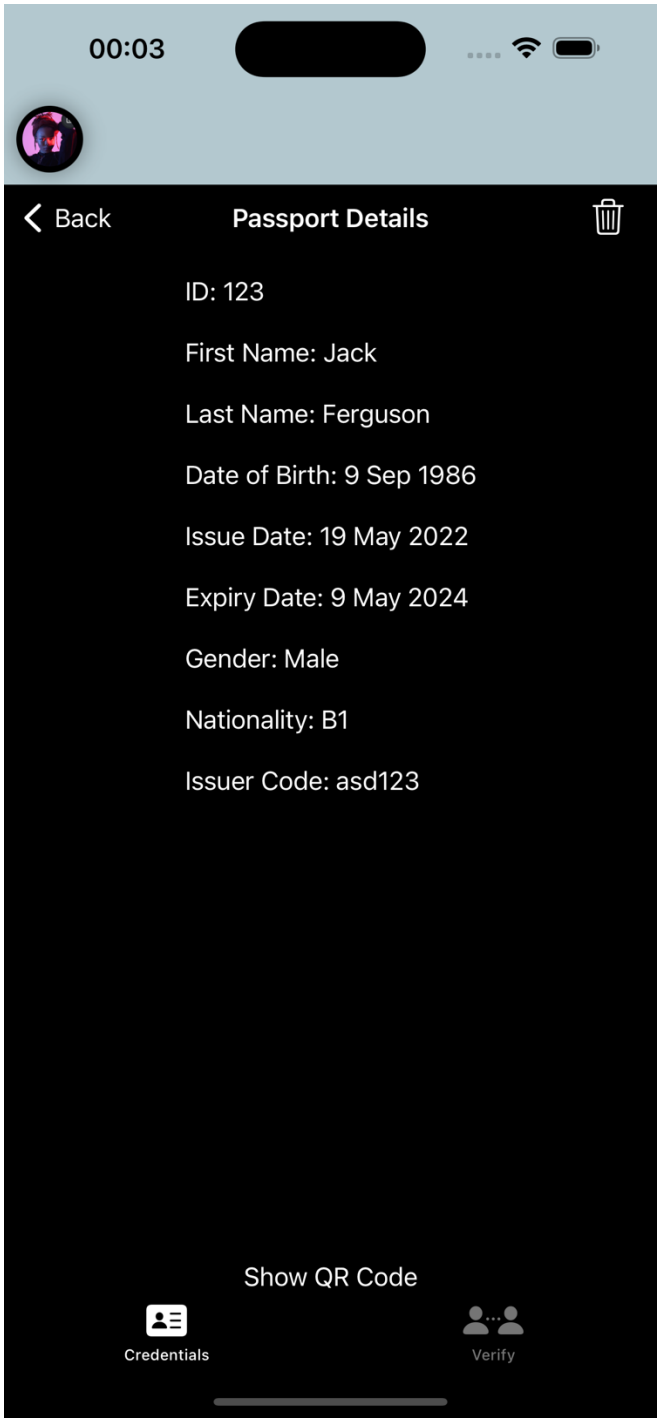
Додаток В. Екрани форм для верифікації документів для посвідчення водія/паспорту



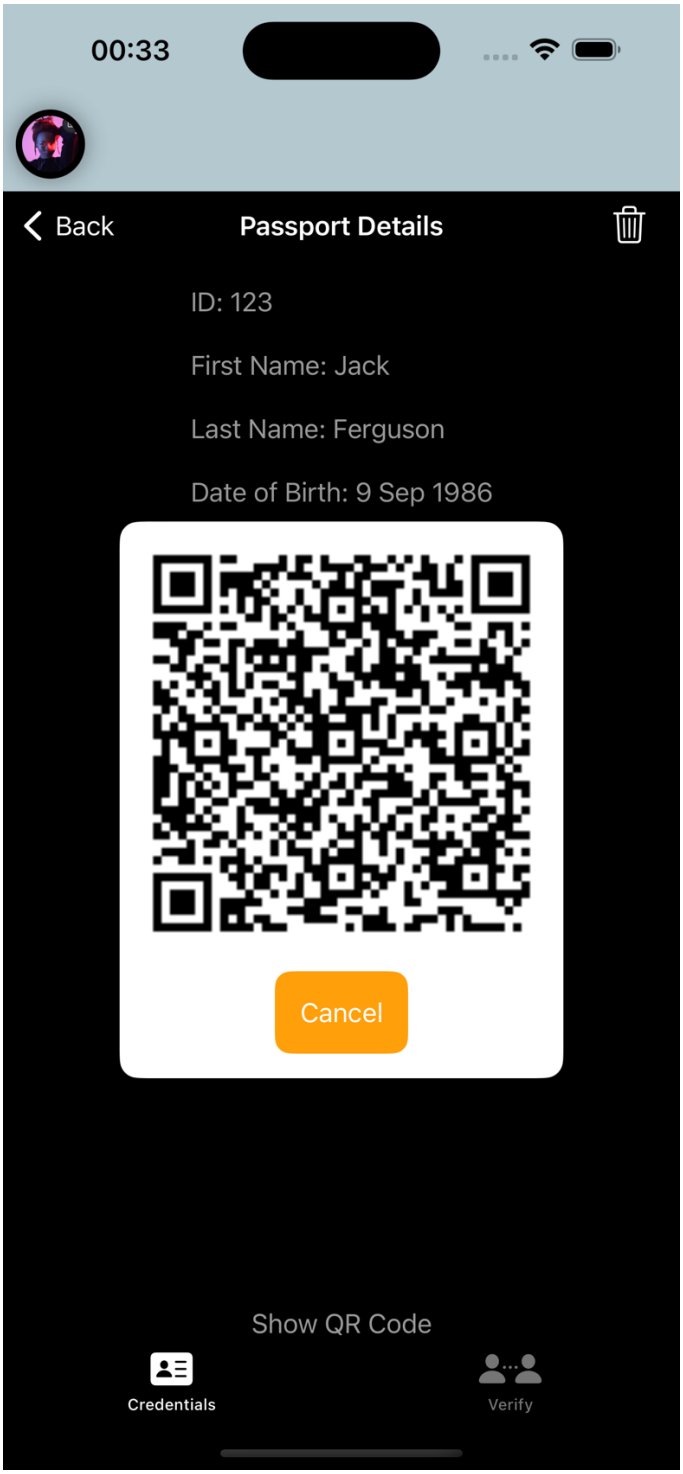
Додаток Г. Екран підтвердження надсилання документу на верифікацію



Додаток Ґ. Екран з деталями підтвердженого документу

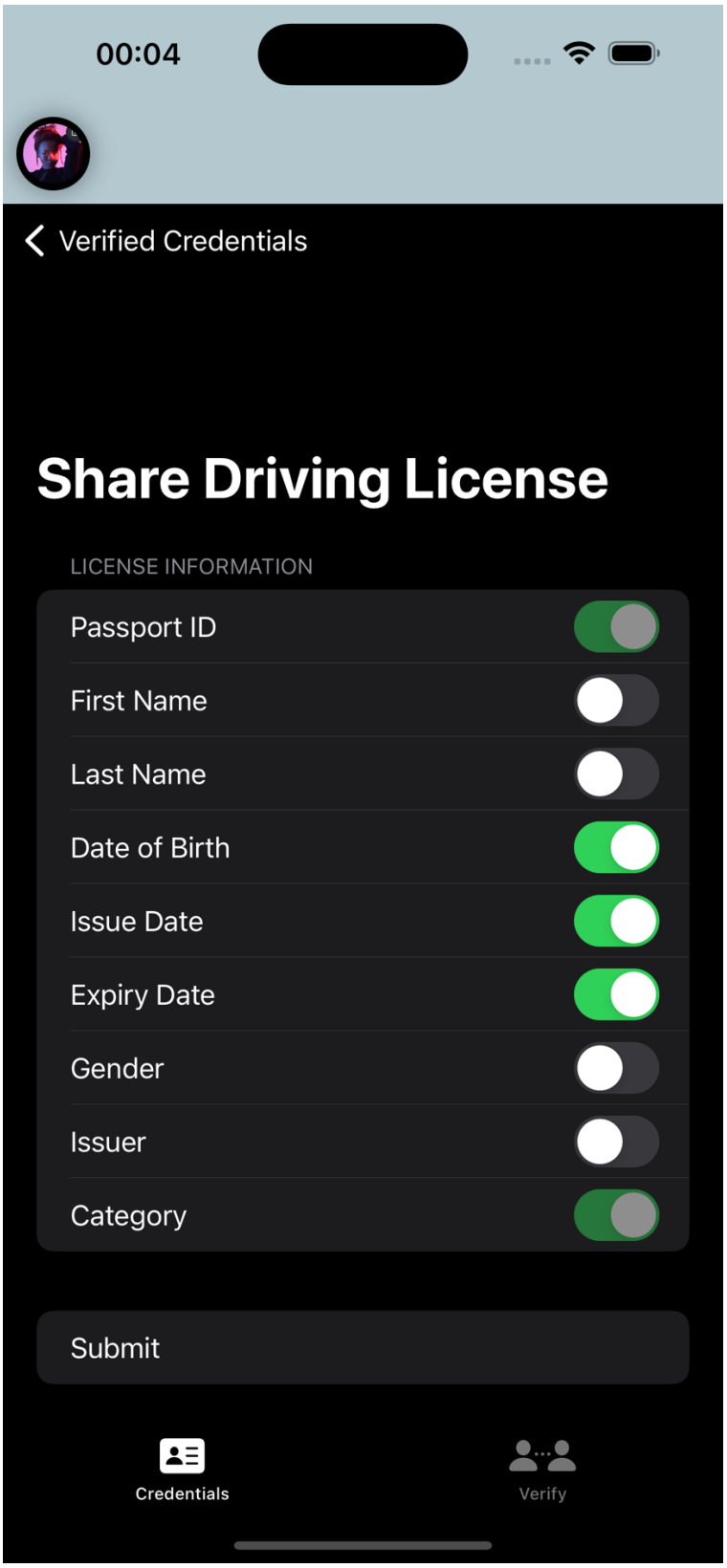


Додаток Д. Екран презентації верифікованого документу у вигляді QR



Додаток Е. Екран звернення серверу до клієнта з запитом на представлення документу

Додаток Є. Вибір полів для поширення з сервером при подальшому відправленні інформації на сервер



Додаток Ж. Кінцеві точки для звертання користувача на сервер

```
"use strict";
const router : Router  = require('express').Router();

let requests : any[]  = []

router.get( path: '/pollRequests',  handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void  => {
    1+ usages
    const sendData = () : void  => {
        if (requests.length > 0) {
            res.json(requests);
            requests = [];
        } else {
            setTimeout(sendData,  timeout: 3000);
        }
    };

    sendData();
});

router.post( path: '/verify',  handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void  => {
        res.status( code: 200).send(req.body);
});

router.post( path: '/submitRequest',  handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void  => {
    const { clientId, credentialType } = req.body;
    requests.push({ clientId, credentialType });
    res.send( body: 'Request submitted successfully.:'+  clientId  + credentialType);
});

router.post( path: '/shareCredential',  handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void  => {
   console.log(req.body)
});

module.exports = router;
```

Додаток З. Власний клас помилок

```
//
//  ApiErrors.swift
//  Digital-Wallet
//
//  Created by Daniil on 07.05.2024.
//

import Foundation

enum ApiErrors: Error, LocalizedError {
    case clientError(String)
    case serverError(String)
    case encodingError(String)
    case unexpectedError

    var errorDescription: String? {
        switch self {
        case .clientError(let message), .serverError(let message),
            .encodingError(let message):
            return message
        case .unexpectedError:
            return "An unexpected error occurred"
        }
    }
}
```

Додаток И. Спеціальна ViewModel, яка містить поля для використання всіма екранами додатку

```swift
//
//  CredentialsViewModel.swift
//  Digital-Wallet
//
//  Created by Daniil on 20.04.2024.
//

import Foundation
import Combine

final class CredentialsViewModel: ObservableObject {
    @Published var verifiedCredentials: [Credential] = []
    @Published var credentialsDocDict: [Credential : Document] = [:]
}
```

Додаток І. Моделі додатку (документ та запит користувача)

```swift
//
//  Document.swift
//  Digital-Wallet
//
//  Created by Daniil on 06.04.2024.
//

import Foundation

protocol Document{
    var description: String { get }
}


enum Gender: String, CaseIterable, Identifiable, Codable {
    var id: String {self.rawValue}
    case male = "Male"
    case female = "Female"
}
```

```swift
//
//  ClientRequest.swift
//  Digital-Wallet
//
//  Created by Daniil on 08.05.2024.
//

import Foundation

struct ClientRequest: Codable {
    let clientId: String
    let credentialType: CredentialType
}

enum CredentialType: String, Codable {
    case drivingLicense="Driving License"
    case passport="Passport"
}
```

Додаток Ї. Модель додатку посвідчення водія

```swift
//
//  DrivingLicense.swift
//  Digital-Wallet
//
//  Created by Daniil on 06.04.2024.
//

import Foundation

struct DrivingLicense: Codable, Document {
    let id: String
    let firstName: String
    let lastName: String
    let dateOfBirth: Date
    let issueDate: Date
    let expiryDate: Date
    let gender: Gender
    let issuer: String
    let category: String

    var dateOfBirthString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: dateOfBirth)
    }

    var issueDateString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: issueDate)
    }

    var expiryDateString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: expiryDate)
    }

    init(
        id: String,
        firstName: String,
        lastName: String,
        dateOfBirth: Date,
        issueDate: Date,
        expiryDate: Date,
        gender: Gender,
        issuer: String,
        category: String
    ) {
        self.id = id
        self.firstName = firstName
        self.lastName = lastName
        self.dateOfBirth = dateOfBirth
        self.issueDate = issueDate
        self.expiryDate = expiryDate
        self.gender = gender
        self.issuer = issuer
        self.category = category
    }
```

```swift
    var description: String {
                """
                ID: \(self.id)
                First Name: \(self.firstName)
                Last Name: \(self.lastName)
                DOB: \(self.dateOfBirthString)
                Issue Date: \(self.issueDateString)
                Expiry Date: \(self.expiryDateString)
                Category: \(self.category)
                Gender: \(self.gender.rawValue)
                Issuer: \(self.issuer)
                """
    }
}
```

Додаток Й. Модель додатку паспорта

```swift
//
//  Passport.swift
//  Digital-Wallet
//
//  Created by Daniil on 06.04.2024.
//

import Foundation

struct Passport: Codable, Document {
    let id: String
    let firstName: String
    let lastName: String
    let dateOfBirth: Date
    let issueDate: Date
    let expiryDate: Date
    let gender: Gender
    let nationality: String
    let issuer: Int

    var dateOfBirthString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: dateOfBirth)
    }

    var issueDateString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: issueDate)
    }

    var expiryDateString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: expiryDate)
    }

    init(
        id: String,
        firstName: String,
        lastName: String,
        dateOfBirth: Date,
        issueDate: Date,
        expiryDate: Date,
        gender: Gender,
        nationality: String,
        issuer: Int
    ) {
        self.id = id
        self.firstName = firstName
        self.lastName = lastName
        self.dateOfBirth = dateOfBirth
        self.issueDate = issueDate
        self.expiryDate = expiryDate
        self.gender = gender
        self.nationality = nationality
        self.issuer = issuer
    }

    var description: String {
                    """
                    ID: \(self.id)
                    First Name: \(self.firstName)
                    Last Name: \(self.lastName)
                    DOB: \(self.dateOfBirthString)
                    Issue Date: \(self.issueDateString)
                    Expiry Date: \(self.expiryDateString)
                    Gender: \(self.gender.rawValue)
                    Nationality: \(self.nationality)
                    Issuer: \(self.issuer)
                    """
    }
}
```

Додаток К. Модель підтвердженого документа

```swift
//
//  Credential.swift
//  Digital-Wallet
//
//  Created by Daniil on 16.04.2024.
//

import Foundation

struct Credential: Codable, Hashable, Identifiable {
    let id: UUID
    let credentialId: String
    let name: CredentialType
    let personFullname: String
    let issuer: String
    let issueDate: Date
    let expirationDate: Date

    var issueDateString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: issueDate)
    }

    var expirationDateString: String {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        return formatter.string(from: expirationDate)
    }
    init(credentialId: String, name: CredentialType, personFullname: String, issuer: String, issueDate: Date, expirationDate: Date) {
        self.id = UUID()
        self.credentialId = credentialId
        self.name = name
        self.personFullname = personFullname
        self.issuer = issuer
        self.issueDate = issueDate
        self.expirationDate = expirationDate
    }

    init(drivingLicense: DrivingLicense) {
        self.id = UUID()
        self.credentialId = drivingLicense.id
        self.name = .drivingLicense
        self.personFullname = "\(drivingLicense.firstName) \(drivingLicense.lastName)"
        self.issuer = drivingLicense.issuer
        self.issueDate = drivingLicense.issueDate
        self.expirationDate = drivingLicense.expiryDate
    }

    init(passport: Passport) {
        self.id = UUID()
        self.credentialId = passport.id
        self.name = .passport
        self.personFullname = "\(passport.firstName) \(passport.lastName)"
        self.issuer = passport.issuer.description
        self.issueDate = passport.issueDate
        self.expirationDate = passport.expiryDate
    }

    static var mock: Credential {
        Credential(
            credentialId: "dasd12312ads",
            name: .passport,
            personFullname: "Robert Hopes",
            issuer: "CNAP",
            issueDate: Date(),
            expirationDate: Date()
        )
    }
}
```

Додаток Л. Менеджер, який займається перевіркою чи на сервері немає запиту на поширення документа

```swift
//
//  PollingManager.swift
//  Digital-Wallet
//
//  Created by Daniil on 01.05.2024.
//

import Combine
import CombineLongPolling
import Foundation


class PollingManager: ObservableObject  {
    var cancellable: AnyCancellable?
    @Published var requests: [ClientRequest] = []
    @Published var showRequestAlert = false
    let url = "http://localhost:3000/pollRequests"
    func startPolling() {
        let config = URLSessionConfiguration.default

        config.timeoutIntervalForRequest = 30000
        config.timeoutIntervalForResource = 30000
        guard let url = URL(string: url) else {
            print("Invalid URL")
            return
        }

        func startNewRequest() {
            cancellable = URLSession(configuration: config)
                .longPollingPublisher(for: url)
                .tryMap { (data, response) -> [ClientRequest] in
                    guard let httpResponse = response as? HTTPURLResponse,
                          (200...299).contains(httpResponse.statusCode) else {
                        throw URLError(.badServerResponse)
                    }
                    return try JSONDecoder().decode([ClientRequest].self, from: data)
                }
                .receive(on: DispatchQueue.main)
                .sink(receiveCompletion: {[weak self] completion in
                    switch completion {
                    case .finished:
                        self?.requests = []
                        self?.startPolling()
                    case .failure(let error):
                        if (error as NSError).code == NSURLErrorTimedOut {
                            print("Timeout warning, trying to restart the polling")
                            self?.startPolling()
                        }
                    }
                }, receiveValue: { [weak self] newRequests in
                    self?.requests = newRequests
                    self?.showRequestAlert = true
                })
        }
        startNewRequest()
    }

    func stopPolling() {
        self.cancellable?.cancel()
    }
}
```

Додаток М. Структура для надсилання документів на сервер для верифікації

```swift
//
//  ApiCaller.swift
//  Digital-Wallet
//
//  Created by Daniil on 18.04.2024.
//

import Foundation

struct ApiCaller {
    private static let basicUrl = "http://localhost:3000"
    private static let httpVerifyMethod = "POST"
    private static let contentType = "application/json"


    static func sendForVerification<T: Encodable & Document>(document: T) async
        throws -> String {
        guard let url = URL(string: basicUrl + "/verify") else {
            throw ApiErrors.serverError("Invalid URL")
        }

        var request = URLRequest(url: url)
        request.httpMethod = httpVerifyMethod
        request.addValue(contentType, forHTTPHeaderField: "Content-Type")

        let encoder = JSONEncoder()
        do {
            let jsonData = try encoder.encode(document)
            request.httpBody = jsonData
        } catch {
            print("Failed to encode the document: \(error)")
            throw ApiErrors.encodingError("Failed to encode the document")
        }

        do{
            let (data, response) = try await URLSession.shared.data(for: request)

            guard let httpResponse = response as? HTTPURLResponse,
                  (200...299).contains(httpResponse.statusCode) else {
                throw ApiErrors.serverError("Server error: \(response)")
            }

            guard let mimeType = httpResponse.mimeType, mimeType == contentType,
                  let dataString = String(data: data, encoding: .utf8) else {
                throw ApiErrors.unexpectedError
            }
            return dataString

        } catch {
            throw ApiErrors.clientError("Could not connect to the server")
        }
    }
}
```

Додаток Н. Допоміжний екран з QR кодом певного документу

```swift
//
//  QrCodeView.swift
//  Digital-Wallet
//
//  Created by Daniil on 22.04.2024.
//

import SwiftUI
import QRCode

struct QrCodeView: View {
    var document: any Document
    @Binding var isPresented: Bool

    var body: some View {
        ZStack {
            Rectangle()
                .fill(Color.black.opacity(0.4))
                .edgesIgnoringSafeArea(.all)

            VStack(spacing: 20) {
                let qrImage = try? QRCode(string: document.description)?.image()
                    Image(uiImage: qrImage ?? UIImage())
                        .resizable()
                        .scaledToFit()
                        .frame(width: 250, height: 250)
                        .contextMenu {
                            Button(action: {
                                UIPasteboard.general.image = qrImage
                            }) {
                                Text("Copy QR Code")
                                Image(systemName: "doc.on.doc")
                            }
                        }

                Button("Cancel") {
                    isPresented = false
                }
                .foregroundColor(.white)
                .padding()
                .background(Color.orange)
                .cornerRadius(10)
            }
            .padding()
            .background(Color.white)
            .cornerRadius(12)
            .shadow(radius: 10)
        }
    }
}
```

Додаток О. Екран поширення документу (на прикладі посвідчення водія)

```swift
//
//  ShareDLView.swift
//  Digital-Wallet
//
//  Created by Daniil on 09.05.2024.
//

import SwiftUI

struct ShareDLView: View {
    @Environment(\.presentationMode) var presentationMode
    @State private var sendFirstName = false
    @State private var sendLastName = false
    @State private var sendDob = false
    @State private var sendIssueDate = false
    @State private var sendExprityDate = false
    @State private var sendGender = false
    @State private var sendIssuer = false

    let dl: DrivingLicense

    var body: some View {
        NavigationView {
            Form {
                Section(header: Text("License Information")) {
                    Toggle("Driving License ID", isOn: Binding.constant(true))
                        .disabled(true)
                    Toggle("First Name", isOn: $sendFirstName)
                    Toggle("Last Name", isOn: $sendLastName)
                    Toggle("Date of Birth", isOn: $sendDob)
                    Toggle("Issue Date", isOn: $sendIssueDate)
                    Toggle("Expiry Date", isOn: $sendExprityDate)
                    Toggle("Gender", isOn: $sendGender)
                    Toggle("Issuer", isOn: $sendIssuer)
                    Toggle("Category", isOn: Binding.constant(true))
                        .disabled(true)
                }
                Button("Submit") {
                    Task {
                        do {
                            try await submitToServer(
                                document: DrivingLicense(
                                    id: dl.id,
                                    firstName: sendFirstName ? dl.firstName : "",
                                    lastName: sendLastName ? dl.lastName : "",
                                    dateOfBirth: sendDob ? dl.dateOfBirth : Date(),
                                    issueDate: sendIssueDate ? dl.issueDate : Date(),
                                    expiryDate: sendExprityDate ? dl.expiryDate : Date()
                                    gender: sendGender ? dl.gender : .male,
                                    issuer: sendIssuer ? dl.issuer : "",
                                    category: dl.category
                                )
                            )
                        } catch (let error){
                            print("There is an error: \(error)")
                        }
                    }
                    presentationMode.wrappedValue.dismiss()
                }
            }
            .navigationBarTitle("Share Driving License")
        }
    }
}
```

Додаток П. Екран форми для відправлення документу на перевірку дійсності

```swift
//
//  DLFormView.swift
//  Digital-Wallet
//
//  Created by Daniil on 18.04.2024.
//

import SwiftUI

struct DLFormView: View {
    @State private var id = ""
    @State private var firstName = ""
    @State private var lastName = ""
    @State private var dateOfBirth = Date()
    @State private var issueDate = Date()
    @State private var expiryDate = Date()
    @State private var gender: Gender = .male
    @State private var issuer = ""
    @State private var category = ""

    @State private var showingAlert = false
    @State private var showingError = false
    @State private var errorMessage = ""

    @EnvironmentObject private var credentialsVm: CredentialsViewModel
    @Environment(\.presentationMode) var presentationMode

    private let keyboardIcon = "keyboard.chevron.compact.down"

    var body: some View {
        NavigationView {
            Form {
                Section(header: Text("Personal information")) {
                    TextField("ID", text: $id)
                    TextField("First Name", text: $firstName)
                    TextField("Last Name", text: $lastName)
                    DatePicker("Date of Birth", selection: $dateOfBirth, displayedComponents: .date)
                    Picker("Choose a gender", selection: $gender) {
                        ForEach(Gender.allCases, id: \.self) { genderCase in
                            Text(genderCase.rawValue)
                        }
                    }
                    TextField("Category", text: $category)
                }

                Section(header: Text("Issue and expiry dates")) {
                    TextField("Issuer", text: $issuer)
                    DatePicker("Issue Date", selection: $issueDate, displayedComponents: .date)
                    DatePicker("Expiry Date", selection: $expiryDate, displayedComponents: .date)
                }
            }
        }
        .navigationTitle("Driving License")
        .toolbar {
            ToolbarItemGroup(placement: .topBarTrailing) {
                Button {
                    hideKeyboard()
                } label: {
                    Image(systemName: keyboardIcon)
                }

                Button("Verify") {
                    Task{
                        do{
                          try  await verifyDocument()
                        } catch let error as ApiErrors{
                            errorMessage = error.localizedDescription
                            showingError = true
                        } catch {
                            errorMessage = "An unexpected error occurred"
                            print(errorMessage)
                            showingError = true
                        }
                            }
                        }
                }
                .alert("Error: \(errorMessage)",isPresented: $showingError) {
                    Button("Cancel", role: .cancel) {}
                }
                .alert("Document has been sent", isPresented: $showingAlert) {
                    Button("OK", role: .cancel) {
                        presentationMode.wrappedValue.dismiss()
                    }
                }
            }
        }
    }
    private func verifyDocument() async throws{
        let dl = DrivingLicense(id: id,
                                firstName: firstName,
                                lastName: lastName,
                                dateOfBirth: dateOfBirth,
                                issueDate: issueDate,
                                expiryDate: expiryDate,
                                gender: gender,
                                issuer: issuer,
                                category: category
        )
            let receivedData = try await ApiCaller.sendForVerification(document: dl)
            print("Received data: \(receivedData)")
            let credential = Credential(drivingLicense: dl)
            credentialsVm.verifiedCredentials.append(credential)
            credentialsVm.credentialsDocDict[credential] = dl
            showingAlert = true
    }
}
```

Додаток Р. Екран клітинки документу (в списку)

```swift
//
//  CredentialCell.swift
//  Digital-Wallet
//
//  Created by Daniil on 16.04.2024.
//

import SwiftUI

struct CredentialsCell: View {
    let credential: Credential
    var body: some View {
        VStack {
            VStack (alignment: .leading) {
                Text(credential.name.rawValue)
                    .font(.title2)
                    .bold()

                Text("Provided by \(credential.issuer)")
                    .font(.subheadline)
            }
            .frame(maxWidth: .infinity, alignment: .leading)
            .padding(.top, 10)

            Spacer()

            HStack (spacing: 1) {
                Text(credential.name.rawValue)
                    .bold()
                Text(" – ")
                Text(credential.personFullname)
                    .font(.headline)
            }

            Spacer()

            HStack {
                VStack(alignment: .leading) {
                    Text("Issued on")
                        .font(.headline)
                        .bold()
                    Text(credential.issueDateString)
                }
                .padding(.leading, 1)
                VStack(alignment: .leading) {
                    Text("Expiration date")
                        .font(.headline)
                        .bold()
                    Text(credential.expirationDateString)
                }
                .padding()
                Spacer()
            }
        }
        .foregroundColor(.black)
        .padding(.horizontal, 15)
        .frame(maxWidth: 350, maxHeight: 230)
        .background(RoundedRectangle(cornerRadius: 20).fill(Color("CredentialCellColor")))
    }
}
```

Додаток С. Головний екран з усіма підтвердженими документами

```swift
//
//  CredentialsView.swift
//  Digital-Wallet
//
//  Created by Daniil on 06.04.2024.
//

import SwiftUI

struct CredentialsView: View {
    @ObservedObject private var pollingManager = PollingManager()
    @EnvironmentObject private var credentialsVm: CredentialsViewModel
    @State private var isActiveShareLink = false

    var body: some View {
        VStack(spacing: 0){
            // MARK: - Header
            HeaderView()

            NavigationStack {
                List(credentialsVm.verifiedCredentials) { credential in
                    NavigationLink(value: credential) {
                        CredentialsCell(credential: credential)
                    }
                }
                .navigationTitle("Verified Credentials")
                .navigationDestination(for: Credential.self) {
                    destinationDetailView(for: credentialsVm.credentialsDocDict[$0]!, credential: $0)
                }
                NavigationLink("", destination: shareView(), isActive: $isActiveShareLink) ⚠ 'init(_:destination:isActive
            }
            .listStyle(PlainListStyle())
        }
        .alert("Share Credential?", isPresented: $pollingManager.showRequestAlert) {
            Button("Share") {
                isActiveShareLink = true
            }
            Button("Cancel", role: .cancel) {}
        } message: {
            if !pollingManager.requests.isEmpty {
                let request = pollingManager.requests[0]
                Text("Would you like to share your \(request.credentialType.rawValue) with \(request.clientId)?")
            }
        }
        .onAppear {
            pollingManager.startPolling()
        }
        .onDisappear {
            pollingManager.stopPolling()
        }
    }
    private func getCredentialByType(credentialType: CredentialType) -> Document? {
         credentialsVm.credentialsDocDict.first(where: { (key, _) -> Bool in
            key.name.rawValue == credentialType.rawValue
        })?.value
    }
    @ViewBuilder
    private func shareView() -> some View {
        if let requestCredentialType = pollingManager.requests.first?.credentialType, let neededCredential =
            getCredentialByType(credentialType: requestCredentialType) {
            switch neededCredential {
            case let passport as Passport:
                SharePassportView(passport: passport)
            case let drivingLicense as DrivingLicense:
                ShareDLView(dl: drivingLicense)
            default:
                Text("Unknown document type")
            }
        }
    }
}
```