Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод і алгоритм покращення аутентифікації людини

(тема)

Виконав:

студент II курсу, групи СПм-21-1

Бондар О.Р.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Бовчалюк С.Я.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Бондарю Олексію Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод і алгоритм покращення аутентифікації людини

затверджена наказом по університету від " 07 " листопада 2022 р. № 1454 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2022 р.

3. Вхідні дані до роботи 1) технологія процедури аутентифікація користувача
2) технології збору і використання біометричних даних
3) технологія .NET
4) хмарні технології
5) технології розробки мобільних застосунків

4. Перелік питань, що потрібно опрацювати у роботі
1) Аналіз актуальності потреби покращення аутентифікації людини
2) Аналіз парольної аутентифікації
3) Аналіз біометричних даних як методу аутентифікації
4) Аналіз мультифакторної аутентифікації
5) Огляд сучасних середовищ розробки та технологій для створення мобільних додатків
6) Розробка архітектури мобільного застосунку
7) Реалізація програмного продукту
8) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз актуальності потреби покращення	8.11.22-13.11.22	
	аутентифікації людини		
2	Аналіз різних видів аутентифікації	14.11.22-18.11.22	
3	Аналіз мультифакторної аутентифікації	19.11.22-20.11.22	
4	Огляд сучасних середовищ розробки та	21.11.22-25.11.22	
	технологій для створення мобільних додатків		
5	Розробка програмного застосунку	26.11.22-4.12.22	
6	Подання кваліфікаційної роботи керівникові	5.12.22-9.12.22	
	та її попередній захист		
7	Подання кваліфікаційної роботи на	10.12.22-11.12.22	
	рецензування		

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Бовчалюк С. Я.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 95 с., 15 рис., 1 табл., 3 дод., 21 джерело.

АУТЕНТИФІКАЦІЯ, ПАРОЛЬ, БІОМЕТРИЧНІ ДАНІ, ХМАРНІ ТЕХНОЛОГІЇ, ДВОХФАКТОРНА АУТЕНТИФІКАЦІЯ, ЗАХИСТ ДАНИХ, МОБІЛЬНИЙ ЗАСТОСУНОК

Метою кваліфікаційної роботи є дослідження сучасних методів і алгоритмів аутентифікації людини, визначення їхніх переваг і недоліків, та покращення функціонування, шляхом уведення додаткових елементів захисту і розробки мобільного застосунку, що реалізує додатковий функціонал.

У ході виконання кваліфікаційної роботи було проведено детальний аналіз поточного стану питання аутентифікації, розглянуто розвиток заходів безпеки, що впроваджувались для захисту даних починаючи від примітивних, закінчуючи найсучаснішими методами, проаналізовано недоліки та сильні місця таких методів. Розглянуто сучасні середовища розробки, технології, фреймворки та мови програмування, що є актуальними при розробці програмного забезпечення у сфері захисту особистих даних, та мобільних застосунків. Розроблено архітектуру та надано інструкцію користування застосунком, як зі сторони користувача, так і зі сторони сервісів, що будуть використовувати цей додаток задля покращення безпеки процесу аутентифікації.

ABSTRACT

Master’s thesis: 95 pages, 15 figures, 1 tables, 3 appendices, 21 sources.

AUTHENTICATION, PASSWORD, BIOMETRIC DATA, CLOUD TECHNOLOGIES, TWO FACTOR AUTHENTICATION, DATA PROTECTION, MOBILE APPLICATION

The purpose of the qualification work is to research modern methods and algorithms of human authentication, to determine their advantages and disadvantages, and to improve their functioning by introducing additional protection elements and developing a mobile application that implements additional functionality.

In the course of the qualification work, a detailed analysis of the current state of the authentication issue was carried out, the development of security measures implemented to protect data from primitive to the most modern methods was considered, the shortcomings and strengths of such methods were analyzed. Modern development environments, technologies, frameworks and programming languages that are relevant in the development of software in the field of personal data protection and mobile applications are considered. The architecture has been developed and instructions for using the application have been provided, both from the user's side and from the side of the services that will use this application to improve the security of the authentication process.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – програмний інтерфейс застосунку (англ., Application Programming Interface)

AWS – веб сервіси Амазон (англ., Amazon Web Services)

CSS3 – каскадні таблиці стилів 3 (англ., Cascading Style Sheets 3)

CTSS – сумісна система розподілу часу (англ., Compatible Time-Sharing System)

DDD – предметно-орієнтоване проектування (англ., Domain Driven Design)

HTML – мова гіпертекстової розмітки (англ., HyperText Markup Language)

IDE – інтегроване середовище розробки (англ., Integrated development environment)

MAUI – інтерфейс користувача для багатьох платформ (англ., Multi-Platform App UI)

MD5 – алгоритм дайджест повідомлення 5 версії (англ., Message Digest 5)

MFA – багатофакторна аутентифікація (англ., Multi-factor authentication)

QR – швидка відповідь (англ., Quick Response)

SHA – алгоритм безпечного шифрування (англ., Secure Hash Algorithm)

WSGI – інтерфейс взаємодії веб серверу з застосунками (англ., Web Server Gateway Interface)

XML – розширювана мова розмітки (англ., eXtensible Markup Language)

ВСТУП

Питання доступу до певних речей, чи інформації, почало набирати ставати все більш актуальним ще з давніх давен, адже в кожної людини виникала потреба обмежити доступ чужаків до свого майна. Звичайно, перші люди входили до малих соціальних груп, як родина, плем’я, тощо, де визначення належності людини до соціальної групи не викликало великих труднощів, адже кожен знав усіх в обличчя, та запросто міг ідентифікувати чужака. Проте, з розвитком суспільства питання ідентифікації людини поставало все частіше та серйозніше, бо речі ставали ціннішими, а інформація більш конфіденційною.

Одним з перших ефективних методів, які люди винайшли для обмеження доступу до певних місць стали замок та ключ. Звичайно, на момент винаходження це були дуже примітивні реалізації замків, з відкриванням яких зараз би напевно справилась би будь-яка розумна дитина, проте, вважається, що це дало дуже великий поштовх для розвитку методів обмеження доступу то тієї, чи іншої інформації. Дуже важко уявити сучасний світ без цього геніального винаходу, саме тому замки зараз можна побачити у кожній будівлі, будь-то жилий будинок, чи продуктова крамниця.

З винайденням навіть примітивної системи захисту ситуація покращилась, проте люди з бажанням отримати доступ до чужих даних, або речей теж не сиділи склавши руки, тому способи для відтворення ключів до тих, чи інших примітивних замків не змусили довго на себе чекати.

З наближенням розвитку суспільства до сучасності випадків, які б потребували визначення особистості людини ставало все більше, й вони вимагали більшої безпечності та індивідуальності. Наприклад, використовуючи замок для обмеження доступу, ключ могли мати декілька осіб, чи він міг передаватись від однієї до іншої особи, чи навіть міг бути викраденим, що представляло собою велику загрозу отримання доступу

небажаними особами. Таким чином в різноманітних банках, чи подібним їм установам, доступ до персональних даних надається саме конкретній особі, що стає можливим завдяки посвідченням особи та базі даних, що вміщує в собі інформацію про всіх відомих системі осіб. Хоча такий спосіб і є доволі безпечним, його реалізація вимагає значно більших ресурсів, бо треба мати людей, які б перевіряли відповідність наданих особою документів, підтримували базу осіб актуальною і т.д.

З подальшим розвитком інформаційних технологій та комп'ютеризацією суспільства проблема, описана раніше, перекочувала до цифрового світу. З кожним роком користувачів інформаційних технологій стає все більше, та вже зараз інформація майже кожної людини в тій чи іншій кількості зберігається на цифрових носіях, чи в мережі інтернет. Ця інформація може бути різною, починаючи від невеликих текстових заміток розміром в декілька кілобайт, закінчуючи архівами з фото, чи відео, що займають десятки терабайт й більшість цієї інформації вимагає високого рівню захисту.

Такий ривок у сфері інформаційних технологій приніс із собою нові виклики для забезпечення надійного механізму надання доступу до інформації, альтернативою до замків та ключів стали шифрування та паролі, а для ідентифікації особистості стали використовувати біометричні дані особи. Таким чином процес надання доступу до інформації став більш автоматизований, швидкий та менш затратний на людський ресурс.

Людські біометричні дані наразі стали дуже точним та найбезпечнішим способом ідентифікування особи, адже виявилось, що більшість з них вміщують в собі велику кількість унікальних рис, які не повторюються серед людей, що надає можливість ідентифікувати конкретну людину. Усі ці різноманітні методи ідентифікації людини отримали назву "аутентифікація". Ця робота присвячена вивченню існуючих та поширених методів аутентифікації людини, аналізу їхніх недоліків та переваг, методів, що вживалися для покращення процесу аутентифікації у процесі її еволюції, та

виявленню моментів, які потребують допрацювання, задля покращення ефективності, чи зручності їх використання.

На основі проаналізованої інформації зроблено висновки про існуючі обмеження при використанні сучасних методів аутентифікації. Аргументовано доцільність використання мобільного пристрою для проведення процедури аутентифікації з використанням сучасних методів та розроблено застосунок для демонстрації декількох з них. Такий застосунок поєднав в собі декілька різних методів та підходів для забезпечення максимальної надійності.

Враховуючи вищевказане, головною метою кваліфікаційної роботи є дослідження сучасних методів і алгоритмів аутентифікації людини, визначення їхніх переваг і недоліків, та покращення функціонування, шляхом уведення додаткових елементів захисту і розробки мобільного застосунку, що реалізує додатковий функціонал.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз актуальності потреби покращення аутентифікації людини

Обмеження доступу до конфіденційної інформації, чи особистих даних є невід'ємною складовою сьогодення. Наразі для вирішення цієї проблеми використовується велика кількість методів, таких як паролі, ключі шифрування [1], якими може володіти як одна людина, так і декілька осіб, або ж біометричні дані, що є унікальними для кожної людини. Проте люди, що бажають заволодіти чужими конфіденційними даними теж не сидять на місці, методи обходу захисту розвиваються дуже швидко, шахраї витягують паролі користувачів обманом, системи основані на біометричних даних обходяться, хоча й з більшими зусиллями. Все це призводить до нових ризиків потрапляння особистих даних до небажаних рук, що все частіше вимагає від світу перегляду існуючих методів аутентифікації.

Впевнитись в необхідності перегляду та покращення існуючих методів аутентифікації можна розглянувши статистику, що відноситься до популярних методів шахраїв. Найпоширенішими зазвичай є прості методи такі як фішинг, чи підбір паролів, а також більш складні методи, які використовують слабкі місця та вразливості тієї, чи іншої системи. У випадку з останніми методами відповідальність за безпеку системи повністю лягає на плечі розробників системи, що є меншим ризиком, адже у сучасному світі все більше використовуються стандартизовані підходи та практики, що зазвичай вже давно перевірені на всі нині відомі уразливі місця, проте з фішингом та підбором паролів все не так просто. Уразливість конфіденційних даних при використанні таких методів напряму залежить саме від користувача, його уважності та обережності.

Найпоширенішим методом наразі є фішинг – вид шахрайства, який має

за мету виманити у неуважного, чи довірливого користувача персональні дані, такі як паролі, номери телефонів, та інші данні, що можуть знадобитись для доступу до захищеної інформації особи [2]. Переважна більшість фішингових атак проводяться через електронну пошту, менша частина розташована на різноманітних сайтах у вигляді рекламних повідомлень, мізерна частина розсилається на мобільні телефони, проте принцип завжди залишається одним, заманити користувача на сайт шахраїв, який зазвичай копіює вигляд популярних ресурсів, та змусити неуважного користувача ввести на цьому сайті свої дані. На жаль таких сайтів з кожним роком стає все більше (рисунок 1.1). Тільки за останні 2 роки їх кількість збільшилась більш ніж в 3 рази.

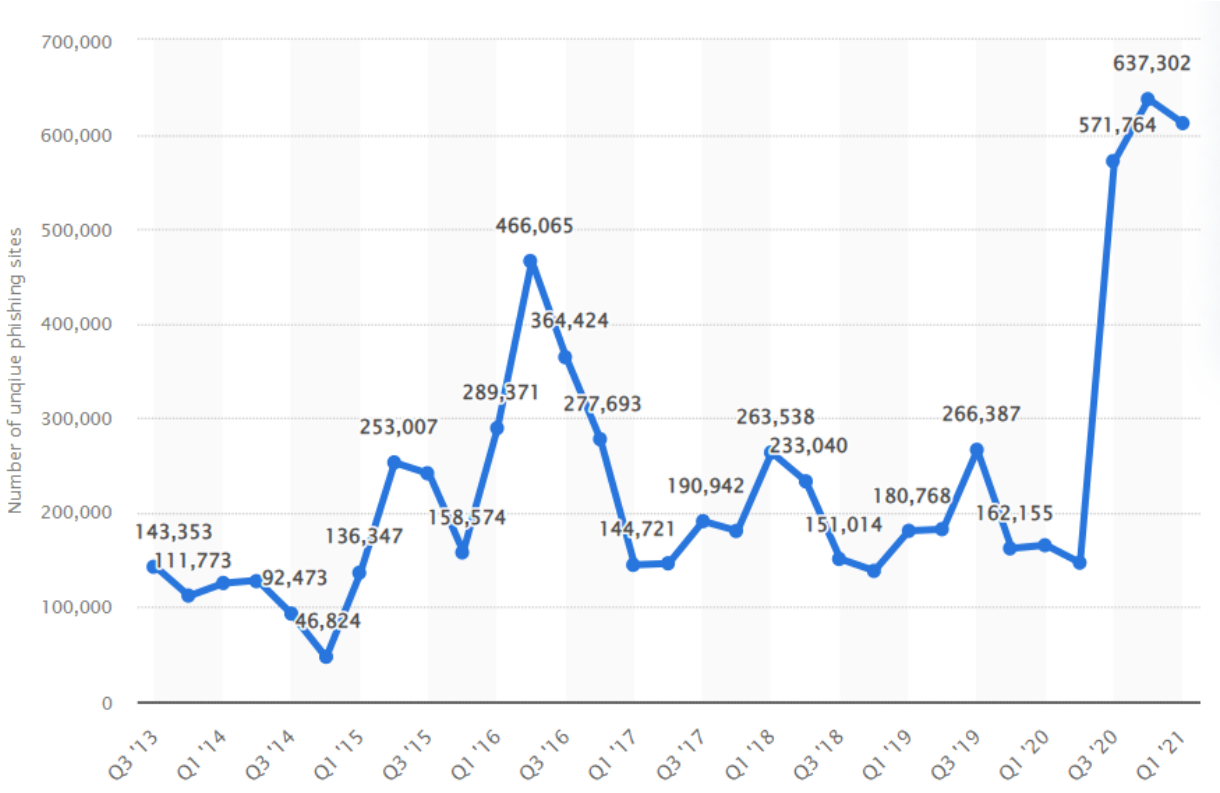

Рисунок 1.1 – Кількість унікальних фішингових сайтів за роками

Наведена вище цифри відображають саме кількість фішингових сайтів, якщо ж взяти до уваги, що кожен з них щодня задіяний у тисячах атак, можна

припустити, що таким чином втрачати особисті дані може дуже велика кількість людей. За статистикою ресурсу CyberTalk щодня відправляється приблизно 15 мільйонів листів з фішинговими сайтами [11], звичайно, велика кількість з них потрапляють під спам фільтри [17] поштових сервісів, проте приблизно 30% з них переглядається користувачами.

Розглянувши навіть верхівку айсбергу статистичних даних, пов’язаних з фішинговими атаками, можна зробити висновки щодо доцільності перегляду існуючих методів аутентифікації та їх безпекового аспекту. Тенденція збільшення кількості подібних атак потребує зміщення акценту саме на забезпечення захисту користувача від фактору неуважності, чи необережності поводження використовуючи аутентифікацію до того, чи іншого ресурсу, що часто може призвести до втрати особистих даних. Таким чином стає очевидною актуальність вивчення нових методів, та привнесення прогресивних ідей до вже існуючих, адже безпека є невід’ємним та найголовнішим аспектом взаємодії людиною з будь-якою сучасною системою.

1.2 Аналіз парольної аутентифікації користувача

Парольна аутентифікація досі залишається найпопулярнішим методом аутентифікації користувача та захисту його даних. Цей метод використовується чмайже у кожній системі, яка тим, чи іншим шляхом пов’язана з особистими даними людини та обмежує доступ до них небажаними особами. Таким чином можна вважати пароль першою лінією захисту проти неавторизованого доступу до інформації користувача.

Використовувати перші паролі у комп’ютерних системах почали майже одночасно зі створенням перших таких систем. Операційна система Compatible Time-Sharing System (CTSS) [14], що була розроблена в 1961 році у Массачусетському технологічному інституті була першою, що розробила та використала парольну аутентифікацію для входу в систему [16]. Після того,

як користувач намагався увійти в систему, вона вимагала ввести пароль. При введенні цей пароль не відображався на екрані задля забезпечення більшої безпеки. Хоча перше використання такого методу аутентифікації у комп’ютерних системах й було примітивним, проте саме цей перший крок заклав основополягаючі принципи, які досі використовуються у сучасності.

З моменту першого використання паролів у комп’ютерних системах пройшло вже багато часу і, якщо ще пів сторіччя тому паролі були чимось невідомим для широких мас та використовувались переважно у науковій, чи військовій спільноті, де зазвичай на той час компютер’ні системи набирали широких обертів, то зараз ситуація істотно змінилась. Сьогодні паролі використовуються будь-де, для доступу в соціальні мережі, електронну пошту, банківські системи та в багатьох інших місцях. Зараз навіть складно уявити систему, яка б не використовувала паролі.

Феномен популярності паролів полягає у відносній простоті користування ними, достатньо лише вигадати комбінацію символів, що буде достатньо складною та такою, що легко запам’ятовується. Надалі від користувача вимагається лише пам’ятати свій пароль, а сам процес аутентифікації з його використанням займає декілька секунд, що значно швидшве та простіше ніж деякі методи, як наприклад аутентифікація на основі сертифікату.

Однак окрім простоти використання такий метод має серйозні недоліки. Оскільки пароль треба постійно тримати в голові, люди намагаються вигадувати паролі, що легко запам’ятовуються. Зазвичай під такий критерій підпадають легкі та короткі паролі. Такі паролі є великим ризиком для користувача та легкою здобиччю для шахраїв. Все частіше для решення цієї проблеми в системах встановлюється обмеження на складність паролів, проте таке рішення веде за собою іншу проблему. Більш складні паролі стають також складнішими для запам’ятовування, тому люди часто прибігають до записування таких паролів у блокнотах, чи в інших подібних місцях, таким чином зберігаючи свій пароль у відкритому доступі, що лише

збільшує ризики отимання доступу до конфіденціальної інформації небажаними особами.

Таким чином, можна зробити висновок, що відповідальність користувача, його відношення до створення надійного, складного паролю та збереження його від сторонніх очей є значним фактором у забезпеченні безпеки особистих даних при використанні паролю як методу аутентифікації. Саме через цей аспект зараз паролі не вважаються ідеальним та максимально безпечним методом аутентифікації, тому їх рекомендуються використовувати лише в комбінації з додатковими методами для забезпечення більшої надійності.

Іншим не менш важливим фактором для безпеки парольного методу аутентифікації є підхід, що використовується для зберігання паролів у системі. Як би сильно не покращувався прогрес у сфері кібербезпеки, час від часу шахраям все ж вдається отримати доступ до баз даних різноманітних систем, які вони використовують для своїх цілей, чи навіть викладають у вільний доступ. До таких баз часто потрапляють дані паролі користувачів та інші різноманітні деталі. Саме через це велика увага повинна приділятися методам, що використовується для зберігання паролів, адже недостатньо безпечні методи можуть привести до величезних збитків як зі сторони користувача, так і зі сторони власників компаній, які допустили такі витоки баз даних.

Навіть великі компанії, що займають провідні місця в індустрії та гарно зарекомендували себе час від часу залишають слабкі місця в системах, що дає змогу шахраям отримати доступ до гігабайтів конфіденційної інформації. Великі корпорації несуть найтяжчі збитки при таких мільйонних витоках даних. Чим більша кількість користувачів використовують той чи інший сервіс, тим більшу ціну доводиться заплатити компанії навіть за маленьку помилку, що призводить до витоку. За останнє десятиліття таким чином було оприлюднено мільйони записів (таблиця 1.1).

Таблиця 1.1 – Оприлюднені під час витоку даних записи акаунтів користувачів різних компаній

Компанія	К-ть оприлюднених записів користувачів
Yahoo!	~3 мільярди
Facebook	~533 мільйони
FriendFinder	~412 мільйонів
MySpace	~360 мільйонів
Marriott	~323 мільйони
LinkedIn	~165 мільйонів
Equifax	~145 мільйонів
HeartLand	~130 мільйонів
Target	~110 мільйонів
Capital One	~106 мільйонів

З перших днів існування комп’ютерних систем паролі дуже часто зберігалися у вигляді звичайного тексту. Оскільки за тих часів взломи систем не відбувалися так часто, цей метод мав право на життя, він є дуже швидким, оскільки не потребує ніяких додаткових дій над паролем, задля його захисту, проте ризик розкриття списку паролів усіх користувачів при витоку бази даних перекривав усі можливі переваги такого методу. Наразі жодна прогресивна система не ризикує використовувати такий метод для зберігання паролів своїх користувачів.

З поширенням комп’ютерних систем серед все більшої кількості користувачів питання зберігання паролів ставало гострішим. Кількість таких систем була великою, та не всі з них були гарно захищені, чим часто користувались шахраї. Продовжувати тримати паролі у відкритому вигляді було небезпечно, тому все частіше починали використовуватись шифрування та хешування.

Шифрування – алгоритмічне перетворення даних, що зазвичай

виконується у посимвольній послідовності задля приховання інформації та отримання шифрованого тексту. Метод зберігання паролю у зашифрованому вигляді вирішує проблему його потрапляння до чужих рук у відкритому вигляді, проте такий метод має свої недоліки. Процес шифрування передбачає можливість розшифрування зашифрованої послідовності. Таким чином цей метод залишає люфт для злочинців та дає змогу отримати пароль у початковому вигляді, якщо отримати доступ до алгоритму шифрування, тому цей метод наразі не є дуже популярним, хоча й надає більшу надійність, ніж зберігання паролів у відкритому вигляді.

Хешування – процес перетворення вхідного набору даних на результуючій набір фіксованої довжини. Таке перетворення відбувається за допомогою хеш-функцій, які в свою чергу представляють собою алгоритм, що покладається на різноманітні критерії з вхідного набору. Хешування, на відміну від шифрування, є незворотнім процесом. Тому, отримавши одного разу хешоване значення, отримати початкове значення отримати вже не вийде. Саме цей аспект надає перевагу хешуванню над шифруванням для використання його як вигляду для зберігання паролів. Для перевірки правильності паролю користувача при використанні хешування, системі не потрібно намагатись отримати з хешу, що зберігається в базі даних, початкове значення паролю. Перевірка проводиться повторним хешуванням пароля та порівняння проводиться саме над хешем введеного пароля та хешем, що вже зберігається в базі даних.

Таким чином, при використанні хешування для зберігання паролів, у випадку витоку бази даних, дізнатись реальний пароль буде можливо лише методом підбору та тільки за умови, якщо відома хеш-функція, що була використана для хешування. Для ускладнення ж такого варіанту подій використовують різноманітні алгоритми хешування, які зазвичай відрізняються своєю складністю та довжиною результуючого хеш-значення.

MD5 – алгоритм хешування, розроблений в Массачусетському технологічному інституті 1991 року професором Рональдом Л. Рівестом.

Алгоритм призначений для створення контрольних сум або відбитків повідомлення довільної довжини і подальшої перевірки їх автентичності.

Особливість алгоритму MD5 полягає в тому, що хеш, отриманий після роботи функції видає рядок 16 байт (128) біт [8]. Сам алгоритм складається з 5 кроків:

- вирівнювання потоку;
- додавання довжини початкового повідомлення;
- ініціалізація буферу;
- розрахунок значень в циклі;
- складання результату всіх обчислень.

Алгоритм MD5 є класичним алгоритмом хешування, проте наразі вважається застарілим для використання у критичних до безпеки сферах. Цей алгоритм є дуже швидким, що надає йому перевагу у сферах, де безпека не відіграє настільки вагому роль, наприклад підтвердження цілісності файлу, проте він є дуже вразливим до деяких типів атак:

- перебір за словником;
- атаки з використанням райдужних таблиць;
- колізії хеш-функцій.

Перебір за словником – атака на систему захисту, яка застосовує метод повного перебору можливих паролів, що часто використовуються для аутентифікації. Така атака здійснюється шляхом послідовного перегляду всіх паролів у відкритому вигляді певного виду зі словника, з метою подальшого злому системи та отримання доступу до секретної інформації. Як видно з визначення, словникові атаки є атаками, що використовують перебір. Єдина відмінність полягає в тому, що ці атаки зазвичай більш ефективні, оскільки стає непотрібним пробувати всі комбінації символів, щоб досягти успіху. Зловмисники використовують великі списки часто використовуваних паролів, таких як імена домашніх тварин, вигаданих персонажів або інші групи слів – звідси й назва атаки. Однак, якщо пароль справді унікальний, атака за словником не спрацює.

Атаки з використанням райдужних таблиць це ще один метод злому хешу. Він заснований на генеруванні великої кількості хешей з набору символів, щоб по базі, що вийшла, вести пошук заданого хешу. Райдужні таблиці складаються з хеш-ланцюжків і є більш ефективними, ніж словниковий тип атаки, оскільки вони оптимізують вимоги до зберігання підготовлених даних. Таким чином замість перебору з проходженням кожного варіанту паролю через хеш-функцію цей тип атак оперує з заздалегідь підготовленим списком вже порахованих хеш-значень.

Колізії хеш-функції – отримання того й самого значення функції для різних повідомлень з ідентичним початковим буфером. Додатково до колізій існують також псевдоколізії, що визначаються як такі, що мають однакові хеш-значення для різних початкових значень буфера, самі ж повідомлення для таких колізій можуть бути однаковими або різними. У 1996 році Ганс Доббертін виявив псевдоколізію в алгоритмі MD5 з певними нестандартними векторами ініціалізації. Виявляється, можна створити друге повідомлення, яке буде мати той самий хеш, що й перше.

MD5 перебував під пильною увагою криптоспільноти з моменту його першого випуску і до 2004 року мав лише незначні недоліки. Проте влітку 2004 року криптографи Ван Сяоюнь і Фен Денгу продемонстрували алгоритм, здатний генерувати колізії MD5 за допомогою стандартного вектора ініціалізації.

Пізніше цей алгоритм був вдосконалений, що значно скоротило час на пошук пари повідомлень, що дозволило знаходити колізії з прийнятною кількістю обчислень. Проте, як виявилося, проблема зіткнень в MD5 не може бути повністю вирішена. Через це MD5 майже не використовується для зберігання паролей у сучасному світі, на зміну йому прийшли інші алгоритми, які зробили спробу врахувати його основні недоліки.

Сімейство алгоритмів SHA (Secure Hash Algorithm) – алгоритми хешування, що були розроблені Агенством національної безпеки США. Перша версія алгоритму безпечного шифрування SHA-0 була розроблена

сумісно з Національним інститутом стандартів та технологій США у 1993 році, проте з часом була відізвана через помилку знайдену самими ж розробниками. Суть помилки так і не була оприлюднена, але потім SHA-0 була замінена на нову версію SHA-1, де знайдена раніше помилка була виправлена. Порівнюючи SHA-1 з MD5 можна виділити значні зміни в кращу сторону у питанні колізій, проте повністю ця проблема нікуди не зникла. Тому велика кількість провідних компаній відмовилась від SHA-1 віддавши перевагу більш надійним алгоритмам. Значних покращень зазнала наступна версія алгоритму SHA-2. Наразі вона використовується дуже великою кількістю компаній та стала стандартом в індустрії. Використання сімейства SHA не обмежується лише зберіганням паролів, список основних векторів його використання [19] налічує велику кількість напрямків:

- хешування паролів;
- створення електроних цифрових підписів сертифікатів;
- блочні алгоритми шифрування;
- знаходження комбінацій даних, що є доказами виконання роботи при емісії криптовалюти.

Законами деяких країн навіть зазначено можливість використання алгоритму в урядових системах, що підтверджує рівень безпеки алгоритму. Проте під час досліджень та експериментів над алгоритмом було виявлено потенційну та теоритичну можливість колізій у алгоритмі SHA-2. Враховуючи цей факт наступна версія SHA-3 планується розроблятись з використанням кардинально іншого алгоритму, замість допрацювання вже існуючих версій.

Окрім алгоритмів хешування, що розробляться та вдосконалюються десятиліттями в сучасному світі набирають обертів і нові алгоритми, що враховують сучасні потреби та виклики. Одним з таких алгоритмів є Argon2. В 2015 році цей алгоритм переміг у конкурсі Password Hashing Competition, в якому велика кількість спеціалістів прийняли участь у створенні новітнього алгоритму, що мав відповідати додатковим вимогам з використання об’єму

пам’яті, кількості проходів по блокам пам’яті, можливості використання паралелізму та стійкості до криптоаналізу до якого як раз входять такі поняття як колізії та інше.

Враховуючи популярність методу зберігання паролів з використанням хешування можна зробити висновки, що цей напрямок є дуже надійним. Спільнота комп’ютерних інженерів гідно справляється з різноманітними викликами, які знаходяться в існуючих методах та алгоритмах, що підтверджується постійним вдосконаленням вже існуючих та розробкою нових алгоритмів.

Використання хешування для зберігання паролів суттєво покращило стан інформаційної безпеки, проте використання такого способу з часом втрачає свою силу. Бази хакерів стають все більшими, що робить звичайне хешування уразливим до таких атак, як райдужні таблиці, де використовується база з заздалегідь порахованими хеш-значеннями.

Іншим недоліком при оприлюдненні бази даних є також акаунти, що використовують однакові паролі. Використання хеш-функції на однакових паролях призводить до однакового результату хеш-значення. Таким чином підібравши пароль для одного акаунту користувача, доступ буде отримано до всіх користувачів, що мають ті ж паролі.

Алгоритм хешування з використанням солі захищає хеші паролів від словникових атак, додаючи додаткову випадковість до їх генерації. Хешування пароля з сіллю – це коли випадкові дані, які називаються сіллю, використовуються як додаткові вхідні дані для хеш-функції, яка хешує пароль. Метою використання солі є захист від словникових атак або атак на хешовані паролі за допомогою райдужної таблиці.

Щоб використати сіль для генерації хеш-значення, для кожного пароля випадковим чином генерується новий параметр [3]. Сіль і пароль поєднуються, а потім обробляються за допомогою криптографічної хеш-функції. Отриманий результат зберігається в базі даних разом із сіллю.

Оскільки людям не потрібно запам’ятовувати та навіть знати значення

солі, що використовується разом з їх паролем, цей спосіб може зробити розмір райдужної таблиці, необхідний для успішної атаки, непомірно великим. Оскільки солі відрізняються для кожного пароля, вони також захищають паролі, які часто використовуються або акаунти користувачів, що використовують той самий пароль на кількох веб-сайтах (рисунок 1.2), відрізняючи всі екземпляри хешу, отриманого з використанням солі, для того самого пароля один від одного.

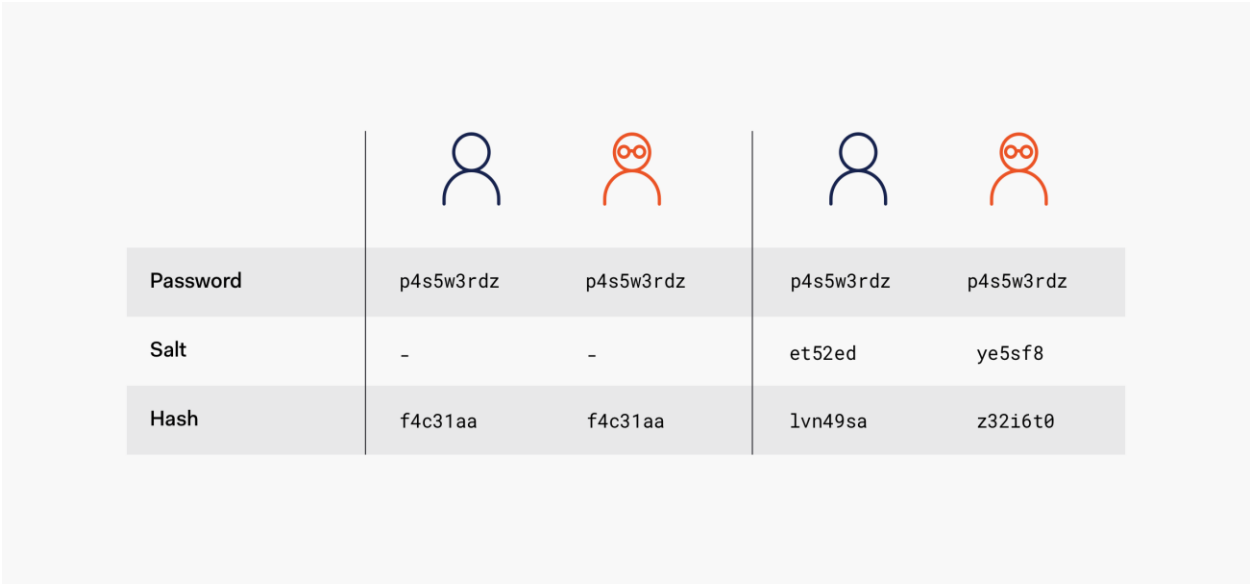

Рисунок 1.2 – Приклад використання солі при хешуванні паролю

Окрім забезпечення безпеки доступу до персональних даних методом постійного вдосконалення комп'ютерної системи, існує певна кількість факторів на які користувач має безпосередній вплив. Одним з таких чинників є вміст паролю. Люди часто не мають бажання запам'ятовувати складні паролі, надаючи переваги більш простим, іноді навіть примітивним, що зазвичай стає найбільшим ризиком для втрати доступу до особистих даних.

Розвиток комп'ютерних систем та інформаційних технологій з кожним роком набирає приголомшливих обертів, разом з цим стрімко зростають обчислювальні можливості комп'ютерів. Зараз звичайний комп'ютер, що можна знайти майже в кожній домівці, за одиницю часу може проводити

кількість операцій в тисячі разів більшу, аніж це навіть можна було уявити декілька десятків років тому, що вже казати про обчислювальні машини, які використовуються в науковій діяльності. Таке стрімке зростання обчислювальних можливостей стало можливим завдяки збільшенню вмісту транзисторів у новітніх процесорах (рисунок 1.3).

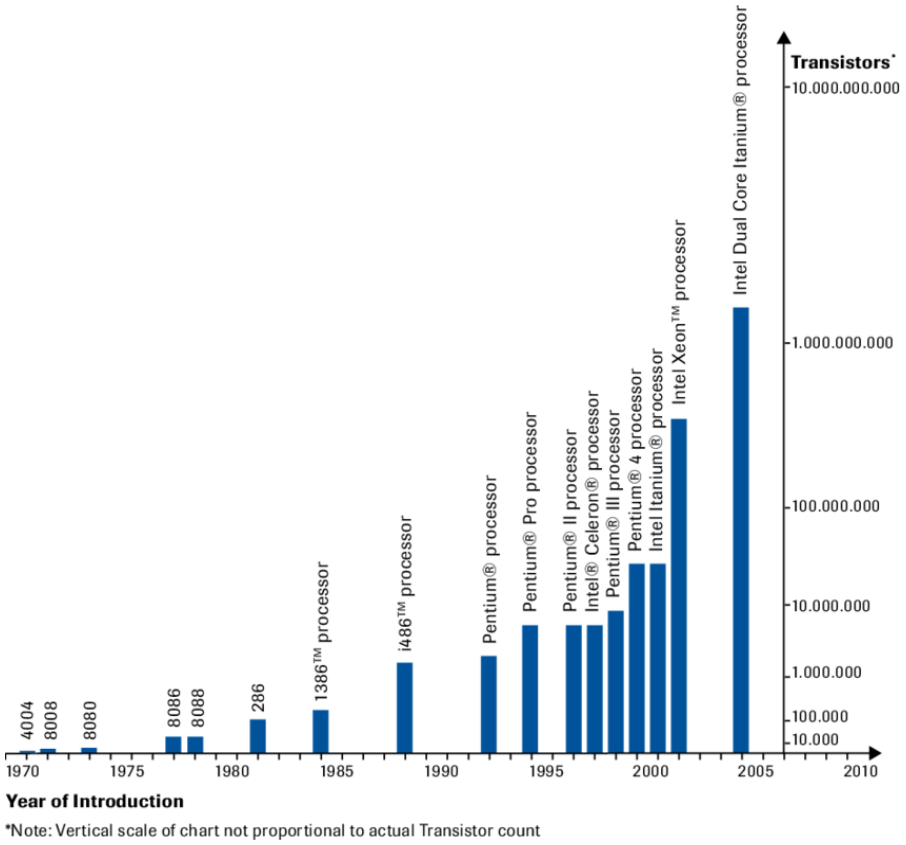

Рисунок 1.3 – Відображення кількості транзисторів у процесорах випущених в різні роки

Такі потужні можливості комп’ютерів відкривають широкі можливості для хакерів. Підбір хешованих паролів у деяких випадках стає можливим за лічені хвилини. Для більш складних випадків повний перебір паролів може зайняти від декількох днів, до десятків років. Проте зазвичай процес передору оптимізують використанням різноманітних словників, комбінацій, правил, тощо. Такі методи оптимізації використовуються найчастіше:

- використання слів із заздалегіть підготованого словника найчастіше

використовуємих слів для паролів;

- використання слів з додаванням усіх можливих символів чи чисел;
- використання слів з заміною букв на схожі цифри (наприклад «А» замінена на «4», «О» на «0» тощо).

У своєму експерименті Аліса Хеншау використала перелічені вище правила при переборі паролів з 14 мільйонів записів [7], що були отримані та оприлюднені хакерами під час однієї з атак. Результати виявились приголомшливими (рисунок 1.4), вже за 2 години було підібрано 50% паролів, а через 20 годин підібрано вже 80%.

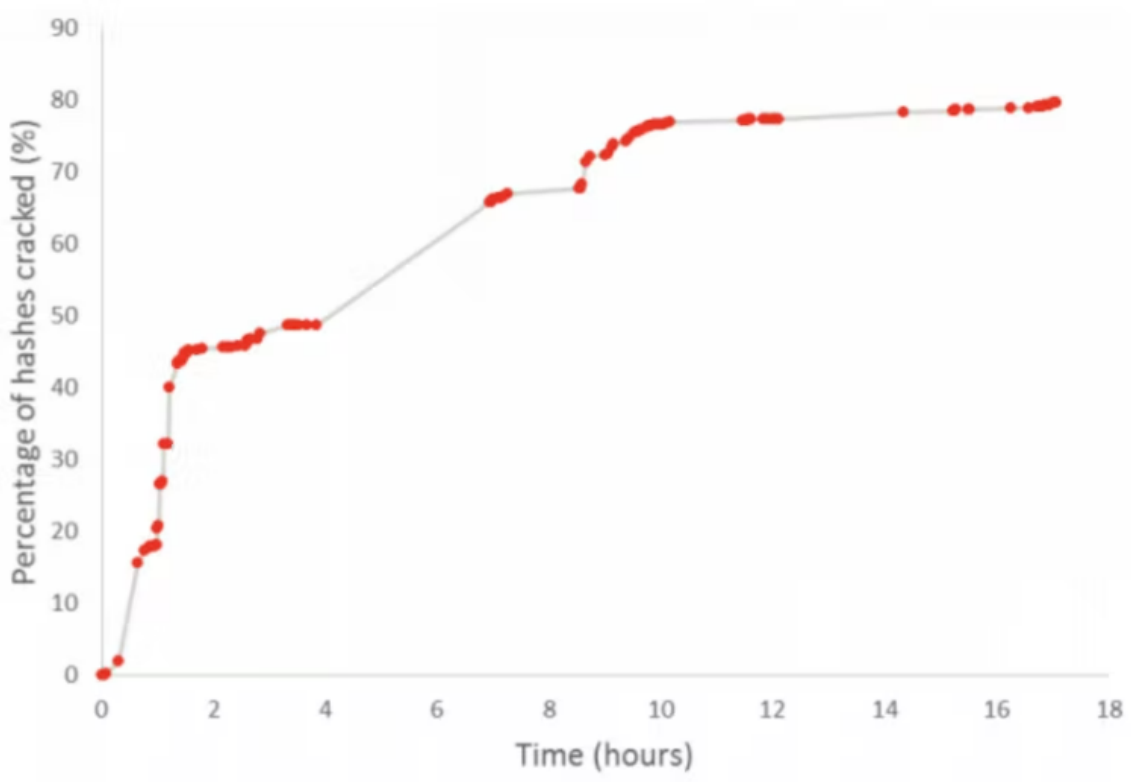

Рисунок 1.4 – Співвідношення підібраних паролів за проміжок часу виконання перебору

Оцінюючи проведений експеримент можна зробити висновки, що вміст паролю відіграє найбільш значиму роль у захисті доступу до особистих даних і, на жаль, більшість користувачів нехтують цим фактором. Навіть невеликі зміни пароля можуть дати значимий приріст до захищенності даних

людини. В ідеальному випадку в сучасному світі надійний пароль має відповідати наступним критеріям:

- довжина паролю мінімум 12 символів;
- вміщувати як великі, так і малі букви;
- вміщувати як букви, так і цифри;
- вміщувати спеціальні символи.

Використання паролів, що відповідають наведеним правилам забезпечить достатню надійність, щоб при оприлюднені бази даних з хешами паролів, не втратити доступ до своїх даних за лічені хвилини.

Компанія Hive Systems, що працює над різноманітними рішеннями пов’язаними з кібербезпекою підрахувала скільки знадобиться часу сучасному комп’ютеру для повного перебору паролів, що відповідають різноманітним правилам (рисунок 1.5). Отримані дані цілком та повністю корелюють з критеріями надійного пароля та підтверджують їх доцільність [4].

За підрахунками, короткі паролі, що складаються з 5-6 символів, можуть бути підібрані за лічені секунди, навіть не зважаючи на комбінації букв у різних регістрах, наявності спецсимволів, тощо. Починаючи з 8 символів час, необхідний для підбору паролю, стрімко збільшується до декількох годин, а то й навіть тижнів, в залежності від поєднання різноманітних комбінацій регістрів та спецсимволів. Найбільша ступінь надійності досягається при використанні паролю довжиною 12 символів та більше, навіть найпростіший пароль, що складається лише з букв однакового регістру займе декілька тижнів, а якщо ускладнити його різноманітними комбінаціями, складність зросте до тисяч років, необхідних для підбору такого пароля.

Таким чином, відповідальність користувача за безпеку своїх даних стає очевидною, свідомі люди, що переживають за свою безпеку вже давно використовують всі описані критерії для убезпечення свого паролю, проте залишається дуже вилка кількість людей, що не вважають це за необхідне.

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	61tm years	100tn years	7qd years

Рисунок 1.5 – Відображення часу, необхідного для повного перебору паролю, що відповідає різноманітним правилам

Повністю змусити всіх людей використовувати величезні паролі з купою спецсимволів, звісно, не вийде, проте можливо встановити в системі обмеження на використання паролів тієї, чи іншої довжини, наявності спецсимволів, цифр, та букв у різних регістрах. За останні декілька років такі обмеження набули широкої популярності серед різноманітних систем, тому зараз дуже складно знайти систему, яка б дозволила зареєструвати акаунт з паролем, що вміщує в собі менше 8 символів. Така цифра є тим мінімумом, який надає хоч якусь значиму безпеку при повному переборі та для деяких користувачів стає поштовхом до використання паролів ще більшої довжини.

1.3 Аналіз біометричних даних як методу аутентифікації

Біометрія – це метод вимірювання фізичних характеристик людини з метою підтвердження її особи. Щоб бути максимально ефективними, біометричні дані мають бути унікальними, незмінними та в будь-який момент готовими до їх збору, тому далеко не кожна риса людини підходить для таких цілей. Цей процес може включати в себе різноманітні фізіологічні особливості, як унікальні візерунки райдужки ока, відбитки пальців, індивідуальні шаблони вен долонь та пальців, голос людини.

Біометрична аутентифікація – це процес в кібербезпеці, який перевіряє особу користувача за допомогою його унікальних біологічних характеристик, таким же чином, як і зазвичай користувач робить це за допомогою пароля. Шляхом порівняння так званого біометричного підпису користувача з надісланим раніше підписом, що зберігається в базі даних, системи біометричної аутентифікації можуть визначити, що саме за особа намагається отримати доступ в систему, чи до даних, тощо. Такі системи перевіряють, чи справді ви той, за кого себе видаєте.

Біометрична аутентифікація являє собою технологію, яка швидко розвивається, і за правильного впровадження вона може допомогти вашій організації забезпечити доступ до конфіденційної інформації лише потрібним людям.

Сьогодні існує низка біометричних систем аутентифікації, деякі з яких все ще досліджуються для впровадження у більш широке застосування, проте інші широко використовуються вже зараз. Нижче наведено список технологій, які зараз стрімко використовуються, чи розвиваються:

- розпізнавання відбитків пальців;
- розпізнавання обличчя;
- розпізнавання райдужки та сітківки ока;
- розпізнавання голосу.

Розпізнавання відбитків пальців – це автоматизований процес

ідентифікації або підтвердження особи на основі порівняння двох відбитків пальців. Розпізнавання відбитків пальців є одним із найвідоміших біометричних методів і, безумовно, найпоширенішим біометричним рішенням для аутентифікації в комп'ютерних системах. Причини популярності розпізнавання відбитків пальців полягають у простоті отримання, та знаходження відмінностей, порівняно з іншими біометричними даними, а також у тому, що існує кілька джерел цієї біометрії для кожної людини, адже для цього методу можна використовувати різні пальці, що відповідно матимуть різні візерунки.

Три основні візерунки ребер відбитків пальців, що використовуються для розпізнавання: арка, петля та гофр [5]. Арка – це візерунок, де гребінь входить до однієї сторони пальця, потім піднімається посередині, утворюючи арку, і виходить з іншої сторони пальця. За допомогою петлі гребінець входить з одного боку пальця, потім утворює вигин і виходить з того ж боку пальця, з якого він вийшов. Петлі є найпоширенішим візерунком у відбитках пальців. І, нарешті, завиток – це візерунок, який ви маєте, коли хребти утворюють коло навколо центральної точки.

Проте важливішими у розпізнаванні відбитків пальців є саме дрібні деталі (рисунок 1.6). Дрібні точки вказують на певні зони на відбитку пальця, це дрібні деталі на відбитку пальця.

Існує три основні типи дрібних елементів: закінчення гребня, розгалуження та крапка (також відома як короткий гребінь). Закінчення хребта, як випливає з назви, це така ділянка, де закінчується хребет. Для розгалуження характерні зони, де хребет розділяється на декілька хребтів. Плями ж представляють собою хребти відбитків пальців, які набагато коротші за інші виступи. Також існує велика кількість інших типів елементів, проте вони використовуються не так часто.

Існує чотири основні типи обладнання для зчитування відбитків пальців:

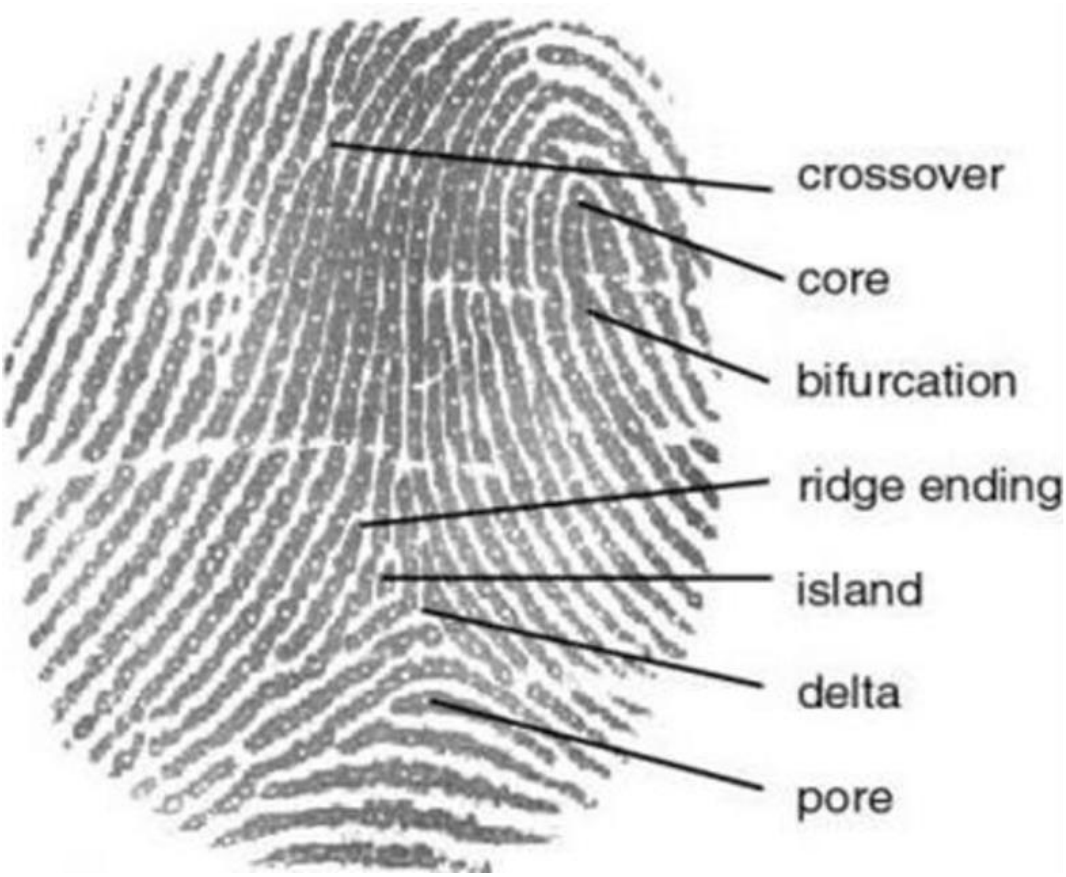

Рисунок 1.6 – Дрібні деталі, що є унікальними для кожного відбитку пальця

- оптичні;
- ємнісні;
- ультразвукові;
- термальні.

Оптичні зчитувачі є найпоширенішим типом зчитувачів відбитків пальців. Типом датчика в оптичному зчитувальному пристрої є цифрова камера, яка отримує візуальне зображення відбитка пальця. Переваги полягають у тому, що оптичні зчитувачі починаються з дуже низьких цін. Недоліки полягають у тому, що на зчитування впливають брудні або плямисті пальці, і цей тип зчитувача відбитків легше обдурити, ніж інші.

Ультразвукові пристрої для зчитування є найсучаснішим типом зчитувачів відбитків пальців, вони використовують високочастотні звукові хвилі для проникнення в епідермальний (зовнішній) шар шкіри. Вони

зчитують відбиток пальця на дермальному шарі шкіри, усуваючи потребу в чистій поверхні без рубців. Усі інші типи зчитувачів відбитків пальців фіксують зображення зовнішньої поверхні, тому перед зчитуванням руки мають бути чистими та без шрамів. Цей тип зчитувача відбитків пальців набагато дорожчий, ніж перші два, але завдяки своїй точності та тому, що їх важко обдурити, ультразвукові зчитувачі вирішують проблеми, які є в інших типів пристроїв, тому потрохи набирають популярності.

Розпізнавання обличчя – метод аутентифікації, який зчитує дані про обличя, аналізує та потім підтверджує особу на фотографії чи відео. Цей метод є одним із найпотужніших інструментів безпеки, що були створені людством. Хоча багато людей взаємодіють із функцією розпізнавання обличчя лише як спосіб розблокувати свої телефони чи сортувати свої фотографії, те, як різноманітні компанії та уряд використовують цю технологію, матиме набагато більший вплив на життя людей.

Більшість людей десятиліттями бачили розпізнавання обличчя та процес його роботи у безлічі фільмах, але його дуже рідко зображують належним чином. Кожна система розпізнавання обличчя працює по-різному, часто на основі власних алгоритмів, але можна розділити процес на три основні етапи:

- виявлення;
- аналіз;
- порівняння.

Виявлення – це процес пошуку обличчя на зображенні. Якщо ви коли-небудь користувалися камерою, яка розпізнає обличчя та малює рамку навколо нього для автофокусування, ви бачили цю технологію в дії. Виявлення обличчя зосереджується лише на його пошуку, а не на ідентифікації.

Аналіз (також відомий як відображення) – це етап, який відтворює обличчя, часто шляхом вимірювання відстані між очима, форми підборіддя, відстані між носом і ротом та інші риси, а потім часто перетворює це на ряд

чисел або точок. називають «відбитком обличчя». Хоча під час аналізу можуть виникати помилки, включно з неправильною ідентифікацією, зазвичай це є проблема лише на етапі, коли відбиток обличчя додається до бази даних розпізнавання. При порівнянні ж із вже існуючими записами в базі даних це не є великою проблемою, оскільки може розглядатись як похибка.

Порівняння – це спроба підтвердити особу людини за фотографією, а саме за рисами, що були виявлені та закодовані на попередньому кроці. Ця процедура використовується для перевірки, наприклад, за допомогою функції безпеки новіших смартфонів, або для ідентифікації, яка намагається відповісти хто на тій чи іншій фотографії.

Вище розглянуті основні етапи, які можна застосувати до будь якого з існуючих методів, проте кожен етап може мати різну реалізацію в залежності від програмного забезпечення там приладів, що використовуються для отримання інформації про обличчя. Основними варіантами на сьогодні є звичайні камера, та спеціальні сканери (рисунок 1.7).

Рисунок 1.7 – Об’ємно-просторовий сканер обличчя

Звичайні камери зазвичай полягають саме на зображення, тому часто страждають від поганого освітлення та якості зображення. Методи, що використовують камеру часто можна обійти, використавши звичайне фото людини. Сканери ж, такі як наприклад Face ID, або ж аналогічні пристрої, отримують об’ємно-просторове зображення обличчя, що є значно надійнішою, хоч і дорожчою технологією.

Використання просторової моделі обличчя вирішує основні проблем, що мають звичайні камери, такі сканери можуть працювати навіть у повній темряві, що значно зручніше для користувача.

Розпізнавання райдужної оболонки ока – це біометричний метод аутентифікації, який порівнює унікальні риси та візерунки кольорової частини ока для перевірки та аутентифікації особистості людини. Подібно до зіставлення відбитків пальців, розпізнавання райдужної оболонки (рисунок 1.8) є більш точним, не вимагає фізичного контакту та пропонує більше можливостей використання, включаючи можливість працювати на більшій відстані, ніж за відбитками пальців або долонь.

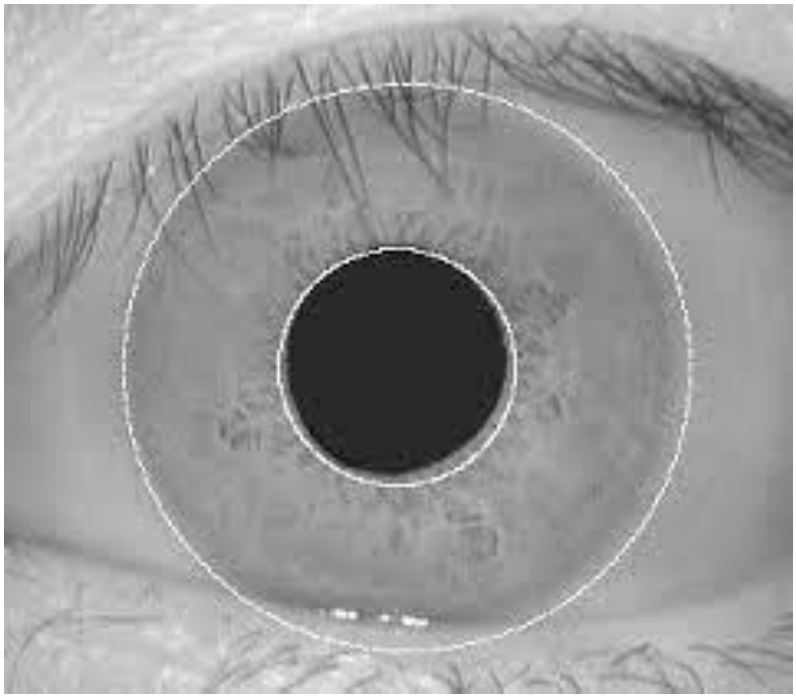

Рисунок 1.8 – Райдужна оболонка ока

Розпізнавання райдужної оболонки ока працює шляхом захоплення зображення та кодування унікальних рис (приблизно 240 рис райдужної оболонки ока) для створення біометричного шаблону, необхідного для пошуку та порівняння. Однак, на відміну від розпізнавання обличчя, для розпізнавання райдужної оболонки потрібна спеціальна цифрова камера, яка використовує невидиме інфрачервоне світло для отримання висококонтрастного зображення райдужної оболонки та унікальних візерунків, що неможливо побачити без спеціальних приладів [6]. Ці камери можна тримати в руках або монтувати для різних випадків використання.

Розпізнавання голосу – це процес автоматичного розпізнавання голосу людини за інформацією про мовця, що міститься в звукових хвилях, для перевірки тієї чи іншої особи, на наявність доступу до певної системи, чи інформації.

Ідентичність голосу співвідноситься з фізіологічними та поведінковими властивостями мовної системи кожної людини. Ці характеристики походять як від спектральної оболонки (особливості голосового тракту), так і від супрасегментарних особливостей (особливості джерела голосу) мови. Найбільш часто для потреб розпізнавання голосу користуються короткочасними спектральними вимірюваннями є кепстральні коефіцієнти та їх коефіцієнти регресії.

Зазвичай прийнято виділяти два методи розпізнавання голосу: тексто-залежний та тексто-незалежний [12]. У випадку з тексто-незалежним розпізнаванням людині достатньо сказати будь-яку речення для процесу розпізнавання. Такий метод має свої недоліки, адже в такому випадку шахрай зможе використати завчасно зроблений запис голосу людини. Тексто-залежний метод покладається на певну фразу, що була використана під час реєстрації користувача.

Таким чином можна встановити фразу, яку в реальному житті людина не використає в жодному разі, забезпечивши більшу надійність при використанні розпізнавання голосу для аутентифікації.

1.4 Аналіз мультифакторної аутентифікації

Вище були описані основні методи, що використовуються сучасними системами для забезпечення безпечного доступу до конфіденційних даних користувачів. Кожен з них має свої переваги та недоліки, особливості використання, рівень безпеки, та безліч інших різноманітних факторів, що може спонукати користувача надавати перевагу одному методу над іншим. Проте, при використанні кожного з них, є один нюанс. Якщо шахраям вдасться обійти обраний вами метод, можна сміливо вважати, що доступ до ваших даних буде втрачено за лічені хвилини. Рано, чи пізно такий сценарій може статись із кожним користувачем, будь-то через необачність самого користувача, чи завдяки постійному розвитку методів злому, що дозволять зламати навіть найсучасніші методи безпеки. Для забезпечення більшої надійності основних методів, описаних раніше, зараз використовують мультифакторну аутентифікацію.

Мультифакторна аутентифікація – метод аутентифікації, що використовується для забезпечення додаткового прошарку безпеки при наданні доступу до конфіденційної інформації. Суть MFA полягає у використанні декількох методів аутентифікації одночасно.

Найпоширенішим варіантом мультифакторної аутентифікації є двохфакторна аутентифікація. Двохфакторна аутентифікація використовує два методи для аутентифікації користувача. Перший метод є основним, зазвичай це пароль, чи біометрична аутентифікація. Другий метод є другорядним, такий метод виконує роль додаткового захисту у разі, якщо шахраю вдалося обійти основним метод. У ролі другого методу найчастіше використовується код, що надсилається у вигляді повідомлення на мобільний телефон користувача [10], оскільки телефон зазвичай постійно присутній поряд з людиною, проте можливий також варіант з використанням більш серйозних методів, як біометрія. Також можна використовувати мультифакторну аутентифікацію з більшою кількістю методів, але такий

підхід погіршує досвід користувача, якому стає необхідним витрачати велику кількість часу на проходження безлічі безпекових мір. Тому оптимальним варіантом мультифакторної аутентифікації прийнято вважати саме двохфакторну.

Таким чином можна зробити висновок, що мультифакторна аутентифікаця значно зменшує ризик втрати доступу над своїми особистими даними за рахунок введення додаткових методів аутентифікації. Надійно підібраний додатковий метод аутентифікації зробить майже неможливим будь-який несанкціонований доступ до акаунту, тому мультифакторна аутентифікація повинна бути обов’язким атрибутом сервісів, що потребують максимального захисту даних, таких як банки, державні ресурси, тощо.

1.5 Обґрунтування переваг використання мобільного пристрою для проведення процесу аутентифікації

Як було зазначено раніше, захист конфіденційних даних є важливою складовою сучасного цифрового світу. Методи захисту доступу до такої інформації з кожним роком стають все прогресивнішими. Наразі одними з провідних методів є методи, що використовують біометричні дані людини. Біометричні дані є високонадійними унікальними рисами людини, є дуже швидкими в зборі та, за умови використання сучасних алгоритмів та рішень, є відносно швидкими в обробці. В поєднанні з використанням мультифакторної аутентифікації така комбінація представляє собою дуже надійне рішення для захисту даних.

Таким чином зроблено висновки, що мобільний додаток є оптимальним варіантом для імплементації такої комбінації двохфакторної аутентифікації з використанням біометричних даних, оскільки зазвичай, смартфони вже оснащені усіма необхідними приладами для зчитування потрібної інформації: камерою, мікрофоном, датчиками для розпізнавання обличчя, тощо.

1.6 Постановка завдань дослідження

Як вже було показано, класичні методи аутентифікації, зазвичай, базуються на статичних даних, це можуть бути паролі, фото обличчя, відбитки пальців. Такі дані є очевидною ціллю, що необхідно подолати шахраям для доступу до отримання чужих даних. Значно більш перспективною є двофакторна аутентифікація, яка вимагає від користувача, окрім основного, встановити ще один додатковий метод. Такий підхід безумовно значно покращує безпековий аспект аутентифікації, проте змушує балансувати між тим, щоб зробити процес більш надійним, але й не зробити його дуже складним та часовитратним для користувача. Отже існує необхідність обґрунтувати і реалізувати перспективні підходи і технології для покращення результатів цього процесу. Різноманітні додаткові дані про користувача можуть зіграти визначальну роль в розвитку процесу аутентифікації. Так система може опиратись на місце знаходження людини за IP-адресою, ідентифікаційні параметри пристрою, з якого був здійснений вхід до системи, тощо. Всі ці фактори можуть слугувати важливими індикаторами, для тимчасового блокування доступу, якщо людина використовує її з незвичного місця, чи робить невластиві їй речі.

У зв'язку із зазначеним актуальною є розв’язання науково-прикладної задачі дослідження методу і алгоритму покращення аутентифікації людини. Відповідно до зазначеної мети необхідно розв’язати наступні часткові завдання дослідження:

- провести аналіз різноманітних технологій і алгоритмів аутентифікації;
- виконати огляд сучасних середовищ розробки, технологій створення застосунків для мобільних пристроїв та хмарних біометричних сервісів;
- розробити архітектуру програмного продукту та навести опис технічної реалізації застосунку;
- навести опис інтерфейсу користувача та використання застосунку сторонніми сервісами у якості двохфакторної аутентифікації.

2 ОГЛЯД СУЧАСНИХ ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Огляд сучасних середовищ розробки та технологій для створення застосунків для мобільних пристроїв

З кожним днем індустрія розробки програмного забезпечення набирає все більших обертів. Якщо перші програми були націлені на демонстрацію роботи тих, чи інших найпримітивніших алгоритмів, то зараз же сучасні програми несуть в собі все більше цінності та користі для користувача.

Оскільки, для цілей демонстрації сучасних методів аутентифікації, цільовою платформою було вирішено обрати мобільні девайси, то розглянемо саме технології та середовища розробки, що безпосередньо відносяться до розробки під мобільні пристрої.

За останнє десятиріччя мобільні пристрої зайняли провідне місце у житті рядового користувача цифрових технологій. Майже кожна людина зараз має у своєму кармані смартфон, який допомагає їй вирішувати повсякденні питання. З таким стрімким входженням смартфонів у наше життя, звичайно, став великим і попит на розробку застосунків для таких пристроїв. На разі існує вже створено велику кількість бібліотек, націлених саме на мобільну розробку, що встигли завоювати велику популярність серед розробників [15]. Найпопулярнішими бібліотеками на сьогоднішній день є:

- React Native;
- Flutter;
- Xamarin/MAUI;
- Swiftic;
- Ionic;
- Apache Cordova;
- jQuery Mobile;
- Native Scripts.

React Native, розроблений і підтримується Facebook, є доступною міжплатформною системою розробки мобільних застосунків, яка швидко стала одним з найкращіх варіантів для розробників. React Native дозволяє легко розробляти мобільні програми для Android та iOS. Найбільш показовими прикладами програм React Native є програми відомих компаній, таких як Tesla, Airbnb, Skype або Amazon Prime. React Native входить до списку найкращих фреймворків для розробки мобільних застосунків. Він дозволяє створювати версії кількох функцій для певної платформи, полегшуючи використання єдиного коду на різних платформах, що дуже полюбилося розробникам. Основна перевага React Native полягає в тому, що він дозволяє швидко розробляти та впроваджувати нові функції. Елементи, розроблені за допомогою React Native, можуть бути знов використані безліч разів у різноманітних частинах системи, сумісність із сторонніми розширеннями та створення графічного інтерфейсу на основі компонентів для інтерфейсних програм є іншими ключовими функціями React Native.

Flutter – це відкрита та безкоштовна платформа від Google, яка дозволяє створювати нативні програми для Android та iOS за допомогою простої кодової бази. Це революційний бібліотека для кросплатформної розробки застосунків, яка пропонує новий спосіб створення власних застосунків. Flutter – це комплексна та точна структура, яка включає віджети, механізм візуалізації, інтерфейси для налагодження та інтеграції, а також ресурси, які допомагають розробникам створювати та розгортати мобільні програми з приємним інтерфейсом користувача. Flutter використовує низка відомих організацій, зокрема Google і Abbey Road Studios.

Xamarin – це ще одна кросплатформна платформа розробки застосунків для Android та iOS. Оскільки для розробки на Xamarin використовують мову програмування C#, програми стає простіше відлажувати. В результаті процес розробки відбувається швидше. Крім того, це дозволяє розробнику швидко переносити сценарії на інші системи, такі як Windows і macOS. Microsoft придбала Xamarin та наразі підтримує та покращує цей фреймворк, що

робить його більш привабливим, за рахунок великої корпорації, що його підтримує. Оскільки розробка застосунків сьогодні набагато швидша, можна припустити, що швидша розробка означає жертвування якістю та дизайном. Проте програми на основі Xamarin пропонують бездоганну нативну функціональність з точки зору якості та ефективності. Наразі компанія Microsoft продовжує розвивати фреймворк, проте трохи у іншому вигляді. Зараз вони працюють над розробкою MAUI, який можна вважати продовженням Xamarin, оскільки він побудований на його базі.

Swiftic – це платформа для мобільних застосунків, яка дозволяє кожному легко створювати унікальні застосунки для свого бізнесу. Цей фреймворк спрощує розробку застосунку, дозволяючи програмістам інтегрувати наявні матеріали з інтернету замість того, щоб починати з нуля. Це одна з дуже гнучких систем розробки мобільних застосунків, оскільки вона пропонує досить зручну взаємодію, та прості стратегії розробки. Push-повідомлення, стрічки платформ соціальних мереж, реклама застосунків та інші технологічні досягнення включені в фреймворк. Це одне з найпростіших рішень для розробки мобільних програм, оскільки ви можете створювати, запускати та розширювати свої програми з єдиної панелі керування.

Ionic – це чітка структура для створення прогресивних веб-застосунків, гібридних і кросплатформних мобільних застосунків. Ionic – це доступний фреймворк, який використовує Apache Cordova (PhoneGap) і Angular, щоб дозволити розробникам створювати програми для Android та iOS, які бездоганно працюють разом. Фреймворк допомагає розробникам створювати надійні та багатофункціональні нативні програми. Найвидатнішою особливістю Ionic є те, що він дозволяє розробникам використовувати різні компоненти інтерфейсу користувача в структурі застосунків, включаючи фільтрацію, введення, перегляди, зручну навігацію та таблиці дій.

Apache Cordova, який раніше називався PhoneGap, є популярним фреймворком розробки мобільних застосунків. Це міжплатформний фреймворк для розробки мобільних програми за допомогою CSS3, HTML5 і

JavaScript. Плагіни Cordova дозволяють програмістам використовувати такі апаратні функції смартфонів, як GPS, камери та акселерометри, щоб запропонувати нативну підтримку відповідних модулів. Apache Cordova пропонує швидкий підхід до створення єдиного коду.

jQuery Mobile – це кросплатформенний фреймворк для розробки мобільних застосунків, який підтримує розробку нативних кросплатформенних застосунків. Цей фреймворк використовується для створення гнучких веб-порталів, які працюють на різних платформах, включаючи смартфони, планшети та ПК, за допомогою JavaScript і HTML.

Native Script – це фреймворк з відкритим кодом для створення власних мобільних застосунків, який підкріплюється Angular, Typescript, JavaScript, CSS і Vue.js. Його спроектувала та розробила компанія Progress. NativeScript і всі необхідні плагіни встановлюються через менеджер пакетів npm. Застосунки створюються, налаштовуються та компілюються за допомогою командного рядка або графічного інтерфейсу користувача під назвою NativeScript Sidekick. Незалежні від платформи інтерфейси користувача визначаються за допомогою файлів XML. Потім NativeScript обробляє вихідні дані, представлені у файлах XML, щоб викликати власні компоненти інтерфейсу користувача кожної окремої платформи. Логіку застосунків, розроблену в Angular і TypeScript, також можна створити незалежно від цільової платформи.

Вище описані фреймворки націлені саме на кросплатформену розробку, щоб забезпечити якомога більшу кількість користувачів можливістю використовувати розроблений застосунок, проте є й такі фреймворки, що використовуються лише для однієї цільової платформи. Такими наприклад є низка фреймворків мови Swift, що використовуються для розробки під IOS системи. Такі фреймворки, вузьконаправлені на певну платформу в даній роботі розглядатись не має сенсу.

Всі фреймворки описані вище використовуються саме для розробки інтерфейсу користувача, проте розробка мобільного застосунку не

обмежується тільки цим. Мобільний застосунок слід розглядати більш глобально. Зазвичай більшість застосунків використовують клієнт-серверну модель, де клієнтом є саме застосунок, що представляє собою інтерфейс користувача, а сервер якусь API, що виконує ті, або інші операції необхідні для нормального функціонування застосунку. Розробка API також є дуже широкою. темою, що має велику кількість потужних фреймворків для вирішення різноманітних задач. Найпопулярнішими фреймворками для створення API на сьогодні є:

- Django;
- Express.js;
- Flask;
- Laravel;
- Ruby on Rails;
- Spring;
- ASP.NET Core.

Django – це веб-фреймворк на основі Python, зосереджений на створенні веб-сайтів, зазвичай керованих базами даних, проте може бути використаний і для інших видів сайтів. Якщо високоефективний веб-сайт є вашим пріоритетом, вам слід вибрати Django. Фреймворк відомий як дуже масштабований, зокрема, можливість повторного використання коду полегшує розробникам адаптацію до збільшення відвідуваності веб-сайту. Як дуже популярний бекенд-фреймворк, Django може розраховувати на велику спільноту, готову підтримати будь-кого, хто використовує інструмент.

Express.js – це фреймворк для веб-застосунків Node.js. Express.js є дуже популярним серверним фреймворком, який використовувався для створення відомих веб-застосунків, таких як MySpace і Uber. Для фреймворку доступно багато стандартних функцій Node.js, які можуть полегшити роботу розробника. Також Express пропонує кілька шаблонів, які допоможуть створювати веб-сторінки з меншими зусиллями.

Flask – це легкий фреймворк WSGI для веб-застосунків. Він

розроблений, щоб зробити початок роботи швидким і легким, з можливістю масштабування до складних програм. Він починався як проста обгортка навколо інструменту Werkzeug та Jinja, а потім перетворився на один із найпопулярніших фреймворків для веб-застосунків Python. Flask пропонує використовувати ті, чи інші інструменти, а не ставить розробника перед фактом, маючи купу залежностей, що можуть і не знадобитись. Розробник самостійно вибирає інструменти та бібліотеки, які він хоче використовувати. Спільнота пропонує багато розширень, які спрощують додавання нових функцій.

Laravel – це безкоштовний фреймворк PHP з відкритим кодом, спеціально розроблений для створення складних веб-сайтів і веб-застосунків. Дозволяє спростити аутентифікацію, маршрутизацію, сеанси, кешування, архітектуру програми та роботу з базою даних.

Ruby on Rails – це фреймворк веб-розробки з відкритим кодом на основі мови програмування Ruby. Ruby не є найпопулярнішою мовою програмування, але Ruby on Rails є дуже популярним серверним фреймворком, оскільки він допомагає розробникам у процесі програмування, пропонуючи простоту, автоматизоване тестування та бібліотеки.

Spring – це платформа веб-розробки, яка використовує мову програмування Java. Він використовувався для створення багатьох дуже популярних і дуже функціональних веб-сайтів, таких як Wix, складна платформа для створення веб-сайтів. Заснований на Java, дуже популярній мові програмування, Spring користується великою популярністю та підтримкою великої спільноти. Він містить багато дочірніх проектів, які можуть покращити продуктивність Spring. Також фреймворк дуже масштабований, особливо з використанням дочірніх проектів, які постачаються з ним.

ASP.NET Core – це технологія, розроблена Microsoft для створення веб-застосунків на платформі .NET. C# і F# використовуються як мови програмування для розробки застосунків на ASP.NET Core. ASP.NET Core

також є кросплатформним фреймворком, високопродуктивною структурою з відкритим кодом, що все частіше використовується для створення сучасних хмарних програм.

Розглянувши як технології для розробки мобільних застосунків, так і для розробки API можна зробити висновки, що популярні фреймворки для розробки повноцінного застосунку можуть використовувати різноманітні мови програмування. Робота з тією, чи іншою мовою програмування зазвичай потребує використання відповідного середовища розробки. Найбільш розповсюдженими середовищами на сьогодні для різних мов програмування є:

- Visual Studio (C++, C#, F#);
- Visual Studio Code (безліч мов, залежить від встановлених плагінів);
- Rider (C#);
- PyCharm (Python);
- InteliJ Idea (Java);
- Eclipse (Java);
- PhpStorm (PHP).

2.2 Обґрунтування обраних технологій для розробки мобільного застосунку

Завчасне планування різноманітних деталей проекту є дуже важливою темою на яку не слід економити час. Необдуманий вибір, зроблений на скору руку, може втратити команді розробки в десятки разів більше часу, аніж час, який би вони витратили на попереднє планування.

Вибір технології розробки є одним з таких факторів. Хоч і може здаватись, що з сучасним рівнем розвитку технологій, вони всі мають приблизно рівні можливості, проте все ж різноманітні особливості того, чи іншого фреймворку, можуть відігравати велику роль у подальшій долі проекту.

Розглядаючи питання вибору технологій для розробки застосунку треба брати до уваги декілька важливих моментів, а саме:

- можливість легкого розширення застосунку у майбутньому;
- підтримка та розвиток фреймворку;
- розмір ком’юніті;
- рівень входу для початку розробки з обраною технологією.

Розглядаючи фреймворки для розробки мобільних застосунків можна сміливо сказати, більшість з фреймворків, що були розглянуті раніше, є дуже сучасними і мають велику кількість пунктів, що забезпечують гарні можливості для розширення застосунку, при необхідності додавання нового функціоналу.

Щодо підтримки та розвитку фреймворка, все вже не так гарно. Велика кількість команд розробки, при старті нових проектів, дуже турбуються про факт, що обрана ними технологія перестане оновлюватись, що у майбутньому може зробити застосунок технологічно застарілим та складним в підтримці та розширенні. З розглянутих раніше фреймворків для розробки мобільних застосунків можна виділити два основні фаворити: React Native, Flutter та Xamarin. Оскільки React Native розроблений компанією Facebook, Flutter компанією Google, а Xamarin компанією Microsoft, то можна не хвилюватись, що підтримка цих фреймворків закінчиться у найближчі роки. Коли такі великі корпорації вкладаються в розробку фреймворку, зазвичай вони планують довготривалу підтримку та мають великі надії на успіх таких проектів.

Розмір ком’юніті в трьох обраних раніше фреймворках приблизно однаково великий, проте є важливий момент відносно Xamarin. Оскільки Microsoft наразі почав більше вкладатись в розробку MAUI, який є більш новим та, відповідно, ще не має дуже великого ком’юніті, краще не ризикувати з обранням такого варіанту.

Щодо рівня входу для початку розробки потрібно зазначити, що React Native використовує JavaScript, або TypeScript, залежно від обраної

конфігурації, які широко відомі серед розробників, а Flutter використовує мову програмування Dart. Хоча вивчення Dart і не повинно бути дуже складною задачею, проте якийсь час все одно потрібно витратити на вивчення. Також великим плюсом React Native є те, що цей фреймворк працює за тим же принципом, що й звичайний React з тією лише відмінністю, що звичайний React для шаблонів використовує елементи HTML, на відміну від React Native, який використовує елементи особливі для мобільних пристроїв.

Таким чином React Native є більш прийнятним варіантом, враховуючи наявність попереднього досвіду з React. Велике ком'юніті React Native допоможе з вирішенням проблем, що постають у процесі розробки, а велика корпорація як Facebook надає більше спокою, що через декілька років проект не перетвориться на репозиторій з застарілим та непідтримуваним кодом.

Розглядаючи технології для розробки API можна сказати, що найпопулярніші фреймворки також знаходяться приблизно в однаковому положенні, проте деякі з них значно простіше використовувати з невеликими системами, такі як Express, інші ж, як ASP.NET Core, чи Spring використовуються для більш масштабних проектів, адже вони легші в масштабуванні та розширенні. Такі фреймворки зазвичай більш масивні та громіздкі, проте ASP.NET Core намагається вирішити це питання, вводячи Minimal API підхід [9], що значно зменшує кількість коду, що потрібна для роботи веб-застосунку, що може бути дуже корисно при розробці невеликих API.

Таким чином, враховуючи стрімкий розвиток, постійні оновлення, дуже велике ком'юніті та довгострокову підтримку, у якості фреймворку для розробки API було обрано ASP.NET Core.

Взявши до уваги технології обрані для розробки демонстраційного застосунку маємо декілька можливих середовищ розробки. Для розробки на ASP.NET Core з використанням мови програмування C# було обрано Rider, оскільки ця IDE стабільно працює на різноманітних платформах, на відміну

від Visual Studio, яка гарно справляється зі своїми задачами лише на Windows. Для розробки мобільного застосунку буде використано Visual Studio Code з додатково встановленими плагінами для роботи з React Native та TypeScript.

2.3 Огляд хмарних біометричних сервісів

Розробка технологій, що використовують біометричні дані людини потребує немалих зусиль. Основною складністю в такій розробці є поріг входу, адже команда розробки повинна гарно знати методи збору, обробки та зберігання подібної інформації. Іншим питанням є тестування таких розробок, адже системи, що працюють з біометричними даними, потребують максимальної точності в своїй роботі, та будь які помилки в таких системах можуть бути критичними. Таким чином процес розробки та тестування таких систем з нуля може займати дуже велику кількість часу.

Хмарні сервіси допомагають вирішити цю проблему. Хмарні сервіси – це набір IT-додатків і ресурсів, включаючи програмне забезпечення, інфраструктуру та платформи, розміщені третіми сторонами на власних серверах та надані організаціям і окремим клієнтам через Інтернет за запитом [18]. Їх також називають хмарними обчисленнями.

Теперішня популярність хмарних сервісів зумовлена їхньою відносною доступністю. Зазвичай користувач такими сервісами платить за кількість транзакцій, чи час, витрачений на ті, або інші обчислення у хмарі. Перед тим як надавати можливість використовувати той, чи інший сервіс, хмарний провайдер запевнюється, що він гарно протестований, що у більшості випадків вже робить використання таких сервісів значно дешевшим, ніж розробку власної реалізації з нуля.

Вже давно звичними для розробників стали хмарні сервіси, що надають можливість запускати у хмарі сервери зі своїми додатками, базами даних, чи використовувати обчислювальні можливості хмарних провайдерів. Проте з

кожним роком хмарні сервіси надають все більше нових та прогресивних можливостей для розробки. Не стали й виключенням сервіси, що надають можливості для роботи з біометричними даними людини.

Одним з найпопулярніших хмарних провайдерів, що мають сервіси, пов’язані з біометричними даними, є Microsoft Azure та його Azure Cognitive Services. Cognitive Services робить штучний інтелект доступним для кожного розробника та дослідника даних. Все, що потрібно, це виклик відповідний API. Azure Cognitive Services мають декілька сервісів, що задовольняють потреби для роботи з біометричними даними: Face API та Speaker recognition.

Face API надає можливості з розпізнавання облич. Цей сервіс може виділяти обличчя на фото, розпізнавати різноманітні емоції, вік та інші аспекти [20]. Також сервіс надає можливість саме розпізнавати людину, зображену на фото, чи відео, він без проблем розпізнає знаменитостей, про яких має інформацію у внутрішній базі даних, проте не є проблемою створити власну базу користувачів, серед яких у майбутньому проводити ідентифікацію облич на фото.

Speaker recognition – сервіс, що надає можливість працювати з голосом людини. Цей сервіс може ідентифікувати людину за її голосом. Ідентифікація може проводитись у декількох режимах, тексто-залежна, та тексто-незалежна. Тексто-назалежна ідентифікація орієнтується лише на сам голос, його коливання, тембр та інші властивості. Тексто-залежна ідентифікація ж додатково перевіряє чи вірна фраза була сказана у голосовому записі.

Деякі інші хмарні провайдери теж надають подібний функціонал. Серед популярних, хмарний провайдер AWS надає сервіс Amazon Rekognition, що дозволяє проводити велику кількість операцій над фото, включно з розпізнаванням та порівнянням облич. Є й інші хмарні провайдери з аналогічними можливостями, проте в даній роботі буде використано Azure Cognitive Services, оскільки платформа Microsoft Azure надає можливість отримати депозит в 200$ на вивчення та експерименти з різними сервісами при реєстрації.

3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ПІДТВЕРДЖЕННЯ БІОМЕТРИЧНОЇ АУТЕНТИФІКАЦІЇ

Основною метою застосунку є забезпечення двохфакторної аутентифікації. Застосунок надає можливість користуватись своїми функціями різним іншим додаткам. Таки чином, будь-яке програмне забезпечення, що потребує додаткового захисту процесу авторизації, має змогу використати розроблений в цій роботі застосунок для задовільнення таких цілей.

Під час процесу аутентифікації сервіси-користувачі даного застосунку покладаються на додаткове підтвердження особистості людини, що намагається увійти до системи. Тобто, розроблений застосунок представляє собою другий метод у двохфакторній аутентифікації до того, чи іншого сервісу. Після проведення тим, або іншим сервісом внутрішньої аутентифікації, наприклад перевірки паролю, сервіс буде робити додатковий запит до розробленого застосунку, щоб провести перевірку біометричних даних людини серед завчасно зібраної бази таких даних.

3.1 Розробка архітектури програмного продукту

Архітектура програмного забезпечення системи відображає організацію або структуру системи та пояснює, як вона така система поводиться. Система – це набір компонентів, які виконують певну функцію або набір функцій. Іншими словами, архітектура програмного забезпечення забезпечує міцну основу, на якій можна створювати програмне забезпечення. Ряд варіантів архітектури та компромісів впливають на якість, продуктивність, підтримуваність та загальний успіх системи. Ігнорування типових проблем і довгострокових наслідків на початку розробки може поставити таку систему під загрозу. Існує кілька високорівневих

архітектурних шаблонів і принципів, які зазвичай використовуються в сучасних системах.

В даному випадку за основу архітектури застосунку було взято тришарову архітектуру [21]. Така архітектура представляє собою розділення програмного забезпечення на три основні шари:

- Presentation;
- Business logic;
- Data.

Шар Data представляє собою частину, що зберігає дані, нехай то база даних, чи інші файли, з якими працює застосунок.

Шар Business logic вміщує в собі основну логіку програми. Якщо та, чи інша логіка потребує роботи з інформаціє в базі даних, цей шар може звернутись до Data шару.

Шар Presentation зазвичай є відповідальним за графічний інтерфейс користувача, або інші види інтерфейсів, що надають можливість користувачу працювати з сервісом. Цей шар єдиний, з яким користувач може взаємодіяти напряму. Власне звідси йдуть всі необхідні комунікації з Business logic шаром. Всі ці шари завжди взаємодіють відповідним чином (рисунок 3.1) та не повинні відхилятись від зазначених норм, адже це зруйнує всю суть.

Спроекціювавши таку архітектурну модель на додаток біометричної аутентифікації можна виділити основні частини системи, що будуть розроблені. Presentation шар в нашому випадку буде представлено мобільним застосунком. Користувач буде взаємодіяти з мобільним застосунком, який в свою черзу буде спілкуватися з API. Business Logic шар буде представлено API, яка буде проводити всі необхідні операції над вхідними даними. Data шар в даному випадку буде представлений звичайною базою даних.

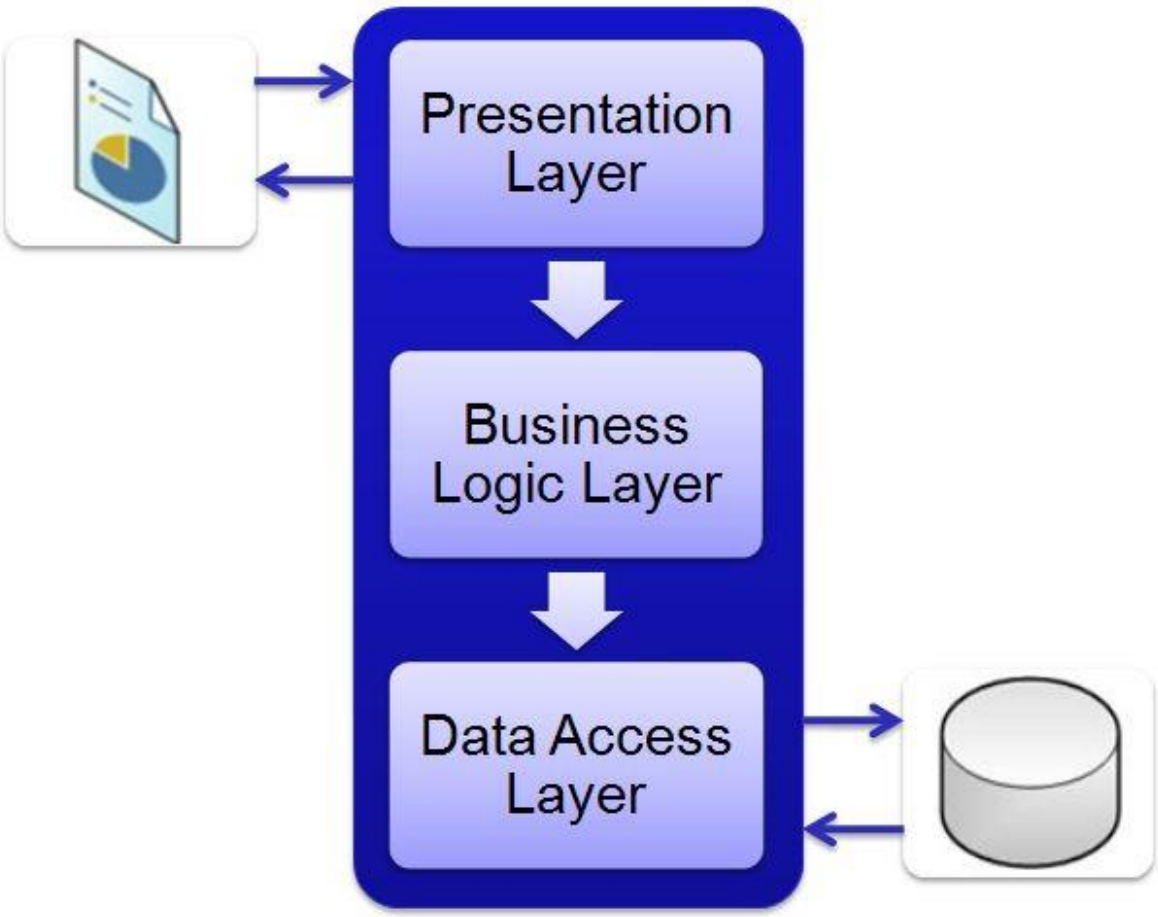

Рисунок 3.1 – Взаємодія шарів в тришаровій архітектурі програмних застосунків

Основні операції пов’язані з перевіркою особистості людини проводяться саме у шарі бізнес логіки. Проте цей шар та застосунок в цілому (рисунок 3.2) не є повністю самостійним, адже залежить від деякі хмарні сервіси. При необхідності опрацювати дані обличчя, чи голосу людини API буде звертатись до Face API та Speaker Recognition сервісів відповідно. Додатково застосунок використовує Firebase Cloud Messaging сервіс для роботи з вдправкою повідомлень на мобільні пристрої.

Firebase Cloud Messaging в цьому випадку як раз забезпечує стабільну можливість відправки повідомлень необхідних для справної роботи двохфакторної аутентифікації.

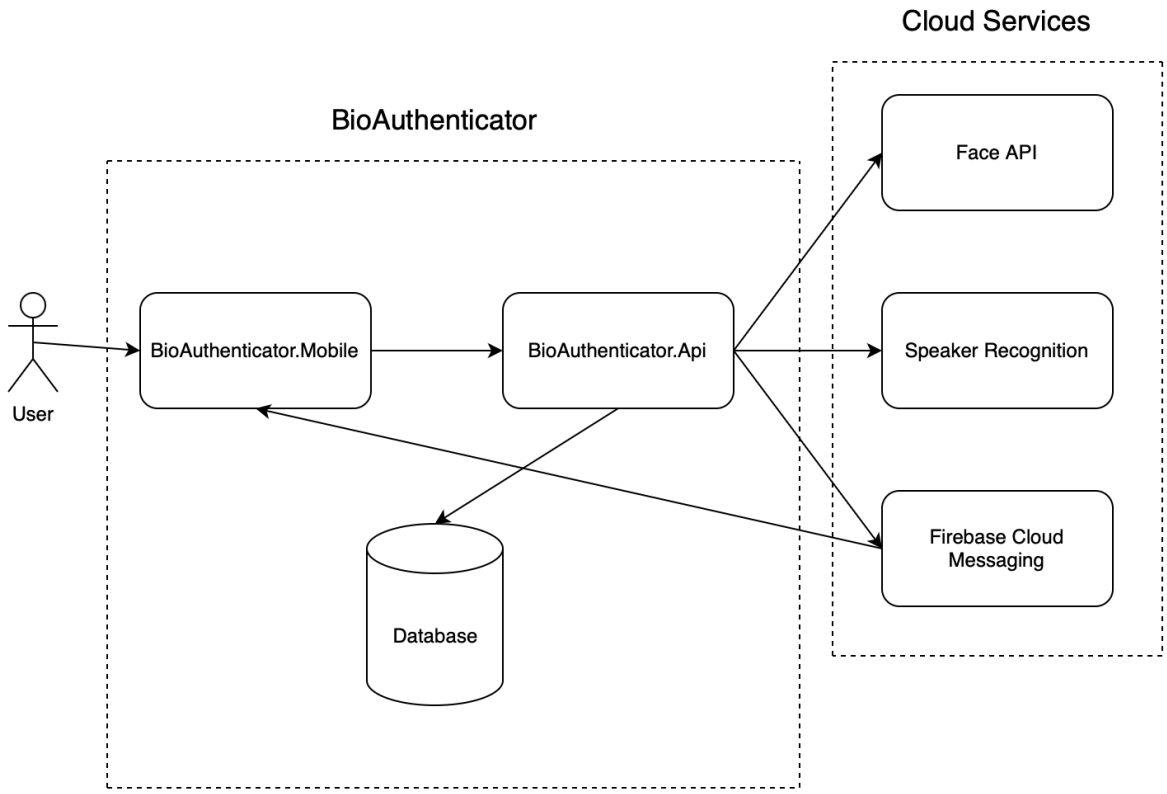

Рисунок 3.2 – Схема основних компонентів застосунку та взаємодії між ними

Таким чином, за необхідності, застосунок може надіслати користувачу запит на підтвердження його особистості для аутентифікації до того, чи іншого ресурсу.

3.2 Опис технічної реалізації застосунку

Для розробки API частини застосунку було вирішено використовувати Domain-Driven Design підхід. Domain-Driven Design – це підхід до розробки програмного забезпечення, який зосереджується на програмуванні моделі домену, яка має глибоке розуміння процесів і правил домену [13]. Хоча цей підхід і більш орієнтований на великі системи, зі складними взаємовідносинами між різними моделями, він все ж є дуже зручним і для невеликих застосунків, адже вносить в проект багато зручності.

При використанні DDD підходу дуже зручно розділяли застосунок на

наступні 3 проекти:

- Api;
- Core;
- Infrastructure/BusinessLogic.

Проект Api зазвичай містить лише код пов’язаний з налаштуванням роботи власне фреймворку ASP.NET Core. Також цей проект не повинен містити жодної бізнес логіки, а лише мати залежність на інші проекти для цих потреб.

Core проект містить лише моделі, що несуть цінність в описі того, чи іншого домену. В даному випадку доменом можна вважати аутентифікацію користувача в тому, чи іншому сервісі, тому доменна модель включає в себе такі класи:

- AppInstallation;
- Organization;
- OrganizationUser;
- OrganizationUserEnrollment;
- EnrollmentCode;
- UserAuthMethod;
- AuthMethodEnrollmentResponse;
- AuthenticationHistory.

Більшість з цих класів мають відповідну таблицю, що вміщує дані про ту, чи іншу модель в базі даних, інші ж просто описують деякий набір даних для зручності.

AppInstallation клас представляє собою модель з інформацією про встановлення мобільного застосунку. При використанні Firebase Cloud Messaging сервісу надає можливість отримати ідентифікатор встановлення застосунку. Таким чином можна розрізняти той, чи інший пристрій із встановленим застосунком. Саме цей ідентифікатор використовується для відправки push-повідомлень на мобільний пристрій. AppInstallation якраз описує модель з таким ідентифікатором, та різноманітну інформацію, що

пов’язана з тим, чи іншим девайсом, таку як ідентифікатори профілів обличчя та голосу, що використовуються для роботи з хмарними сервісами.

Organization клас описує інформацію про організацію. Зазвичай при використанні того, чи іншого сервісу користувач не вдається в деталі того, як все працює, та просто сприймає різноманітні сервіси як ту, чи іншу організацію, як наприклад Facebook, чи Instagram, тому модель, що зберігає інформацію про такі сервіси було вирішено назвати організацією. Ця модель наразі зберігає назву та фото організації, проте з легкістю може бути розширена додатковою інформацією у майбутньому.

OrganizationUser клас описує модель користувача з точки зору організації. Кожна організація має свій список користувачів, проте користувач застосунку може мати декілька пов’язаних з ним організацій.

OrganizationUserEnrollment клас описує запис в базі даних, що зберігає інформацію про налаштування зв’язку між користувачем певної організації та встановленим мобільним застосунок. Такими даними є ідентифікатор користувача всередині організації, код налаштування, дата створення такого коду та поле, що відображає стан налаштування, чи воно завершилось, чи ні.

EnrollmentCode – клас, що описує код налаштування, який буде використано для генерації QR коду. Ця модель вміщує в собі власне код, інформацію про організацію та користувача. Інформація використовується мобільним застосунком при скануванні QR коду для налаштування зв’язку між користувачем організації та мобільного застосунку.

UserAuthMethod описує модель для зручності та поєднує дані з декількох таблиць бази даних. Ця модель вміщує інформацію про метод аутентифікації, та статус налаштування того, чи іншого методу для конкретного девайсу.

AuthMethodEnrollmentResponse – клас, що описує проміжний результат налаштування методу аутентифікації для девайсу. Ця модель вміщує в собі статус налаштування, що представлений перерахуванням AuthMethodEnrollmentResult, кількість повторень, необхідних для

завершення налаштування, та повідомлення, що може бути відправлене при невдалій спробі налаштування. Статус налаштування може бути завершеним, в процесі та невдалим.

AuthenticationHistory описує запис, необхідний для аудиту спроб аутентифікації. Модель вміщує в собі інформацію про користувача тієї, чи іншої організації, дату спроби та стан аутентифікації. Стани аутентифікації описуються перерахуванням AuthenticationAttemptState та можуть бути такими: в процесі, підтверджений, відхилений, невдалий, з вичерпаним часом підтвердження, відхилений на етапі збору біометричних даних. Така інформація може бути корисною при аналізі історії аутентифікацій користувача, оскільки з легкістю можна виявити спроби неправомірного доступу.

Під час встановлення застосунку перш за все викликаються відповідні ендпоінти, що додають базову інформацію про користувача та девайс до бази даних. На цьому етапі користувач вже має можливість переглянути список пов’язаних організацій, додати нову, відсканувавши QR код, та налаштувати методи аутентифікації. Налаштування методів аутентифікації можливе навіть без прив’язки організацій, тому розглянемо процес налаштування (лістинг 3.1).

Лістинг 3.1 – Налаштування методів аутентифікації

```
public async Task<AuthMethodEnrollmentResponse>
EnrollAsync(string appInstallationId, AuthMethod method,
        IFormFile file, string? passphrase)
    {
        var appInstallation =
            await
_context.AppInstallations.FirstOrDefaultAsync(x =>
x.InstallationId == appInstallationId);
        if (appInstallation is null)
        {
            throw new
BioAuthenticatorEntityNotFoundException(nameof(AppInstallation),
                nameof(AppInstallation.InstallationId),
appInstallationId);
```

```
        }
        return method switch
        {
            AuthMethod.Face => await
EnrollNewFaceAsync(appInstallation, file),
            AuthMethod.Voice => await
EnrollNewVoiceAsync(appInstallation, file, passphrase),
            _ => throw new BioAuthenticatorException($"Invalid
auth method: {method}")
        };
    }
```

Даний метод використовується для налаштування методів аутентифікації. Перш за все проводиться перевірка на наявність інформації про девайс, що налаштовується, у базі даних. Запис у базі даних про девайс необхідна, адже саме вона буде вміщувати в собі дані про той, чи інший метод аутентифікації після його налаштування, тому її відсутність можна вважати за несправність. В нормальній ситуації запис повинен існувати і в такому випадку буде викликано функцію налаштування іншого сервісу, відповідно до обраного методу.

Для налаштування методу з використанням обличчя до API необхідно передати зображення та відповідний метод. Наступним кроком буде проведення необхідних перевірок та виклик хмарного сервісу Face API (лістинг 3.2).

Лістинг 3.2 – Налаштування методу аутентифікації з використанням розпізнавання обличчя.

```
private async Task<AuthMethodEnrollmentResponse>
EnrollNewFaceAsync(AppInstallation appInstallation, IFormFile
faceImage)
    {
        if (appInstallation.FaceProfileId is not null)
        {
            throw new
BioAuthenticationInvalidEnrollmentException(
                "Provided app installation already has face
enrolled.");
        }
        var personGuid = await
```

```
_faceService.CreatePersonAsync(appInstallation.InstallationId);
        var result = await
_faceService.AssignFaceToPersonAsync(personGuid, faceImage);
        if (result.Result ==
AuthMethodEnrollmentResult.EnrollmentFailed)
        {
            await _faceService.DeletePersonAsync(personGuid);
            return result;
        }
        appInstallation.FaceProfileId = personGuid;
        await _context.SaveChangesAsync();
        return result;
    }
```

Перш за все, при налаштуванні методу аутентифікації з використанням розпізнавання обличчя, проводиться перевірка, чи обраний метод ще не є налаштованим для користувача. Користувач не повинен мати можливість налаштовувати метод декілька разів, без попереднього видалення старого налаштування. Це зумовлено особливостями роботи з Face API. Сервіс Face API має певну ієрархію. Перш за все людина у цьому сервісі представлена сутністю Person, яка може мати декілька пов’язаних облич. Кожна сутність Person є частиною сутності PersonGroup, яку можна вважати контейнером для групи користувачів. Так розподілення має сенс через те, що сервіс використовує штучний інтелект та проводить тренування моделі саме на таких групах користувачів, що значно пришвидшує процес. Порівняння обличь також проводиться саме серед користувачів певної групи. Таким чином застосунок використовує одну групу, щоб можна було проводити розпізнавання обличчя серед усіх користувачів, а при налаштуванні методу, API створює для кожного користувача окрему сутність Person. Якщо ж користувачу необхідно заново налаштувати метод, API потрібно видалити існуючий запис Person та лише після цього проводити нове налаштування.

Налаштування методу аутентифікації за голосом має схожий принцип, проте має відмінності (лістинг 3.3).

Лістинг 3.3 – Налаштування методу аутентифікації з використанням розпізнавання голосу.

```
private async Task<AuthMethodEnrollmentResponse>
EnrollNewVoiceAsync(AppInstallation appInstallation, IFormFile
voiceRecord, string? passphrase)
    {
        if (appInstallation.SpeechProfileId is null)
        {
            var profileId = await
_speechService.CreateProfileAsync();
            appInstallation.SpeechProfileId = profileId;
            await _context.SaveChangesAsync();
        }
        if (passphrase is null)
        {
            throw new BioAuthenticatorException("Passphrase is
required for voice enrollment");
        }
        return await
_speechService.EnrollProfileAsync(appInstallation.SpeechProfileI
d, voiceRecord, passphrase);
    }
```

Сутність, що використовується для представлення користувача у Speaker Recognition сервісі називається Profile. На відміну від методу з розпізнаванням обличчя в даному випадку було вирішено не вважати вже існуючий профіль пов’язаний з користувачем за невалідний випадок. Це зумовлено тим, що для налаштування профілю у Speaker Recognition сервісі необхідно виконати декілька турів.

Коли методі аутентифікації налаштовані, постає питання прив’язки користувача певної організації до відповідного їй девайсу. Тож розглянемо метод генерації QR коду для прив’язки пристрою (лістинг 3.4).

Лістинг 3.4 – Метод генерації QR коду для прив’язки пристрою

```
public async Task<byte[]> GetEnrollmentQrCodeAsync(Guid
organizationId, Guid organizationUserId)
    {
        var userEnrollment = new OrganizationUserEnrollment
        {
```

```
            IsEnrollmentCompleted = false,
            OrganizationUserId = organizationUserId,
            EnrollmentCode = Guid.NewGuid(),
            EnrollmentCodeGeneratedDate = DateTime.UtcNow
        };
        await
_context.OrganizationUserEnrollments.AddAsync(userEnrollment);
        await _context.SaveChangesAsync();
        var code = new
EnrollmentCode(organizationUser.Organization.Name,
organizationId, organizationUserId,
            userEnrollment.EnrollmentCode);
        var qrCodeData =
_qrGenerator.CreateQrCode(JsonSerializer.Serialize(code),
QRCodeGenerator.ECCLevel.Q);

        return new PngByteQRCode(qrCodeData).GetGraphic(20);
    }
```

Для прив’язки користувача організації до девайсу генерується QR код який містить необхідну інформацію. Під час генерації коду також створюються записи в базі даних з станом прив’язки та датою генерації коду, що дає змогу встановлювати проміжок часу, за який користувач повинен використати код, та вважати його недійсним, якщо відведений час пройшов.

Надалі, коли сервісу, що використовує даний застосунок знадобиться підтвердження особистості користувача, йому треба буде викликати відповідний ендпоінт (лістинг 3.5), що відповідає за надсилання запиту на авторизацію до відповідного девайсу.

Лістинг 3.5 – Запит на підтвердження аутентифікації

```
public async Task<bool> RequireAuthenticationApprovalAsync(Guid
organizationId, Guid organizationUserId)
    {
        var user = await _context.OrganizationUsers
            .Include(x => x.AppInstallation)
            .Include(x => x.Organization)
            .FirstOrDefaultAsync(x =>
            x.OrganizationId == organizationId && x.Id ==
organizationUserId);
        // Some validation code was skipped here
        await
MarkAllActiveAuthenticationAttemptsCanceledAsync(organizationUse
```

```
rId);
        await SendAuthenticationRequestNotification(user);
        var historyRecord = new AuthenticationHistory
        {
            AuthenticationAttemptDate = DateTime.UtcNow,
            AuthenticationAttemptState =
AuthenticationAttemptState.InProgress,
            OrganizationUserId = organizationUserId
        };
        await
_context.AuthenticationHistories.AddAsync(historyRecord);
        await _context.SaveChangesAsync();
        return await
WaitForUserAuthApprovalAsync(historyRecord);
    }
```

В даному випадку з бази дістається необхідна інформація про користувача, яка також містить ідентифікатор, що використовується для надсилання push-повідомлень на відповідний девайс. На девайс користувача за допомогою сервісу Firebase Cloud Messaging надсилається повідомлення, що сигналізує про необхідність підтвердити свою особистість за допомогою біометричних даних.

Кожен такий запит на підтвердження особистості у цілях безпеки є валідним лише певний проміжок часу, за замовчуванням це 40 секунд, проте це значення може бути змінено в конфігураційних файлах проекту. По вичерпаною часу такий запит стає невалідним. Також, якщо сервіс надіслав новий запит в той час, коли попередні запити залишались активними, усі попередні запити автоматично становляться відхиленими.

Після надсилання запиту API додає запис з відповідним статусом до AuthenticationHistories таблиці та очкує на зміну цього статусу протягом 40 секунд, щоб надати відповідь сервісу, що запросив підтвердження особи користувача. Далі в хід входить інший метод, що викликається з мобільного застосунку, та передає до API біометричні дані користувача. Отримавши дані, API викликає хмарні сервіси, відповідно до обраного методу аутентифікації та отримує результат порівняння обличчя, чи голосу, на основі якого оновлює запис в AuthenticationHistories таблиці, відображаючи актуальний статус.

4 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ПРОВЕДЕННЯ БІОМЕТРИЧНОЇ АУТЕНТИФІКАЦІЇ

4.1 Використання мобільного застосунку

При вході в застосунок, можна побачити 3 основні функціональні сторінки (рисунок 4.1):

- Organizations;
- Camera;
- Methods.

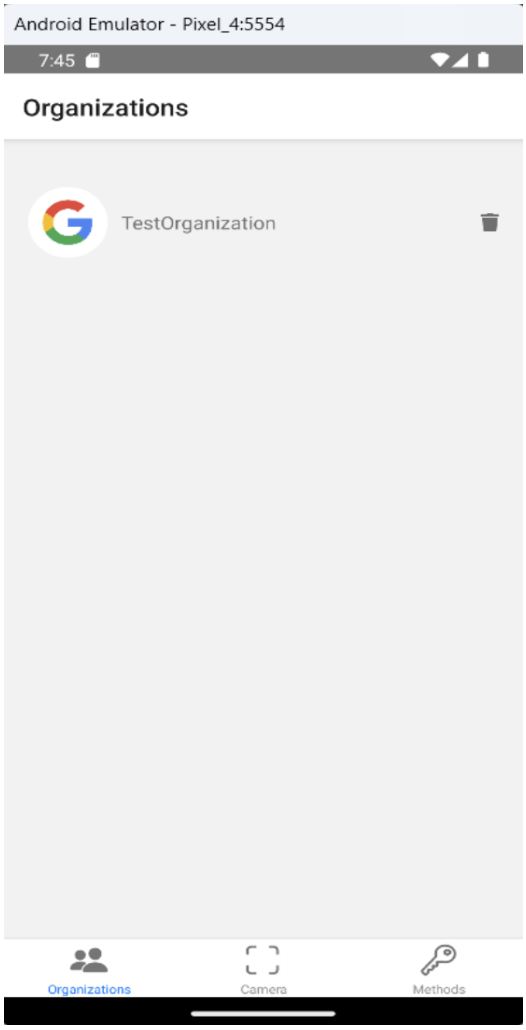

Рисунок 4.1 – Головна сторінка застосунку

На сторінці Organizations можна побачити список організацій, що пов’язані з поточним пристроєм. Для кожної організації відображається її назва, іконка та кнопка, що надає можливість видалити зв’язок пристрою з тією, чи іншою організацією.

Для додавання асоціації з новою організацією, необхідно перейти на сторінку Camera. Ця сторінка відкриває камеру на весь екран, за допомогою якої потрібно відсканувати QR код, наданий відповідною організацією.

На сторінці Methods можна побачити всі можливі методи аутентифікації, що наразі існують в застосунку (рисунок 4.2).

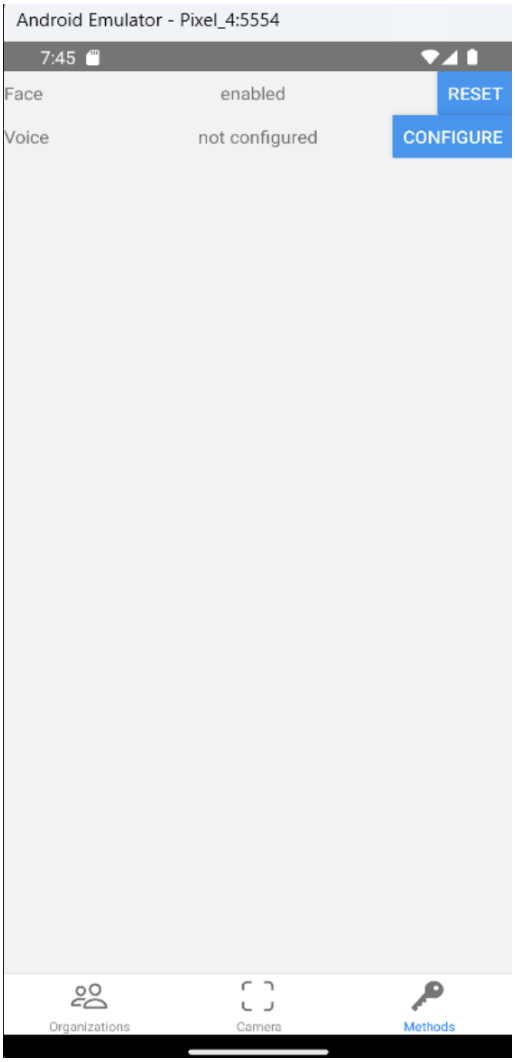

Рисунок 4.2 – Сторінка з методами аутентифікації

Ця сторінка надає можливість налаштувати, або скинути налаштування для того, чи іншого методу аутентифікації. При налаштуванні методу з використанням розпізнавання обличчя, застосунок відкриває камеру телефону, де користувачу потрібно зробити фото свого обличчя. Після відправки та обробки фотографії застосунком, вже можна починати використовувати цей метод.

Для налаштування аутентифікації за голосом, повинен розпочатись запис звуку та відобразитись фраза, яку потрібно вимовити декілька разів для завершення налаштування. Проте нажаль на сьогоднішній день компанія Microsoft закрила доступ до Speaker Recognition сервісу, та надає його лише для користувачів, що є партнерами компанії. Весь код та інфраструктура для роботи з даним сервісом були розроблені, але нажаль ця функція застосунку наразі недоступна.

При необхідності підтвердити свою особу, на телефон приходить повідомлення (рисунок 4.3).

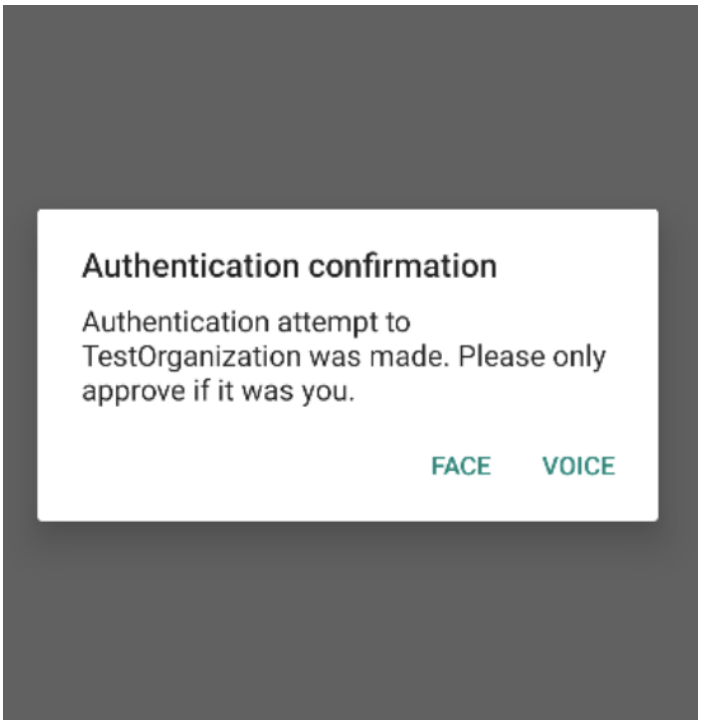

Рисунок 4.3 – Повідомлення про необхідність підтвердження аутентифікації

Якщо користувач впевнений, що цей запит був надісланий після його спроби пройти аутентифікацію у вказанову в повідомленні сервісі, він може підтвердити цю спробу за допомогою налаштованих методів аутентифікації. Якщо ж запит був надісланий без відома користувача, йому слід переглянути свій основний метод захисту.

4.2 Використання застосунку сторонніми сервісами у якості двохфакторної аутентифікації

Для використання застосунку сторонніми сервісами розроблено окремий контролер з ендпоінтами специфічними для такого випадку (рисунок 4.4).

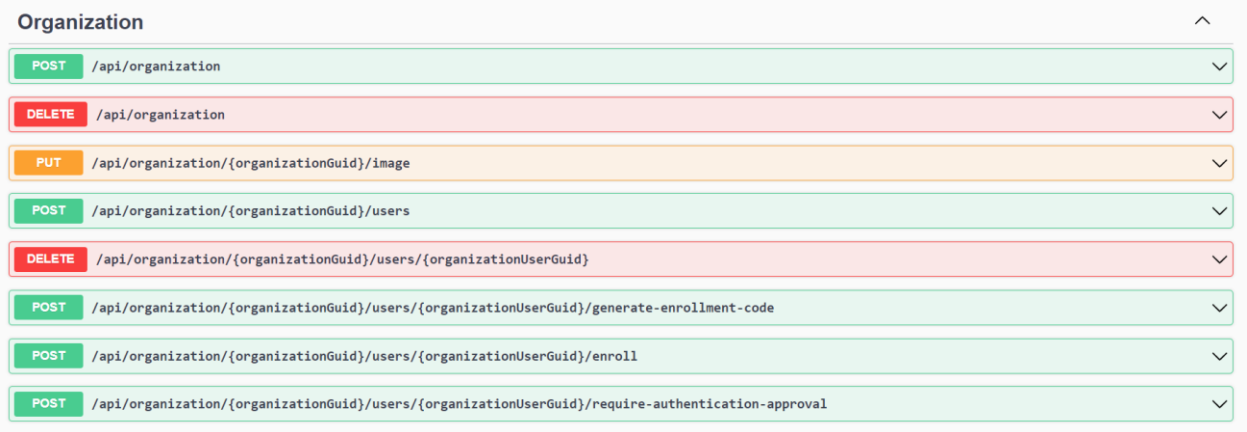

Рисунок 4.4 – Ендпоінти Organization контролеру

Основною одиницею організаці є користувачі даної організації. Для створення користувачів в середині організації використовується POST метод /api/organization/{organizationId}/users ендпоінту.

Наступним кроком для забезпечення додаткового захисту при аутентифікації користувача потрібно сгенерувати QR (рисунок 4.5) код для пов’язування користувача з його пристроєм. API надає зображення у вигляді base64 строки, перетворивши її в зображення організація повинна передати

його користувачу для подальших дій.

{
'OrganizationName': 'TestOrganization',
'OrganizationId': 'a7a56ad1-b91f-4c44-90a0-c58671257f4c',
'OrganizationUserId': 'b722b9bd-31d9-44e2-acb0-d1a60ef17e77',
'Code': '9cdc5e3f-b3e0-49c9-b2c2-2b6f64ee0a11'
}

Рисунок 4.5 – Сгенерований код та його контент

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було наведено обґрунтування актуальності потреби покращення аутентифікації людини, проведено аналіз парольної аутентифікації користувача і біометричних даних, як методу аутентифікації. Зроблено аналіз мультифакторної аутентифікації і виконано обґрунтування переваг використання мобільного пристрою для проведення процесу аутентифікації.

У роботі наведено огляд хмарних біометричних сервісів, сучасних середовищ розробки та технологій для створення застосунків для мобільних пристроїв, також наведено обґрунтування обраних технологій для їх розробки.

Виконано розробку архітектури програмного продукту і надано опис його технічної реалізації. Зроблено опис інтерфейсу користувача мобільного застосунку, показано процедуру використання застосунку сторонніми сервісами у якості двохфакторної аутентифікації.

Розроблений мобільний застосунок вміщує в собі такі популярні методи біометричної аутентифікації як розпізнавання обличчя та голосу. Для більшої ефективності, прийнято рішення використати сервіси, що надаються хмарними провайдерами, та мають відповідний функціонал для обробки біометричних даних. Використання мобільного застосунку для цілей двохфакторної аутентифікації з використанням біометричних даних виявилось дуже зручним та швидким, тобто можна зробити висновок, що такі методи є дійсно перспективними.

За результатами досліджень кваліфікаційної роботи опубліковано тези доповіді на десятій міжнародній науково-технічній конференції «Проблеми інформатизації» 24-25 листопада 2022 року [1].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондар О. Р. Технології покращення аутентифікації людини [Текст] / Бовчалюк С. Я., Бондар О. Р. // ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ. Тези доповідей десятої міжнародної науково-технічної конференції (24 – 25 листопада 2022 року), Том 1:секція 3. – 2022. – С. 123

2. Шиллер Р., Акерлоф Дж. Фішинг. Хто і як маніпулює вашим вибором [Текст] / Наш Формат, 2017. – 278 с.

3. Adding Salt to Hashing: A Better Way to Store Passwords [Текст] / Auth0 blog – Режим доступу до ресурсу: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/

4. Are Your Passwords in the Green? [Текст] / Hive Systems – Режим доступу до ресурсу: https://www.hivesystems.io/blog/are-your-passwords-in-the-green

5. Fingerprint Recognition [Текст] / Biometric Solutions – Режим доступу до ресурсу: https://www.biometric-solutions.com/fingerprint-recognition.html

6. Iris Scanner: How It Works | Eye Scan Technology Overview [Текст] / RecFaces – Режим доступу до ресурсу: https://recfaces.com/articles/iris-scanner

7. Just 20 Hours, $18, and 11 Million Passwords Cracked [Текст] / Hackernoon – Режим доступу до ресурсу: https://hackernoon.com/20-hours-18-and-11-million-passwords-cracked-c4513f61fdb1

8. MD5 Algorithm | Know Working And Uses Of MD5 Algorithm [Текст] / Educba – Режим доступу до ресурсу: https://www.educba.com/md5-alogrithm/

9. Minimal APIs quick references [Текст] / Microsoft – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis?view=aspnetcore-7.0

10. Multi-factor authentication [Текст] / Wikipedia – Режим доступу до

ресурсу: https://en.wikipedia.org/wiki/Multi-factor_authentication

11. Phishing attack statistic 2022 [Текст] / Cybertalk – Режим доступу до ресурсу: https://www.cybertalk.org/2022/03/30/top-15-phishing-attack-statistics-and-they-might-scare-you

12. Speaker Verification: Text-Dependent vs. Text-Independent - Microsoft Research [Текст] / Microsoft – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/project/speaker-verification-text-dependent-vs-text-independent/

13. The Concept of Domain-Driven Design Explained | by Sara Miteva | Microtica | Medium [Текст] / Medium – Режим доступу до ресурсу: https://medium.com/microtica/the-concept-of-domain-driven-design-explained-3184c0fd7c3f

14. The World's First Computer Password? It Was Useless Too [Текст] / Wired – Режим доступу до ресурсу: https://www.wired.com/2012/01/computer-password/

15. Top mobile frameworks in 2022 [Текст] / Clarion Technologies – Режим доступу до ресурсу: https://www.clariontech.com/blog/top-mobile-app-development-frameworks-in-2019

16. Troy Hunt: Passwords Evolved: Authentication Guidance for the Modern Era [Текст] / Troyhunt – Режим доступу до ресурсу: https://www.troyhunt.com/passwords-evolved-authentication-guidance-for-the-modern-era

17. What is a Spam Filter & Spam Filtering? [Текст] / Fortinet – Режим доступу до ресурсу: https://www.fortinet.com/resources/cyberglossary/spam-filters

18. What is Cloud Computing? Pros and Cons of Different Types of Services [Текст] / Investopedia – Режим доступу до ресурсу: https://www.investopedia.com/terms/c/cloud-computing.asp

19. What is SHA? What is SHA used for? | Encryption Consulting [Текст] / Encryption Consulting – Режим доступу до ресурсу:

https://www.encryptionconsulting.com/education-center/what-is-sha/

20. What is the Azure Face service? - Azure Cognitive Services | Microsoft Learn [Текст] / Microsoft – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-identity

21. What is Three-Tier Architecture [Текст] / IBM – Режим доступу до ресурсу: https://www.ibm.com/cloud/learn/three-tier-architecture