Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу, комп'ютерного моніторингу
та інженерної графіки

ІНФОРМАЦІЙНА СИСТЕМА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ БІОМЕТРИЧНИХ ДАНИХ

Пояснювальна записка до магістерської кваліфікаційної роботи

Виконала: студентка 2 курсу, групи ІСТ-18м
спеціальності 126 –
«Інформаційні системи та технології»
Волошина В.А.
Керівник: к.т.н., доц. Жуков С.О.
Рецензент: професор Васюра А.С.

Вінниця ВНТУ – 2019 року

ЗМІСТ

ВСТУП

Актуальність теми. Вирішення питання захисту інформації і комп'ютерних системах необхідно для того, щоб нормально функціонуюча інформаційна система була ізольована від несанкціонованих дій і доступу сторонніх осіб або програм до конфіденційних комп'ютерних даних [1].

Для того щоб отримати можливість працювати з програмним продуктом, необхідно пройти етап ідентифікації з подальшою авторизацією.

Потреба у дослідженні методів та засобів захисту користувача зумовлена тим, що ідентифікація користувачів є невід'ємним та важливим елементом і основою ефективності будь-якої системи управління доступом до інформаційних ресурсів комп'ютерних систем.

Звичайний захист паролем має значний ряд недоліків. У разі порушення конфіденційності пароля, відразу порушується захист всієї інформації, до якої він має доступ. Альтернативою заміною парольному захисту може бути ідентифікацію на основі біометричних даних. Біометричні технології ідентифікації користувачів мають переваги перед традиційними і з кожним роком все більш поширеними у застосуванні в комп'ютерних системах [2].

Особливість ідентифікації за біометричними характеристиками базується на їх винятковості. Мала ймовірність, що знайдуться хоча б дві людини з однаковими ознаками [3]. Ідентифікація за відбитком пальця та геометрією обличчя є найбільш поширеною та актуальною, тому існує велика кількість розроблених продуктів. Кожен з яких має свої переваги і недоліки, проте є достатньо дорогими. Отже, розроблення нового продукту для захисту інформації шляхом ідентифікації за відбитками пальців та геометрією обличчя через смартфон є актуальним.

Зв'язок з науковими програмами, планами, темами. Проведене у роботі дослідження здійснювалося в межах секційного наукового напрямку кафедри системного аналізу, комп'ютерного моніторингу та інженерної графіки Вінницького національного технічного університету.

Об’єктом дослідження є процес захисту інформації шляхом обмеженням доступу в комп’ютерних системах.

Предметом дослідження — метод ідентифікації користувача за біометричними даними через смартфон із подальшим розмежуванням доступу та авторизацією.

Метою дослідження є підвищення ефективності захисту конфіденційних даних від несанкціонованих дій шляхом розмежування доступу за допомогою ідентифікації користувачів за біометричними характеристиками.

Для досягнення даної мети було поставлено низку таких **задач**:

– визначити зміст та значення проблеми захисту інформації у процесі ідентифікації користувачів;

– здійснити класифікацію методів ідентифікації;

– проаналізувати сучасний стан наукової проблеми та практичний досвід її реалізації;

– вивчити теоретичні засади методів біометричної ідентифікації;

– вивчити теоретичні засади методів розмежування доступу;

– обґрунтувати підхід до ідентифікації користувача за біометричними даними через смартфон як оптимальний;

– обґрунтувати вибір мови програмування та середовища розробки;

– побудувати базу даних користувачів;

– описати принцип роботи системи;

– розробити інформаційну систему ідентифікації користувача за біометричними даними через смартфон;

– розробити інструкцію користувача.

Наукова новизна результатів дослідження полягає у вдосконаленні методу ідентифікації користувачів з подальшою авторизацією та розмежуванням доступу шляхом використання біометричних даних, а саме відбитків пальців та геометрії обличчя, який, на відміну від уже створених підходів, підвищує точність

вимірювань, розширює функціональні можливості засобів автентифікації, здешевлює їх та підвищує ефективність системи захисту даних.

Практичне значення одержаних результатів дослідження полягає у розробленні інформаційної системи ідентифікації користувача за біометричними даними через смартфон із подальшою авторизацією та розмежуванням доступу.

Усі запропоновані в роботі шляхи до вирішення поставлених задач розроблено автором самостійно.

Апробація. Результати роботи було представлено на щорічній конференції Міжнародної науково-практичної конференції Одеської національної академії харчових технологій (м. Одеса, 2019).

Публікації. Результати дослідження було опубліковано у збірнику доповідей XII Міжнародної науково-практичної конференції Одеської національної академії харчових технологій [4].

1 СУЧАСНИЙ СТАН ПРОБЛЕМИ ТА АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

Даний розділ призначений для визначення та детального аналізу проблем захисту інформації під час ідентифікації користувачів. Ідентифікація й автентифікація – це основа програмно-технічних засобів безпеки, оскільки інші сервіси розраховані на обслуговування іменованих суб'єктів.

1.1 Ідентифікація в захищених системах

Останнім часом у зв’язку зі збільшенням загроз для цифрової інформації все більше уваги приділяється задачам вдосконалення існуючих та розробці нових способів захисту інформаційних систем від небажаного доступу з боку неавторизованих користувачів. Одним з напрямків для досягнення цієї мети є системи контролю та управління доступом, в яких розмежування доступу проводиться шляхом ідентифікації користувачів, тобто процесом розпізнавання параметрів, що точно визначають користувача якому вони належать [5]. Ідентифікація дозволяє користувачеві назвати себе (повідомити своє ім'я).

Автентифікацією називається процедура перевірки автентичності входу в систему об'єкта, що пред'явив свій ідентифікатор [6]. Залежно від ступеня довірчих відносин, структури, особливостей мережі і віддаленістю об'єкта перевірка може бути односторонньої або взаємної. У більшості випадків вона полягає в процедурі обміну між входять в систему об'єктом і ресурсом, відповідальним за прийняття рішення («так» чи «ні»). Перевірка здійснюється за допомогою криптографічних перетворень. Вони необхідні для того, щоб упевнитися в тому, що суб'єкт є тим, за кого себе видає, а також для захисту трафіку обміну суб'єкта і системи від зловмисників.

За допомогою автентифікації друга сторона впевнюється, що суб'єкт є тим, за кого він себе видає. Автентифікацію можна назвати перевіркою дійсності.

У будь-якій системі автентифікації зазвичай можна виділити кілька елементів (рис. 1.1).

Рисунок 1.1 – Складові системи автентифікації

У відкритому мережевому середовищі не існує довіреного шляху між сторонами ідентифікації/автентифікації – це означає, що в загальному випадку дані, передані суб'єктом, можуть не збігатися з даними, отриманими й використаними для перевірки дійсності [7]. Необхідно забезпечити захист від пасивного й активного прослуховування мережі, тобто від перехоплення, зміни й/або відтворення даних. Передача паролів у відкритому вигляді незадовільна; не допомагає й шифрування паролів, тому що воно не захищає від відтворення. Потрібні більш складні протоколи автентифікації.

Надійна ідентифікація й автентифікація ускладнена не тільки через мережеві загрози, але й з цілого ряду причин. По-перше, майже всі автентифікаційні сутності можна дізнатись, вкрасти чи підробити. По-друге, є протиріччя між надійністю автентифікації, з одного боку, і зручностями користувача й системного адміністратора з іншого. Задля безпеки необхідно з періодичністю просити користувача повторно вводити інформацію, що необхідна для автентифікації (адже

її могла замінити інша людина), а це не тільки не зручно, але й існує ймовірність того, що введені данні можуть бути підглянуті. По-третє, надійний захист збільшує витрати на нього [8].

Сьогоденні засоби ідентифікації/автентифікації мають підтримувати концепцію єдиного входу в мережу. Єдиний вхід у мережу – це вимога зручності. Якщо в корпоративній мережі безліч інформаційних сервісів, що допускають незалежний обіг, то багаторазова ідентифікація/автентифікація стає досить обтяжною [9]. Не можна ствердити, що єдиний вхід у мережу став нормою, бо домінантні рішення поки не сформувалися, тому необхідно шукати компроміс між надійністю, доступністю у цінових рішеннях та зручністю у використанні й адмініструванні засобів ідентифікації й автентифікації.

Однак, сервіс ідентифікації/автентифікації може стати об'єктом атак на доступність. Якщо система налаштована так, що після певного числа невдалих спроб пристрій уведення ідентифікаційної інформації блокується, то зловмисник може припинити роботу легального користувача декількома натисканнями клавіш [10,11].

Коректність рішення розпізнавання і перевірки дійсності впливає на те, чи буде дозволений конкретному користувачеві доступ до ресурсів системи, тобто чи буде він авторизований.

Авторизація являє собою процедуру надання суб'єкту певних прав доступу до ресурсів системи після успішного проходження ним процедури автентифікації. Для кожного суб'єкта в системі назначається набір прав, які він може використовувати при зверненні до її ресурсів.

Процес управління доступом користувачів до ресурсів системи називається адмініструванням (рис. 1.2) [12].

Рисунок 1.2 – Елементи процесу адміністрування

Логічне управління доступом важливе не тільки на рівні операційної системи, але і в рамках інших сервісів, що входять до складу сучасних додатків, а також, наскільки це можливо, на “стиках” між сервісами. Тут на перший план виходить існування єдиної політики безпеки організації, та кваліфіковане і узгоджене системне адміністрування.

Для великої кількості користувачів традиційні підсистеми управління доступом стають дуже складними для адміністрування. Кількість зав’язків в них пропорційно добутку кількості користувачів на кількість об’єктів.

1.2 Класифікація способів ідентифікації користувачів

Основу системи інформаційної безпеки утворюють засоби ідентифікації користувачів і управління їх доступом до корпоративних інформаційних ресурсів. Основні вимоги до системи ідентифікації і можуть бути вирішені нею завдання [13-15]:

– однозначність розпізнавання користувача по унікальним, властивим йому одному ознаками;

– неможливість розкрадання, втрати, підміни ідентифікаційних ознак та / або оволодіння ними обманним шляхом;

– запобігання можливому обміну ідентифікаторами і відповідними повноваженнями користувачів;

– реалізація принципу «неможливості відмови» користувача від транзакцій, здійснених із застосуванням ідентифікаторів, що відповідають перерахованим вище критеріям;

– ефективна інтеграція засобів ідентифікації та управління доступом в інформаційну інфраструктуру компанії зі збереженням безперервності поточних бізнес-процесів;

– зниження навантаження на користувачів, адміністраторів, фахівців із захисту інформації;

– мінімізація витрат, пов'язаних з використанням коштів ідентифікації та управління доступом;

– відмовостійкість і масштабованість формованої системи ідентифікації та управління доступом.

Процедури ідентифікації та автентифікація пов’язані між собою, оскільки спосіб перевірки вказує користувач що він має надати системі задля отримання доступу. Існує декілька способів ідентифікації користувачів [16]. Кожен має свої переваги і недоліки, тому деякі технології використовують в одних комп’ютерних системах, інші – в інших. Однак у багатьох випадках не існує конкретного рішення. А тому і розробникам програмного забезпечення, і користувачам необхідно самостійно приймати рішення, який спосіб реалізовувати у власних комп’ютерних системах. В цілому всі способи ідентифікації можна розділити на чотири великих групи: парольна, апаратна, біометрична та багатофакторна (рис. 1.3).

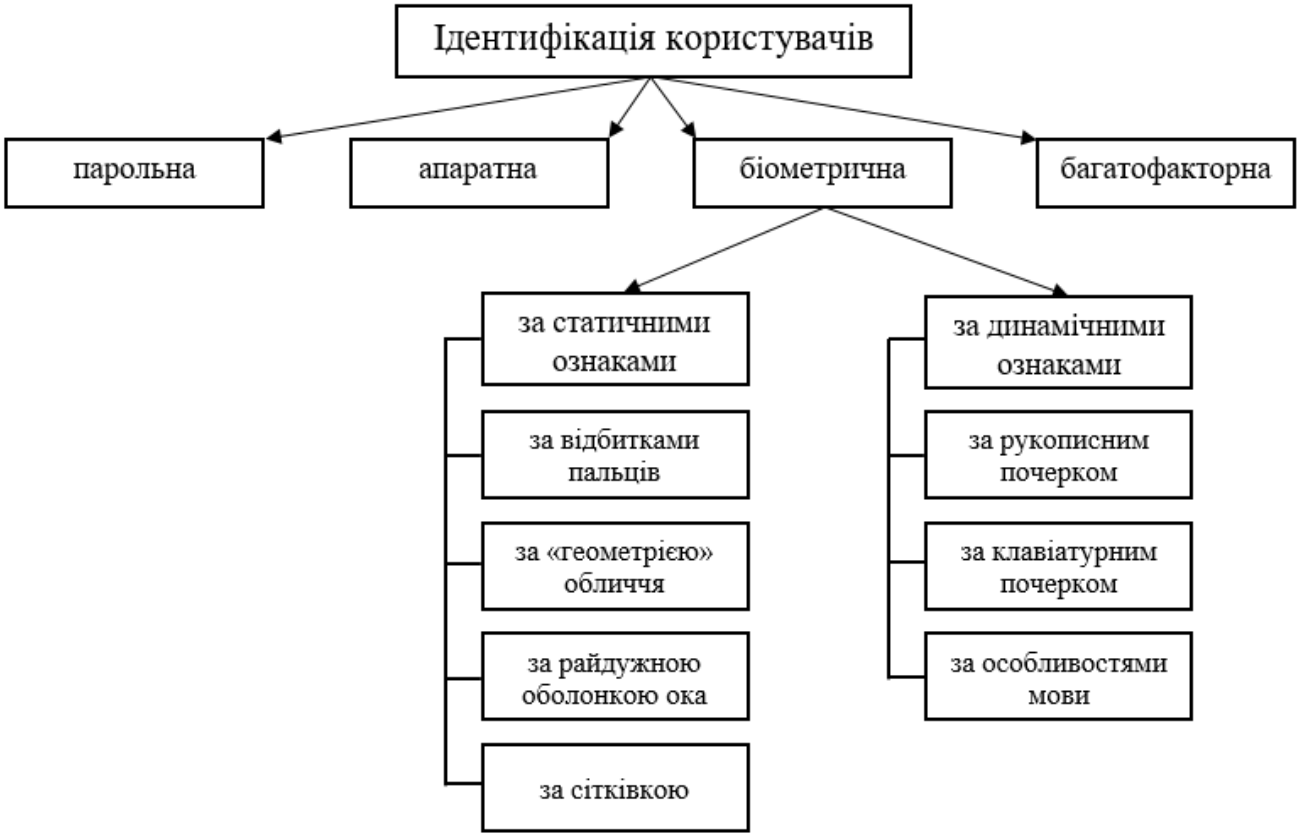


Рисунок 1.3 — Способи ідентифікації користувачів

Парольна ідентифікація. Головна перевага – простота й звичність. Паролі давно використовують в операційних системах та інших сервісах. При правильному використанні паролі можуть забезпечити достатній рівень безпеки для компанії. Але, по сукупності характеристик їх можна визнати найслабшим методом перевірки дійсності [17].

Парольна автентифікація має досить багато недоліків. На відміну від випадково сформованих криптографічних ключів, пароль користувача легко підібрати через досить недбале ставлення більшості до формування пароля. Досить часто можна зустріти випадки вибору користувачами досить передбачуваних паролів, до прикладу:

– пароль еквівалентний імені користувача (або імені, що записане в зворотному порядку, або легко з нього формується і т.п.). Такий пароль легко вгадати, особливо якщо знати вподобання даного користувача.

– пароль - це слово чи фраза. Такий пароль може бути підібраний за досить короткий час шляхом «словникової атаки» - перебір всіх слів відповідно до

словника, який містить всі слова і загальновживані фрази мови, що використовується;

– часто користувачі застосовують короткі паролі, який можна вгадати "методом грубої сили", за допомогою словника. Якщо файл паролів зашифрований, але доступний для читання, його можна завантажити на комп'ютер і спробувати підібрати пароль, застосувавши повний перебір (це можливо якщо відомий алгоритм шифрування).

– іноді паролі із самого початку не є таємницею, тому що мають стандартні значення, вказані у супровідних документах і рідко після встановлення системи проводиться їх заміна.

Також недоліком є те, що процес вводу паролю можна піддивитись, інколи для цього використовують спеціальні прилади.

Паролі досить часто повідомляють колегам, щоб ті могли, замінити на деякий час власника пароля. У теорії у схожих випадках доцільно залучити засоби керувань доступом, але нажаль це не використовують [18].

Проте, є заходи, які дозволяють значно підвищити надійність парольного захисту(рис 1.4.) .

Рисунок 1.4 – Заходи, що дозволяють покращити захист паролем

Перераховані заходи слід використовувати завжди, навіть якщо разом з паролями використовуються інші методи автентифікації [19].

Апаратна ідентифікація. Під апаратною ідентифікацією розуміють апаратно-програмні системи, засоби або пристрої введення ідентифікаційних ознак. До складу системи входять апаратні ідентифікатори: зчитувачі, контактні/безконтактні пристрої, адаптери, роз'єми системної плати та відповідне програмне забезпечення. Ідентифікатори вони можуть зберігати і обробляти ідентифікаційні ознаки, а також конфіденційні дані [20-22].

В цифрових системах ідентифікаційні ознаки представляються у вигляді коду, що зберігається в пам'яті ідентифікатора. Усі електронні системи ідентифікації/автентифікації за способом обміну даними між ідентифікатором і пристроєм введення-виведення можна поділити на дві групи (рис 1.5).

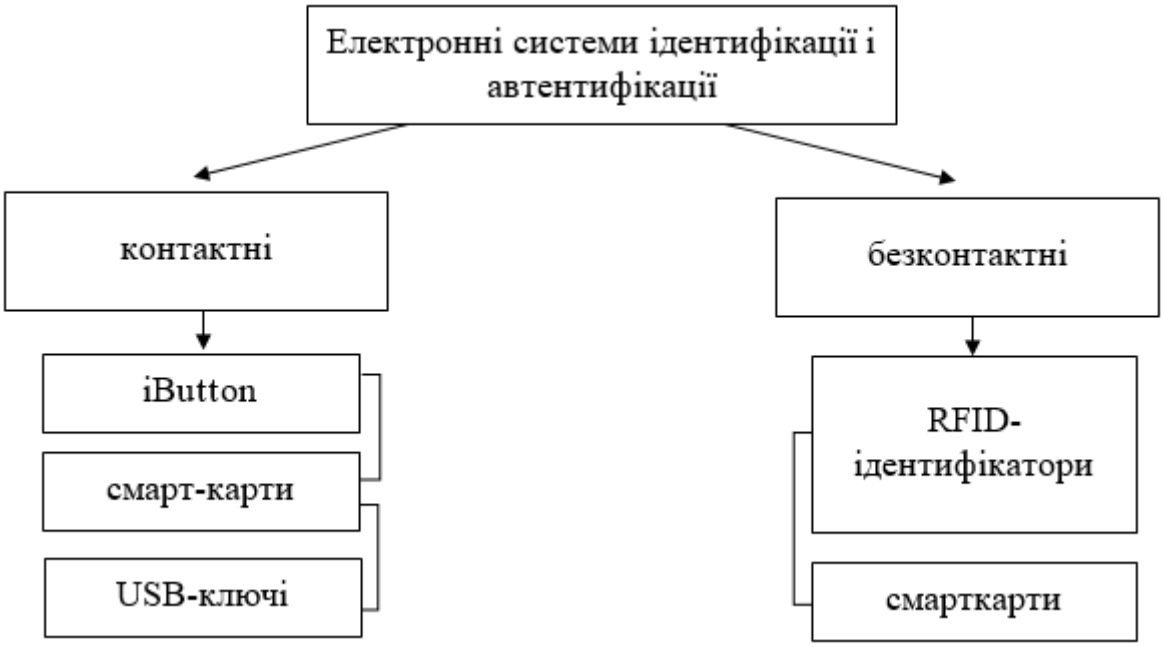


Рисунок 1.5 – Класифікація системи ідентифікації і автентифікації

Контактне зчитування – це пряма взаємодія ідентифікатора з пристроєм вводу-виводу. Безконтактний спосіб обміну не потребує прямої взаємодії ідентифікатора і пристроїв введення-виведення. Запис даних або зчитування відбувається при знаходженні ідентифікатора на певній відстані від пристрою введення-виведення [23].

У надійності системи найслабша ланка це ідентифікатор. Їх можна назвати надійними тоді, коли є захищеність від механічних пошкоджень, стійкості до змін температури, електромагнітних полів, захищеність від пилу, вологи. Ідентифікатор має бути стійким до атак, які спрямовані на розкриття чіпу, що зберігає конфіденційні дані.

Апаратна ідентифікація гарантує високу надійність – це одна із її найголовніших переваг. У пам’яті токенів зберігаються ключі, які не вдасться підібрати хакерам. Також у них розроблені механізми захисту. Вбудований в систему мікропроцесор дозволяє електронному ключу виконувати інші корисні функції.

Але не дивлячись на всі переваги головний недолік даного виду ідентифікації це доволі висока ціна. Незважаючи на те, що ціна електронних ключів та програмного забезпечення, що може працювати з ними, помітно знизилася для введення в експлуатацію системи ідентифікації все одно необхідна велика кількість коштів, адже кожного користувача, що зареєстрований у системі необхідне персональний токен. Також недоліком є те, що деякі ключі мають обмежений час використання, через малу зносостійкість. Також не варто забувати, що ключ можна легко загубити.

Біометрична ідентифікація. Біометрична ідентифікація — це спосіб ідентифікації особи по окремих специфічних біометричних ознаках (ідентифікаторах), властивих конкретній людині [24]. Сучасний рівень розвитку комп'ютерних технологій дозволяє використовувати подібні ознаки як основу для ідентифікації людини і ухвалення рішення про можливість або неможливість доступу до інформаційних комп’ютерних систем.

Всі біометричні системи працюють за схожою схемою. Спочатку система запам'ятовує зразок біометричної характеристики (це називається процесом запису). Під час запису деякі біометричні системи можуть попросити зробити декілька зразків для того, щоб скласти найточніше зображення біометричної характеристики. Потім одержана інформація обробляється і перетворюється в

математичний код. Крім того, система може попросити провести ще деякі дії для того, щоб «приписати» біометричний зразок до певної людини.

Ідентифікація по будь-якій біометричній системі складається з чотирьох стадій [25] (рис. 1.6).

Рисунок 1.6 – стадії біометричної ідентифікації

Сьогодні існує велика кількість біометричних методів. Основними методами, які використовують статичні біометричні характеристики людини, є ідентифікація по папілярних малюнку на пальцях, райдужній оболонці, геометрії особи, сітківці ока, малюнку вен руки, геометрії рук. Також існує сімейство методів, які використовують динамічні характеристики: ідентифікація по голосу, динаміці рукописного почерку, серцевого ритму, ході.

На сьогоднішній день поняття «біометричний алгоритм» і «біометричний сканер» не обов'язково взаємопов'язані. Компанія може випускати ці елементи поодинці, а може спільно. Найбільша диференціація виробників сканерів і виробників софту досягнута на ринку біометрії папілярного візерунка пальців. Найменша на ринку сканерів 3D особи. По суті рівень диференціації багато в чому

відображає розвиненість і насиченість ринку. Чим більше вибору – тим більше тематика відпрацьована і доведена до досконалості. Різні сканери мають різний набір здібностей. В основному це набір тестів для перевірки підроблений об'єкт біометрії чи ні. Для сканерів пальців це може бути перевірка рельєфності або перевірка температури, для сканерів очі це може бути перевірка акомодації зіниці, для сканерів особи – рух особи.

Дактилоскопія (розпізнавання відбитків пальців) – найбільш розроблений на сьогоднішній день біометричний метод ідентифікації особистості. Каталізатором розвитку методу послужило його широке використання в криміналістиці 20 століття [26]. Кожна людина має унікальний папілярний візерунок відбитків пальців, завдяки чому і можлива ідентифікація. Зазвичай алгоритми використовують характерні точки на відбитках пальців: закінчення лінії візерунка, розгалуженні лінії, поодинокі точки. Додатково залучається інформація про морфологічну структуру відбитка пальця: відносне положення замкнутих ліній папілярного візерунка, «арочних» і спіральних ліній. Особливості папілярного візерунка перетворюються в унікальний код, який зберігає інформативність зображення відбитка. І саме «коди відбитків пальців» зберігаються в базі даних, використовуваної для пошуку і порівняння. Час переведення зображення відбитка пальця в код і його ідентифікація зазвичай не перевищує 1с, в залежності від розміру бази. Час, витрачений на піднесення руки – не враховується [27].

Переваги методу. Порівняно з такими методами ідентифікації як розпізнавання за сітківкою і райдужною оболонкою ока технологія сканування відбитків пальців є більш ніж у 10 разів дешевшою і зручнішою; порівняно з технологією ідентифікації за голосом, машинним та рукописним почерком відбитки пальців є статичні, тобто не змінюються, отже це не погіршує статистичну надійність технології. Досить проста процедура сканування відбитку не потребує спеціальних умінь чи знань і займає лише декілька секунд.

Недоліки: папілярний візерунок відбитка пальця дуже легко пошкоджується дрібними подряпинами, порізами. Люди, які брали сканери на підприємствах з

чисельністю персоналу близько декількох сотень людей заявляють про високий ступінь відмови сканування [28]. Так само присутній недостатня захищеність від підробки зображення відбитка, частково викликана широким поширенням методу.

На даний момент системи розпізнавання за відбитками пальців займають більше половини біометричного ринку. Безліч українські і зарубіжних компаній займаються виробництвом систем управління доступом, заснованих на методі дактилоскопічної ідентифікації. У зв'язку з тим, що цей напрямок є одним із самих давніх, воно отримало найбільше поширення і є на сьогоднішній день самим розробленим. Сканери відбитків пальців пройшли дійсно довгий шлях до поліпшення. Сучасні системи оснащені різними датчиками (температури, сили натискання і т.п.), які підвищують ступінь захисту від підробок. З кожним днем системи стають все більш зручними і компактними. Метод є достатньо розвиненим, крім того, більшість компаній виробляють готові системи, які оснащені всім необхідним, включаючи програмне забезпечення. Недоліком таких систем є досить висока вартість, тому існує актуальність розробки нового – дешевшого.

Багатофакторна ідентифікація. На сьогоднішній день вона набирає все більшої популярності. У багатофакторній ідентифікації для визначення користувача застосовують одночасно декілька параметрів.

Фактори ідентифікації можна комбінувати у будь якому порядку. Але у більшості випадків використається тільки одна пара: парольний захист і токен. Користувач у даному випадку може не боятися того, що пароль буде підібраний зловмисником все одно без електронного ключа він не буде працювати. Також дана система буде захищена навіть у випадку крадіжки токена, без пароля він по суті даремний. В той час деяких системах, де потрібний особливий захист даних, використовують такі процедури ідентифікації, у яких одночасно використаються і паролі, і цифрові ключі й біометричні характеристики людини [29].

Багатофакторна автентифікація не стандартизована. Існують різні форми її реалізації. Отже, проблема полягає в її здатності до взаємодії. Існує багато процесів і аспектів, які необхідно враховувати при виборі, розробці, тестуванні,

впровадженні та підтримці цілісної системи управління ідентифікацією безпеки, включаючи всі релевантні механізми автентифікації і супутніх технологій.

В комп'ютерних системах часто використовують комбінацію різних методів автентифікації, наприклад смарт-карту у вигляді предмета з унікальним вмістом (скажімо, криптографічним ключем), для доступу до якого необхідно ввести PIN-код. Тобто користувач для входу в систему не тільки повинен мати при собі смарт-карту, а й знати унікальну послідовність - PIN-код. Така автентифікація називається двухфакторною - по числу перевіряються параметрів[30].

До переваг багатофакторної ідентифікації можна віднести її здатність захистити інформацію, як від внутрішніх загроз, так і від зовнішніх вторгнень. Певною слабкістю можна вважати необхідність використання додаткових програмно-апаратних комплексів, пристроїв зберігання і зчитування даних. У той же час, зараз статистика зломів систем, які застосовують двофакторну автентифікацію, відсутня або незначна [31].

Багатофакторна або розширена автентифікація вже сьогодні застосовується низкою компаній у сфері фінансів при створенні сервісів інтернет-банкінгу, мобільного банкінгу, файлообміну і т.п. рішень для кінцевих користувачів. Вона заснована на спільному використанні декількох факторів автентифікації (знань, засобів або об'єктів зберігання однієї з інформаційних складових легітимної процедури автентифікації), що значно підвищує безпеку використання інформації, щонайменше, з боку користувачів, що підключаються до інформаційних систем по захищеним і незахищених каналах комунікацій.

Як приклад може послужити процес двофакторної автентифікації користувача, реалізований в низкою українських банків: вхід в особистий кабінет користувача за допомогою мережі інтернет можливий після введення пароля на сторінці, після чого (у разі підтвердженої правомірності), слід передача одноразового пароля (у вигляді SMS) на мобільний телефон, раніше зареєстрований користувачем [32].

Аналогічні схеми контролю і управління повноваженнями користувача, його подальших дій в корпоративних або інших інформаційних системах, можуть бути

реалізовані з застосуванням самих різних засобів і методів, вибір яких досить широкий, як по технологічності, вартості, виконання, так і щодо можливих комбінацій перерахованих властивостей.

Сесія роботи користувача може також контролюватися на предмет відповідності, як IP-адреси останньої успішно завершеною сесії, так і MAC-адреси відповідного мережевого обладнання. Далі можуть йти дії підтвердження або відмови в доступі до інформаційних ресурсів, але довіри до цими двома параметрами контролю бути не може в силу їх технологічної слабкості: IP-адреса можна підмінити, а MAC-адресу просто переписати в ході роботи системи, і навіть без перезавантаження . Проте, як якихось контрольних значень ці відомості можуть бути використані.

Недоліками є те, що багатофакторні рішення вимагають додаткових витрат на встановлення та оплату витрат в процесі експлуатації. Комплекси, що засновані на токенах, запатентовані, і більшість розробників періодично стягують з користувачів плату. Розмістити апаратні токени досить важко, адже вони можуть бути пошкоджені або втрачені. Крім витрат на установку багатофакторної автентифікації значну суму також становить оплата технічного обслуговування.

Якщо ж багатофакторна ідентифікації здійснюється через мобільний пристрій ряд недоліків значно зростає:

– щоб прийшло смс підтвердження, смартфон повинен бути у мережі;

– оскільки поширюється дані мобільного номеру, то існує велика ймовірність потрапляння у базу розсилки реклами та спаму;

– підтвердження по смс може бути перехоплене;

– потрібен деякий час на підтвердження;

Сучасні смартфони використовуються як для одержання пошти, так і для отримання SMS. Як правило електронна пошта на мобільному телефоні завжди включена. Таким чином, усі акаунти, для яких пошта є ключем, можуть бути зламані (перший фактор). Мобільний пристрій (другий фактор). Висновок: смартфон змішує два чинника в один [33].

1.3 Аналіз сучасного стану досліджуваної проблеми та практичний досвід її реалізації

Головною тенденцією, що характеризує розвиток сучасних інформаційних технологій є велика кількість комп'ютерних злочинів, та злочинів з розкраданням конфіденційної та іншої інформації. Існують дослідження, за результати яких, близько 60% опитаних стали жертвами комп'ютерних зломів за останній рік. Приблизно 15% опитаних з цього числа заявляють, що втратили понад мільйон доларів в ході нападів, більше 66 відсотків зазнали збитків в розмірі 45 тис. доларів. Понад 20% атак були націлені на таємниці виробництва або конфіденційні документи, в яких зацікавлені у першу чергу конкуренти[34].

Причин активізації комп'ютерних злочинів і пов'язаних з ними фінансових втрат досить багато, істотними з них є:

– перехід від традиційної "паперової" системи зберігання і передачі відомостей на електронну і недостатній розвиток технологій захисту цих даних;

– об'єднання обчислювальних систем, створення глобальних мереж і розширення доступу до інформаційних ресурсів;

– збільшення складності програмних засобів і пов'язане з цим зменшення їх надійності та збільшенням числа вразливостей.

Через відсутність у керівників підприємств і компаній чіткого уявлення з питань захисту інформації призводить до того, що їм складно повною мірою оцінити необхідність створення надійної системи захисту інформації на своєму підприємстві і тим більше складно буває визначити конкретні дії, необхідні для захисту тих чи інших конфіденційних відомостей . У загальному випадку керівники підприємств йдуть по шляху створення охоронних служб, повністю ігноруючи при цьому питання інформаційної безпеки. Негативну роль при цьому відіграють і деякі засоби масової інформації, публікуючи "панічні" статті про стан справ щодо захисту інформації, що формують у читачів уявлення про неможливості в сучасних

умовах забезпечити необхідний рівень захисту інформації. Можна з упевненістю стверджувати, що створення ефективної системи захисту інформації сьогодні цілком реально. Надійність захисту інформації, перш за все, буде визначатися повнотою вирішення цілого комплексу завдань, мова про які буде продовжена далі.

Для розробки необхідного комплексу заходів щодо захисту інформації бажано залучення кваліфікованих експертів в області захисту інформації. Традиційно для організації доступу до конфіденційної інформації використовувалися організаційні заходи, засновані на суворому дотриманні співробітниками процедур допуску до інформації, що визначаються відповідними інструкціями, наказами та іншими нормативними документами. Однак з розвитком комп'ютерних систем ці заходи перестали забезпечувати необхідну безпеку інформації [35].

Загальновизнаним фактом є констатація ненадійності, незручності і неефективності застарілої парольної системи ідентифікації. Уявною виявляється і удавана «безкоштовність» паролів: за даними Lenovo, до 50% всіх звернень до служби технічної підтримки викликані проблемами з паролями, а середня вартість виконання одного звернення в техпідтримку становить від 25 до 50 доларів (Compulenta).

Досить популярні рішення, які використовують для ідентифікації різні матеріальні носії – смарт-карти, токени, «таблетки» Touch Memory (i Button). У деяких з цих коштів захист ідентифікаційної інформації забезпечується шифруванням, генерацією одноразових паролів, електронним підписом і цифровими сертифікатами.

Найбільш ефективним є застосування зазначених носіїв в комплексі із засобами біометричної ідентифікації. Біометрія може використовуватися і самостійно; в будь-якому з цих варіантів зберігаються її головна перевага – точна, безпечна, швидка та зручна ідентифікація користувача [36].

Зазвичай заходи щодо захисту інформації пов'язані з додатковим навантаженням і новими обов'язками, покладеними на користувачів,

адміністраторів, співробітників служби безпеки. Біометрія є винятком: її впровадження полегшує діяльність працівників усіх зазначених груп, одночасно забезпечуючи істотне зростання рівня інформаційної безпеки.

Для захисту інформації застосовується сервіс BioLink IDenium, який реалізує функції ідентифікації користувачів корпоративних мереж і управління їх доступом до інформаційних ресурсів. Сервіс BioLink IDenium реалізує ідентифікацію користувачів по унікальним біометричним ознаками – відбитками пальців.

Ефективна система біометричної ідентифікації виконує наступні функції:

– уніфікація процесів доступу до інформаційних ресурсів на основі єдиного ідентифікатора - відбитка пальця;

– захист інформаційних ресурсів від несанкціонованого доступу і / або доступу, здійснюваного з порушенням встановлених правил (політики безпеки);

– автоматизація та централізація управління користувачами, їх обліковими записами і відповідними повноваженнями в операційних системах і різних прикладних продуктах;

– прискорення доступу легальних користувачів до ресурсів корпоративної мережі із забезпеченням максимальної простоти і прозорості цього процесу;

– скорочення непродуктивних втрат, викликаних помилками при введенні логіна / пароля, блокуванням облікових записів і т.п .;

– оптимізація діяльності системних адміністраторів за рахунок зменшення числа звернень, пов'язаних з помилками користувачів при ідентифікації, автентифікації і авторизації;

– швидка реєстрація нових користувачів і оперативну зміну облікових записів співробітників, які перейшли в інший підрозділ, звільнених і т.п.

Сервіс BioLink IDenium впроваджений в Microsoft Active Directory (AD), можливості якого активно використовуються IT-службами компаній і підприємств. Таким чином, перехід на систему біометричної ідентифікації максимально полегшений, здійснюється в звичному для адміністраторів порядку, а подальша

експлуатація сервісу BioLink IDenium і керування користувачами проводиться в стандартному інтерфейсі AD.

Установка BioLink IDenium здійснюється централізовано; абсолютна більшість відповідних операцій виконується автоматично, від адміністратора потрібно лише загальне керівництво цим процесом.

Сервіс BioLink IDenium не зберігає зображення біометричних ідентифікаторів. При розпізнаванні користувача формується цифрова модель висунутого відбитка пальця, яка порівнюється з моделлю раніше зареєстрованого біометричного ідентифікатора.

Порівняння цифрових моделей здійснює програмний сервер - IDenium Server. Цей сервер реєструється в Microsoft Active Directory автоматично; в корпоративної мережі можна встановити кілька біометричних серверів, що забезпечить балансування навантаження між ними в періоди пікового навантаження (початок / закінчення робочого дня і т.п.).

Наявність декількох серверів IDenium Server гарантує відмовостійкість системи біометричної ідентифікації, безперервність доступу користувачів до ресурсів корпоративної мережі і безперебійне здійснення відповідних бізнес-процесів [37].

Масштабованість сервісу біометричної ідентифікації дозволяє нарощувати кількість його користувачів, а також планомірно охоплювати нові територіальні та функціональні підрозділи підприємства, інкорпоруючи їх в уже наявну інфраструктуру доступу.

Головним недоліком даного сервісу є досить висока вартість. Даний продукт потребує регулярного оновлення системи. Також досить великий ризик загубити електронний носій – ключ, без якого ідентифікація та авторизація користувача неможлива.

Усі наявні продукти на ринку аналогічні цьому і мають свої недоліки. Ідеального – не існує, тому існує необхідність розробки нового програмного модуля ідентифікації з подальшою авторизацією користувача захищеної системи.

1.4 Висновки до розділу

Порівнюючи усі види ідентифікації особистості, можна зробити висновок що кожний метод має свої переваги і недоліки, але біометричні найактуальніші та найстійкіші на ринку інформаційної безпеки. Переваги біометричних систем – незаперечні. Висока швидкість обробки, постійність авторизаційної інформації в поєднанні з порівняно доступною ціною, все це беззаперечно повинно схилити до впровадження біометричних систем ідентифікації.

Аналіз сучасного стану ефективності компонентів безпеки комп’ютерних систем показав, що актуальними та доцільними є розробка нового, ефективнішого та точнішого методу захисту інформаційних систем із використанням біометричних даних.

2 ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ ІДЕНТИФІКАЦІЇ ЗА ДОПОМОГОЮ БІОМЕТРИЧНИХ ДАНИХ ЯК ОПТИМАЛЬНОГО ПІДХОДУ ДО РОЗВ'ЯЗКУ ПОСТАВЛЕНИХ ЗАДАЧ

На теперішній час у зв'язку з інтенсивним зростанням потреби в обчислювальних і телекомунікаційних системах, розвитком комп'ютерної техніки і збільшенням числа користувачів, все більша увага приділяється питанням контролю доступу до їх ресурсів. Подібний контроль реалізується через задані адміністратором обмеження доступу, тобто, можливості виконати певні дії, користувачів по відношенню до ресурсів системи. У даному розділі здійснюється огляд біометричних методів ідентифікації та базових понять методів розмежування доступу, покращення методу за допомогою відбитків пальців та проектування бази даних користувачів.

2.1 Класифікація методів біометричної ідентифікації

Серед біометричних методів ідентифікації можна виділити такі:

– по статичних ознаках – це ті ознаки, що залишаються майже незмінними з часом, від самого народження людини (фізіологічні характеристики);

– по динамічних ознаках – це поведінкові характеристики, тобто такі характеристики, що основані на особливостях та характерних рухів, що виникають в процесі відтворення якоїсь дії. Звичайно, ці ознаки можуть поступово змінюватися з часом [38].

Статичні методи ідентифікації користувача:

– Ідентифікація за відбитком пальця. Основою цього методу є унікальність малюнку папілярних узорів на пальцях. Ідентифікація влаштована так: за допомогою сканера отримують зображення відбитку, потім це зображення перетворюють на спеціальний цифровий код, який далі порівнюється з кодами, які зберігаються в базі даних.

– Ідентифікація по розташуванню вен на долоні. Приладом, що зчитує інформацію в даному випадку, є інфрачервона камера. У результаті на вході програми при формуванні цифрового коду з'являється малюнок вен на руці людини.

– Ідентифікація по сітківці ока. В даному випадку сканується малюнок кровоносних судин очного дна. Зрозуміло, що цей малюнок спостерігається тільки за певних умов, тому при скануванні людина дивиться на видалене світлове джерело і спеціальна камера сканує його очне дно.

– Ідентифікація за райдужкою ока. Малюнок райдужки ока – унікальний для кожної людини. В даному методі важливим компонентом є не тільки камера, але і програмне забезпечення, яке надійно функціонує. Саме програмний продукт виділяється малюнок із зображення потрібної райдужки ока.

– Ідентифікація за формою кисті руки. Цей метод ґрунтується на розпізнаванні геометричних особливостей кисті руки. Спеціальний сканер формує тривимірний малюнок кисті. При аналізі цього малюнка виконуються вимірювання, за допомогою яких формується відповідний цифровий код.

– Ідентифікація за формою обличчя. Метод базується на побудові тривимірного, або двовимірного образу обличчя. Спеціальні програми розпізнають контури очей, губ і інших частин лиця. Далі проводяться точні вимірювання між отриманими контурами. Саме за цими даними будується цифровий код.

Серед динамічних методів можна назвати такі:

– Ідентифікація за голосом. На ринку є велика кількість програм по розпізнаванню голосу. У методі ідентифікації за голосом важливу роль відіграють частотні характеристики голосу людини. Саме за ними і будується цифрова модель.

– Ідентифікація за почерку. Даний метод досліджує підпис людини та перевіряє такі динамічні характеристики: силу натиску на поверхню, швидкість нанесення підпису та графічні параметри. За цими характеристиками і будується цифровий код.

– Ідентифікація за клавіатурним почерком. Суть методу у тому, що користувачу необхідно надрукувати кодове слово. Цифровий код будується за динамікою набору певного слова [39].

У таблиці 2.1 наведено основні характеристики перерахованих вище методів біометричної ідентифікації [40].

Таблиця 2.1 – Основні характеристики методів біометричної ідентифікації

Метод отримання біометричних параметрів	Ймовірність відмови у доступі, %	Ймовірність помилкової ідентифікації «чужого» (без використання муляжу), %	Ймовірність помилкової ідентифікації «чужого» (з використанням муляжу), %	Збереження таємниці образу у процесі ідентифікації абонента	Вартість технічної реалізації в грошовому еквіваленті, у.о.
Геометрична будова руки	0,2…4	0,2….1	10…75	неможливо приховати	Від 600 до 3000
Відбитки пальців	2...6	0,0001	10...70	неможливо приховати	Від 60 до 600
Особливості малюнка сітківки ока	0,4	6...10	_____	неможливо приховати	приблизно 4000
Райдужна оболонка ока	0,2…2	0,0001	_____	неможливо приховати	Від 500 до 6000
Портрет обличчя	0,8…1	_____	_____	неможливо приховати	55000
Рукописний почерк	0,5…5	0,5…5	0,5...5	8-10…10-40	_____
Клавіатурний та комп'ютерний почерк	3…9	3...9	_____	6-10…10-12	_____
Характеристики і особливості мови	0,5...5	0,5...5	25...90 (запис)	10-16…10-30	1...60

Технології біометричної ідентифікації з кожним роком набирають все більшу популярність. За статистикою використання даних методів на 2014 рік (рис 2.1), 53% від усіх використовуваних методів складала ідентифікація за відбитками пальців, 21% – ідентифікація за геометріє обличчя, 26% – інші методи. Проте, за

статистикою, на 2020 рік прогнозують покращення технології ідентифікації за геометрією обличчя, та її популярність зросте до 30% від усього ринку (рис 2.2).

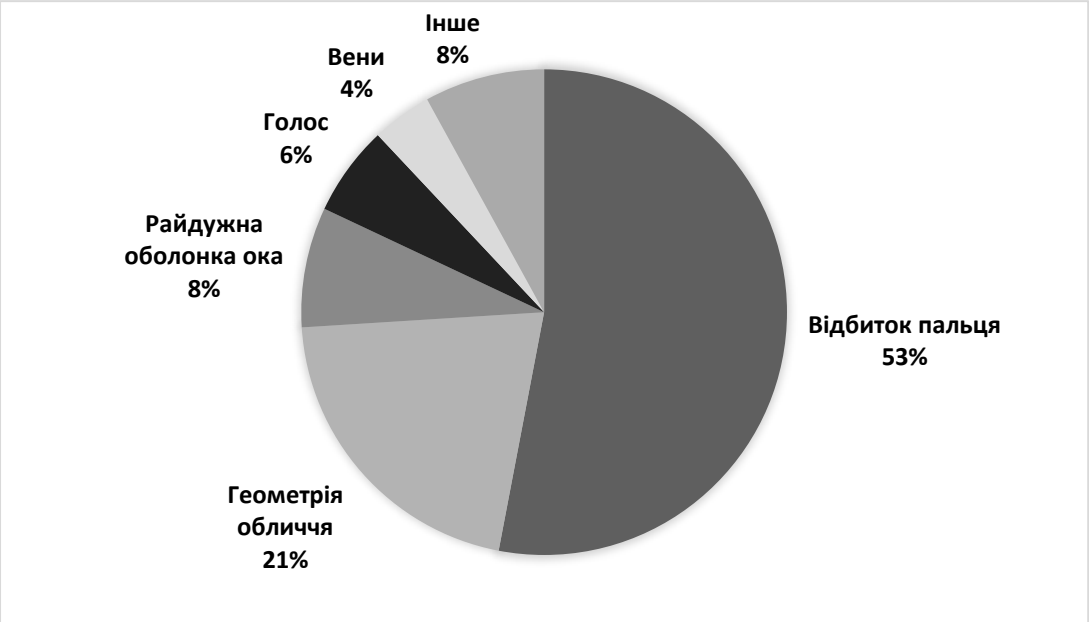


Рисунок 2.1 – Статистика використання біометричних методів ідентифікації за даними на 2014 рік

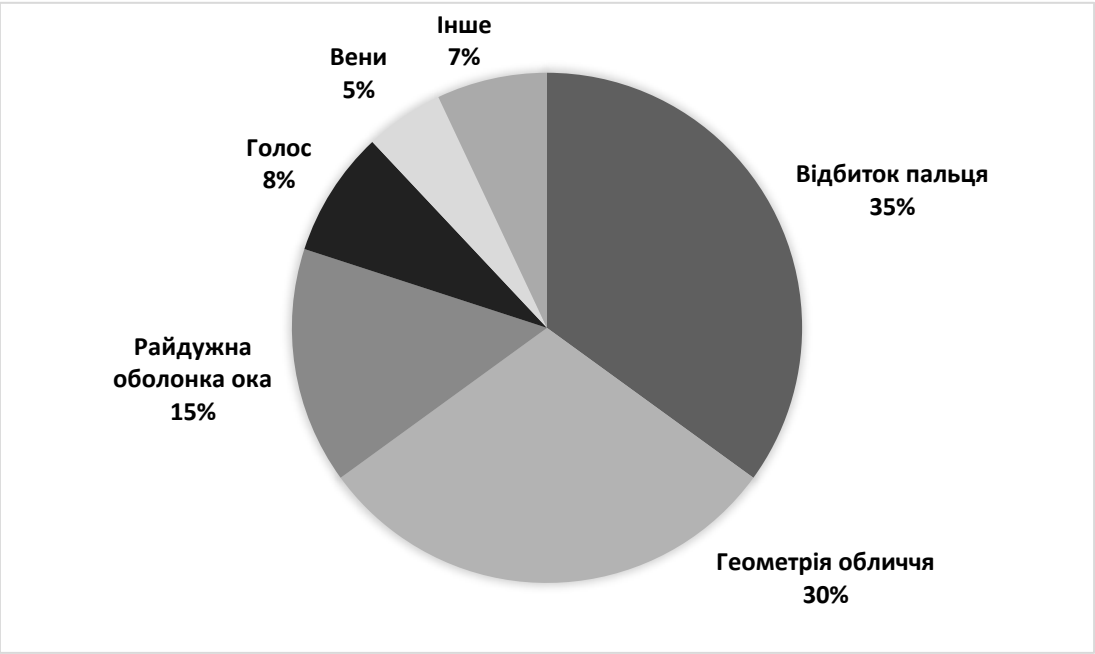


Рисунок 2.2 – Прогнозована статистика використання біометричних методів ідентифікації на 2020 рік

Проаналізувавши статистику та надійність кожного методу, обрано два оптимальних – ідентифікація за відбитками пальців та ідентифікація за геометрією обличчя.

2.2 Базові поняття методів розмежування доступу

Одним із важливих питань інформаційної безпеки є правильне розмежування доступу до інформації з метою забезпечення контролю доступу до конфіденційних даних сторонніх осіб.

Система розмежування доступу здійснює контроль за доступом суб'єктів до об'єктів. Об'єкт доступу — це той елемент, до його потрібно контролювати доступ і обмежувати його. Важливу роль відіграє метод доступу – це та дія, яку виконує об’єкт над суб’єктом.

Задачею розмежування доступу є скорочення кількості користувачів, що не мають відношення до певної категорії інформації при виконанні своїх функцій, тобто захист інформації від порушника серед допущеного до неї персоналу.

Також слід звернути увагу що метод і право доступу це різні речі. Право доступу — це можливість здійснювати доступ до об'єкта з використанням певних методів. Також суб’єкт може мати так звані привілеї. Привілей – це право доступу що використовує певний метод доступу, що надає суб'єкту доступ на всі об'єкти, які використовують цей метод [41].

Для розмежування доступу існують певні правила. Ці правила є певним абстрактним механізмом, який виступає посередником при будь-яких взаємодіях користувачів з інформаційною системою. Правила розмежування доступу мають бути чіткими на рівні вибору профілю захищеності, та впровадження рганізаційних заходів захисту. Частиною правил розмежування доступу є керування доступом (рис 2.1).

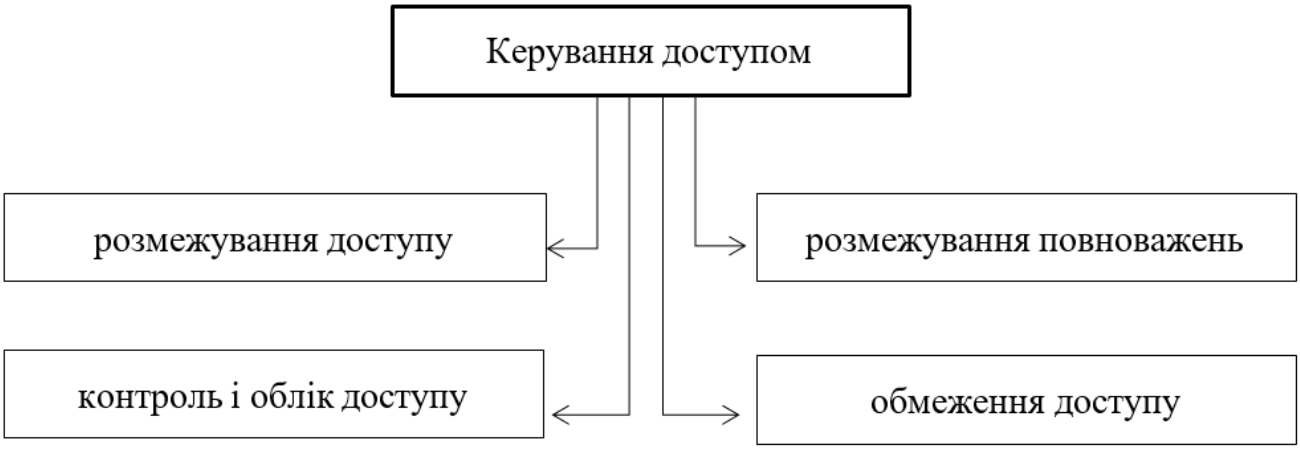


Рисунок 2.3 Елементи системи керування доступом

В інформаційних системах доцільно реалізовувати адміністративне керування доступом, оскільки тільки адміністратори повинні мати право здійснювати дії над користувачами та об'єктами.

Суть розмежування доступу полягає в правильній організації надання доступу користувачам згідно до їхніх повноважень та функціональних обов’язків. Для правильного розподілу необхідно скористатись стандартними кроками (рис 2.4)[42].

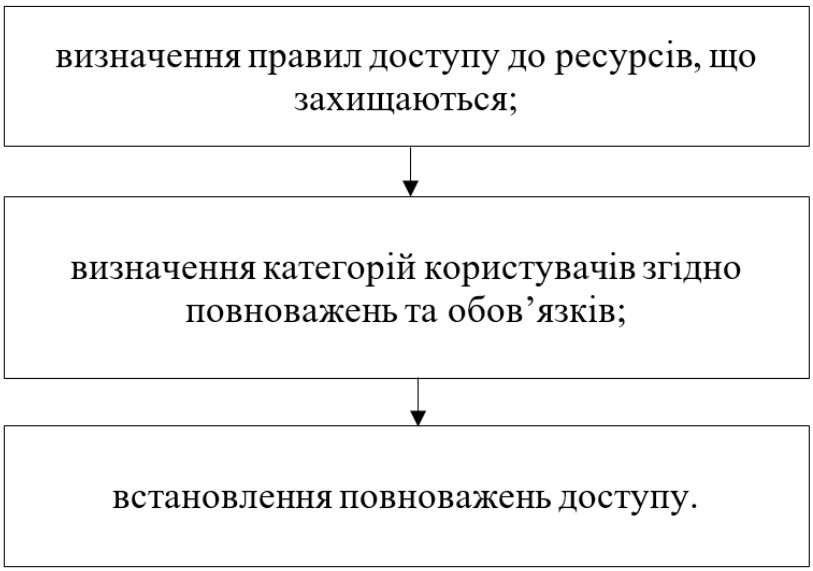


Рисунок 2.4 Кроки правильної організації доступу

Існують різні підходи до визначення можливостей учасників інформаційної взаємодії в доступі до різних ресурсів. Основні з них є такі технології [43]:

– технології систем дискреційного;

– технології систем мандатного;

– технології рольового розмежування доступу;

– суб'єктно-орієнтовна технологія ізольованого програмного середовища. Кожне з цих напрямків є описом набору правил, на підставі аналізу яких приймається рішення про доступ. Разом з тим, реалізація механізмів розмежування доступу до ресурсів кожної конкретної комп'ютерної системи, як правило, ґрунтується на складніших, аніж базові технології, що враховують окремі особливості такої системи, середовища її оточення та положення політики її інформаційної безпеки. У кожному з цих випадків базові моделі або їх комбінації деталізуються цілим рядом додаткових обмежень і правил. З метою вибору напрямку подальшого дослідження проведемо порівняльний аналіз базових технологій розмежування доступу до ресурсів комп'ютерних систем, що представлені на рисунку 2.5.

Рисунок 2.5 – Технології розмежування доступу в комп'ютерних системах

В цілому процедура надання доступу виглядає таким чином:

Коли автентифікований користувач намагається отримати доступ до захищеного ресурсу, оцінка здійснюється у наступному порядку:

1. Відповідність ідентифікатора користувача за допомогою записів користувача у базі. Оцінка зупиняється на вході користувача в систему. Дозволи, які надаються є правами доступу в запису відповідності користувача.

2. Якщо немає відповідного входу до користувача, то визначаються групи, до яких належить користувач, і зіставлення цих групи до записів групи в базі. Оцінка зупиняється на будь-якому груповому рівні. Якщо підписано більше ніж один запис групи, то отримані дозволи є найбільш допустимими для відповідних записів.

3. Якщо немає відповідних записів користувача або групи, надаються дозволи будь-яких іншої записів, якщо вони існують.

4. Якщо немає відповідних записів користувача або групи, і немає будь-яких інших записів, то користувач не має прав.

5. Коли неаутентифікований користувач намагається отримати доступ до захищеного ресурсу, оцінка здійснюється таким чином:

6. Якщо база не містить запис для неаутентифікованого, то доступ заборонено.

7. Якщо база не містить запис для будь-якого іншого, то доступ заборонено.

8. Якщо база містить запис для неаутентифікованого і запис для будь-якого іншого, надаються дозволи, які надаються як для неаутентифікованого, так і будь-якого іншого. Права, надані неавтентифікованим користувачам, не перевищують дозволів, наведених у будь-якому іншому записі (рис. 2.6) [44].

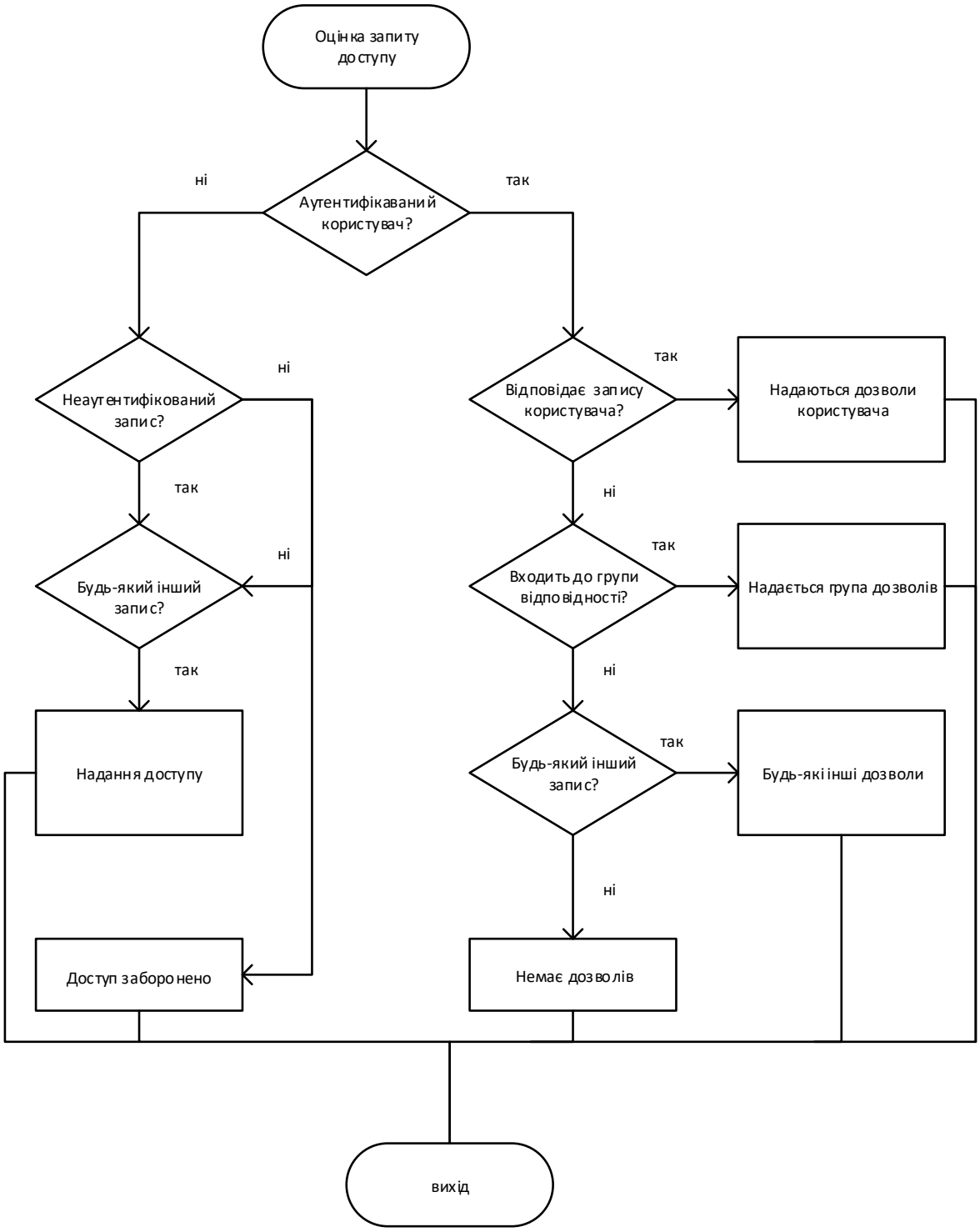


Рисунок 2.6 – Оцінка контролю доступу

Доступ за ролями нейтральний по відношенню до конкретних видів прав і способам їх перевірки; його можна назвати об'єктно-орієнтованим каркасом, що значно полегшує процес адміністрування. Рольовий доступ дозволяє підсистемі

розмежування доступу бути більш керованою при будь-якому числі користувачів. Це відбуваються за рахунок зав’язків, що встановлені між ролями, аналогічні наслідуванню в об'єктно-орієнтованих системах. У таких системах ролей існує менше ніж користувачів. У результаті число зв'язків, що адмініструються, стає набагато менше.

Основні поняття керування доступом за ролями:

- користувач;
- сеанс роботи користувача;
- роль (визначена за організаційною структурою підприємства);
- об'єкт (те, до чого обмежується доступ);
- операція (зчитування, редагування, запис, виконання і т.п.; для деяких об'єктів операції можуть бути набагато складнішими);
- право доступу (дозвіл виконувати певні операції над певними об'єктами).

Ролям присвоюються користувачі та їх права доступу; можна вважати, що ролі мають відношення "багато до багатьох" між користувачами і правами. Одна роль може належати декільком користувачам; один користувач може мати декілька ролей. Коли користувач виконує роботу, активізується підмножина ролей, до яких він належить, в результаті чого він отримує всі права, що належать усім його ролям.

Для програмного модулю ідентифікації користувача за біометричними даними через смартфон з подальшою авторизацією було обрано рольове управління доступом через низку переваг описаних нижче.

Рольове управління доступом має низку важливих позитивних властивостей. наприклад:

Можливість побудови ієрархії ролей з успадкуванням набору прав. Це дозволяє спростити рольову модель, особливо в організаціях з різнорідної інфраструктурою, де використовується багато інформаційних систем. З використанням ієрархії немає необхідності повторно вказувати права в декількох подібних ролях, досить помістити їх в одну велику роль, як дочірні, вказавши лише унікальні права для кожної ролі.

Просто і ефективно здійснюється надання однакових прав великій кількості користувачів - досить призначити користувачам одну роль.

При необхідності зміни набору прав великій кількості користувачів досить змінити набір прав в ролі.

Можливість реалізації принципу поділу повноважень. Це значно знижує ризик надання користувачам надлишкових повноважень, наприклад, коли дві ролі не можуть бути в один момент часу призначені одному користувачеві.

Особливості. При проектуванні, впровадженні та використанні рольової моделі управління доступом потрібно брати до уваги чинники, які можуть привести до серйозних втрат часу і коштів.

По-перше, «різноманітність користувачів»: на практиці у великих організаціях може виявитися досить велика кількість користувачів з унікальними правами, при цьому деякі з них можуть бути на одній посаді, а то і в одному підрозділі. Це ускладнює побудову рольової моделі і може привести до ситуації, коли кожному користувачеві необхідна своя унікальна роль. Такі ситуації можуть виникнути, коли співробітник «виріс» в рамках своєї посади або у нього просто є унікальні функції в рамках свого підрозділу. Це може стати серйозною проблемою для системи управління доступом:

В такому випадку важко визначити "розумно малий" набір ролей, які відповідають за права доступу основної маси користувачів.

Непрактично створювати стільки ж ролей скільки користувачів - це рівносильно ручного управління доступом.

По-друге, «занадто багато ролей»: це не завжди так, але може статися така ситуація, коли при підключенні до системи управління доступом ще однієї керованої системи, ролі, певні для раніше підключених систем, необхідно роздрібнити на кілька інших ролей, з урахуванням всіх можливих варіантів використання спільно з новою системою. Якщо таких нових систем кілька, то може виникнути ситуація, коли ролей виявиться більше ніж користувачів.

По-третє, «зміна обов'язків користувачів і реорганізація бізнесу»: навіть якщо рольова модель відображає поточну ситуацію в організації, її необхідно

підтримувати в актуальному стані, відстежувати зміни обов'язків користувачів і швидко вносити зміни в рольову модель.

2.2 Проектування бази даних користувачів

Основою створення бази даних може бути одна модель, або сукупність декількох. Моделей створення існує три: реляційна, ієрархічна та мережева.

Реляційна модель це двовимірний масив або двовимірна таблиця, а для створення складніших моделей баз даних можна використовувати сукупність взаємопов’язаних таблиць. Запис – рядок цієї таблиці, а стовпець – це поле [45].

Властивості реляційної моделі такі:

- кожний елемент таблиці – це один елемент даних;
- унікальне ім'я для кожного поля;
- усі стовпці в таблиці однакового типу;
- відсутні однакові рядки;
- порядок слідування рядків у таблиці може бути довільним і може характеризуватися кількістю полів, кількістю записів, типом даних.

Ця модель бази даних є зручною для сортування, пошуку певних записів та для роботи з вибірками даних.

Реляційна модель складається з таблиць, що повязані між собою ключами. Ключ – це поле, яке точно визначає відповідний запис. На сьогоднішній день реляційна модель даних користується найбільшою популярністю, через свою зручність.

Ієрархічна модель – обернене дерево (граф), що створене з сукупності елементів, що розташовані у порядку їх підкорення від загального до часткового. Основні характеристики такої моделі це рівні, вузли та зв’язки. Вузол — це інформаційна модель елемента, що міститься на даному рівні ієрархії.

Мережева модель даних дещо схожа з ієрхарічною. У неї такі ж складові вузол, рівень, зв’язок, однак характер їх відносин принципово інший, за рахунок вільного зв’язку між елементами різних рівнів [46].

Проектування методом сутність–зв’язок складається з шести етапів (рис 2.7).

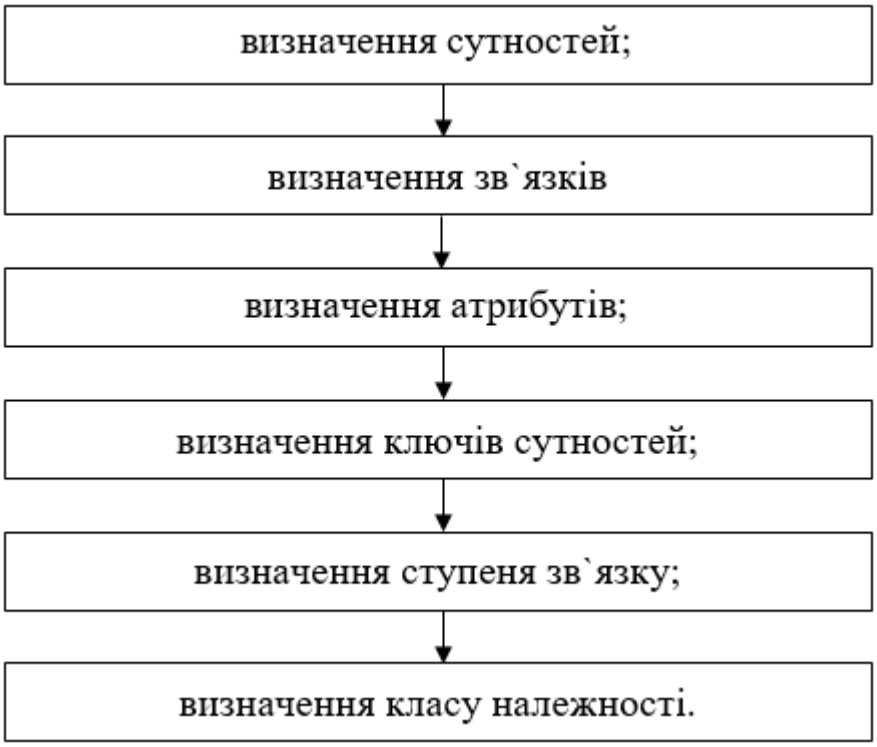


Рисунок 2.7 – Етапи проектування за методом сутність-зв’язок

Для розробки бази даних користувачів програмного модулю ідентифікації користувача за біометричними даними було обрано ER-модель, оскільки для даної розробки вона є оптимальним рішенням.

Оскільки етапі проектування необхідно визначити сутності. Сутність – це деякий об`єкт, що представляє інтерес для користувача. З точки зору заданої предметної області користувачем бази даних являється адміністратор. Для вирішення поставленої задачі буде доцільно визначити такі сутності: КОРИСТУВАЧ, РІВЕНЬ_ДОСТУПУ, РОЛЬ, ДАНІ.

На другому етапі необхідно визначити зв`язки між сутностями:

- КОРИСТУВАЧ – має – РІВЕНЬ ДОСТУПУ;
- ДАНІ – визначає - КОРИСТУВАЧ;
- РОЛЬ – призначена – КОРИСТУВАЧ;
- РОЛЬ – впливає – РІВЕНЬ_ДОСТУПУ;

Далі потрібно визначити атрибути (властивості) кожної сутності:

– КОРИСТУВАЧ (Код_користувача, ПІБ, Дата_народження, Телефон, Код_доступу, Код_даних, Код_ролі, Номер_паспорту);

– РІВЕНЬ_ДОСТУПУ (Код_доступу, Вид_доступу, Права);

– ДАНІ (Код_даних);

– РОЛЬ (Код_ролі, Назва_ролі, Опис_ролі).

Четвертий етап – визначення ключів сутностей. Ключем сутності називається атрибут або набір атрибутів, що дозволяють однозначно ідентифікувати екземпляр сутності. Ключами для даних сутностей будуть:

– КОРИСТУВАЧ (<Код_користувача>, ПІБ, Дата_народження, Телефон, Код_доступу, Код_даних, Код_ролі, Номер_паспорту);

– РІВЕНЬ_ДОСТУПУ (<Код_доступу>, Вид_доступу, Права);

– ДАНІ (<Код_даниху>);

– РОЛЬ (<Код_ролі>, Назва_ролі, Опис_ролі).

П`ятий та шостий етапи потребують визначити ступінь зв`язку та клас належності

Визначення ступеня зв’язку класу належності сутностей «КОРИСТУВАЧ та РІВЕНЬ_ДОСТУПУ» показано на рисунку 2.8.

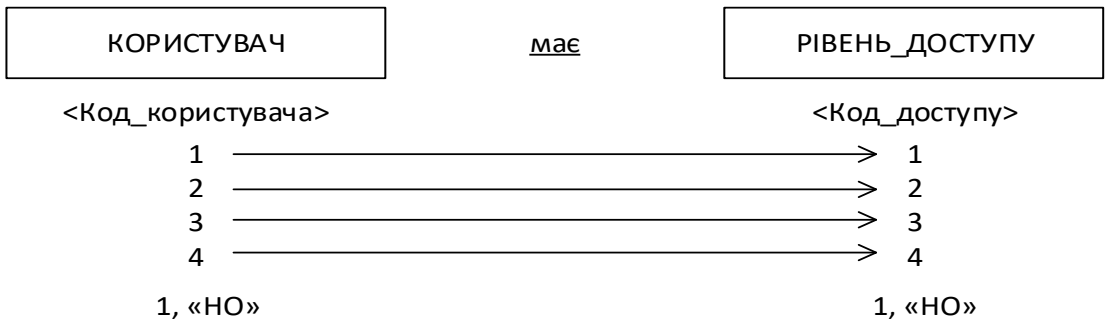


Рисунок 2.8 – Ступінь зв’язку та клас належності сутностей КОРИСТУВАЧ та РІВЕНЬ_ДОСТУПУ

Отже, ступінь зв’язку між сутностями 1:1, клас належності «НО»:«НО».

Визначення ступеня зв’язку класу належності сутностей ДАНІ та КОРИСТУВАЧ показано на рисунку 2.9.

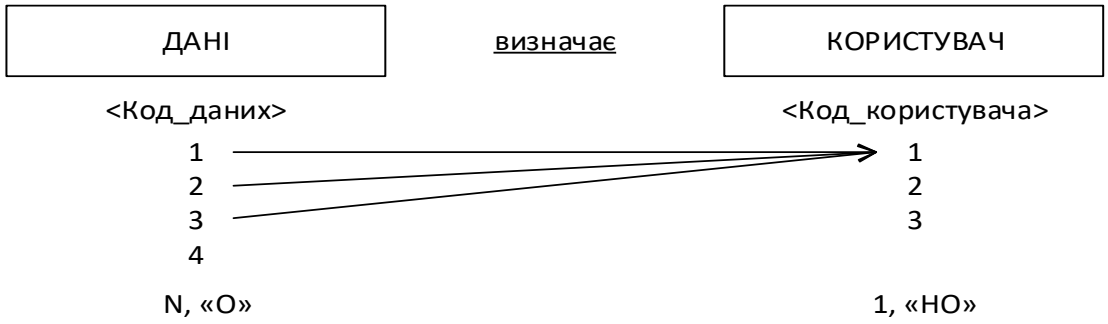


Рисунок 2.9 – Ступінь зв’язку та клас належності сутностей ДАНІ та КОРИСТУВАЧ

Отже, ступінь зв’язку між сутностями N:1, клас належності «О» : «НО».

Визначення ступеня зв’язку класу належності сутностей РОЛЬ та КОРИСТУВАЧ показано на рисунку 2.10.

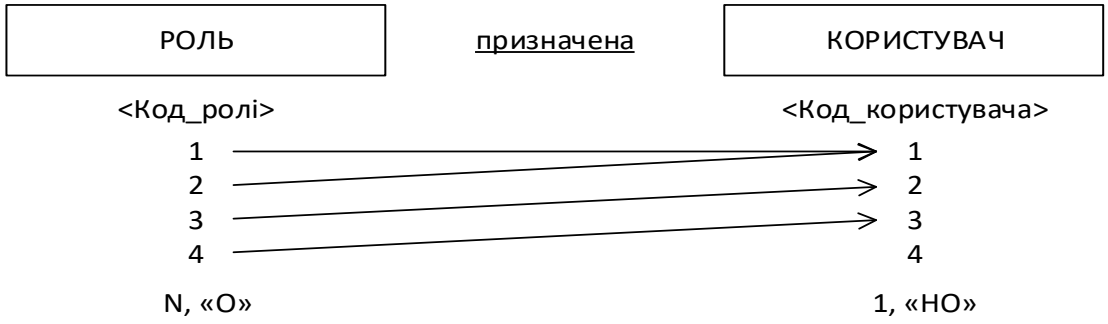


Рисунок 2.10 – Ступінь зв’язку та клас належності сутностей РОЛЬ та КОРИСТУВАЧ

Отже, ступінь зв’язку між сутностями N:1 клас належності «О» : «НО».

Визначення ступеня зв’язку класу належності сутностей РОЛЬ та РІВЕНЬ_ДОСТУПУ показано на рисунку 2.11.

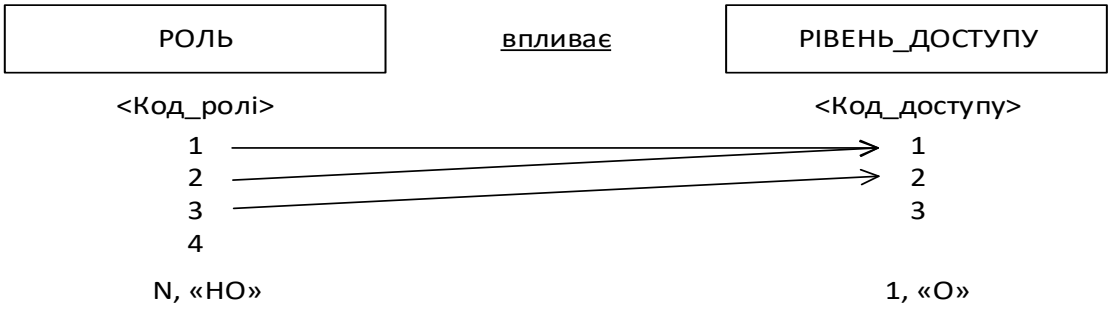


Рисунок 2.11 – Ступінь зв’язку та клас належності сутностей РОЛЬ та РІВЕНЬ_ДОСТУПУ

Отже, ступінь зв’язку між сутностями N:1, клас належності «О»:«НО».

Проектування ER-моделі. Дана модель показує як циркулює інформація у системі, яким чином сутності нею. Результат проектування ER-моделі зображений на рисунку 2.12.

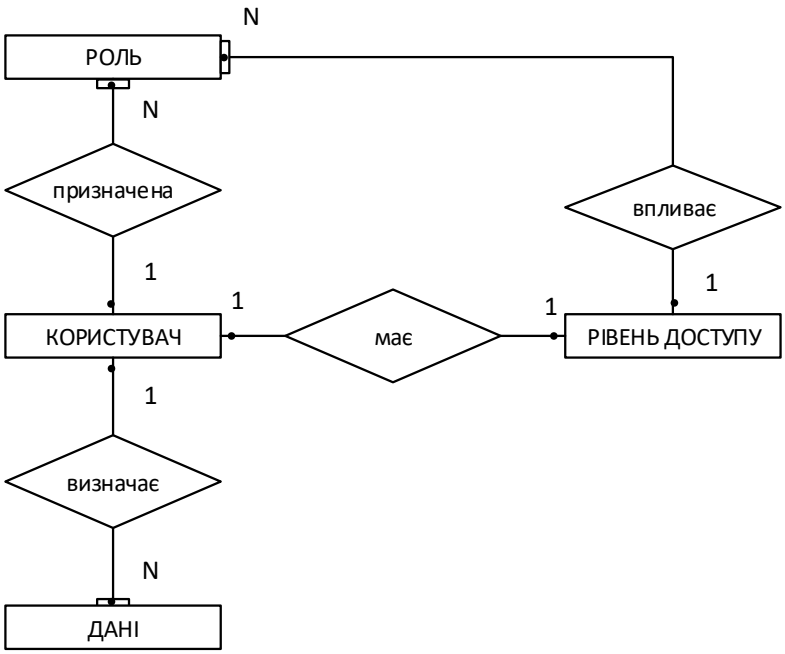


Рисунок 2.12 – ER-модель бази даних користувачів

Нормалізація отриманих відношень. Проектування бази даних за методом нормалізації відношень передбачає проведення таких етапів проектування:

1) Для вирішення поставленої задачі необхідно виділити інформаційні об’єкти: КОРИСТУВАЧ, РІВЕНЬ_ДОСТУПУ, ДАНІ, РОЛЬ.

2) Визначення інформація, яка необхідна при роботі по кожному інформаційному об’єкту:

– КОРИСТУВАЧ (<Код_користувача>, ПІБ, Дата_народження, Телефон, Код_доступу, Код_даних, Код_ролі, Номер_паспорту);

– РІВЕНЬ_ДОСТУПУ (<Код_доступу>, Вид_доступу, Права);

– ДАНІ (<Код_даних>);

– РОЛЬ (<Код_ролі>, Назва_ролі, Опис_ролі).

На основі проведеного аналізу необхідно скласти універсальне відношення для даної задачі. Для реалізації поставленої задачі універсальне відношення буде мати вигляд:

R: (Код_користувача, ПІБ, Дата_народження, Телефон, Код_доступу, Код_даних, Код_ролі, Номер_паспорту, Код_доступу, Вид_доступу, Права, Код_даних, Код_ролі, Назва_ролі, Опис_ролі).

3) Якщо відношення знаходиться в першій нормальній формі (1НФ) , то усі не ключові атрибути функціонально залежні від ключа. Первинним ключем для даного відношення буде <Код_користувача, Код_даних>. В випадку даного модулю відношення знаходиться в першій нормальній формі:

R: <Код_користувача, Код_даних>, ПІБ, Дата_народження, Телефон, Код_доступу, Код_даних, Код_ролі, Номер_паспорту, Код_даних.

4) Відношення знаходиться в другій нормальній формі (2НФ), якщо воно знаходиться в 1НФ і кожен не ключовий атрибут функціонально повно залежить від складеного ключа.

Виходячи з цих правил отримуємо такі відношення в 2НФ:

R1: <Код_користувача>, ПІБ, Дата_народження, Телефон, Код_доступу, Код_даних, Код_ролі, Номер_паспорту.

R2: <Код_доступу>, Вид_доступу, Права.

6) Відношення знаходиться в третій нормальній формі (3НФ), якщо воно знаходиться в 2НФ і в ньому відсутні транзитивні залежності не ключових атрибутів від ключа.

В даному варіанті транзитивна залежність існує у відношенні:

R3<Код_даних>.

R4<Код_ролі>, Назва_ролі, Опис_ролі

R5:<Код_доступу>, Вид_доступу.

R6:<Код_ролі>, Суть_ролі.

У результаті проектування за методом нормалізації відношень виявилось що R2=R5, R4=R6, тому кінцевими відношеннями будуть: R1, R2, R3, R4.

2.3 Висновки до розділу

У даному розділі було проаналізовано різні мотоди біометричної ідентифікації. У результатів досліджень найактуальнішими, надійнішими було обрано два методи – ідентифікація за відбитками пальців та за геометрією обличчя.

Також у розділі було оглянуто технології розмежування доступу, у результаті було виявлено низку характерних особливостей, переваг і недоліків кожного з них. Було обрано оптимальний – розмежування доступу на основі ролей.

Суть даного методу така: підприємство чи організація розподіляє ролі у відповідності до робочих функцій. Далі цим ролям надаються повноваження для виконання певних дій. На відміну від інших методів даний не бере до уваги поточну ситуацію.

Було спроектовано базу даних користувачів за допомогою моделі «сутність-зв’язок». Проектування містило такі етапи: визначення сутностей; визначення зв`язків; визначення атрибутів; визначення ключів сутностей;  визначення ступеня зв`язку; визначення класу належності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЮ

3.1 Вибір оптимальної мови програмування

Вибір мови програмування одне із важливих завдань під час створення будь якої системи. Мова програмування - це інструмент, за допомогою якого будуються правила системи.

Для вибору оптимальної мови програмування необхідно дотримуватись таким критеріям:

– можливості даної мови (мова має вміти опрацьовувати функції, які необхідні у програмі);

– доступність зручного та актуального середовища програмування.

Для реалізації мобільної частини програмного модулю ідентифікації користувачів на основі біометричних даних було обрано мову програмування Java.

Java – це об'єктно-орієнтована мова програмування, особливістю якої є те, що програми на Java запускаються на будь-яких комп’ютеризованих пристроях, які працюють на різних операційних системах (Windows, Linux, MacOS), навіть без повторної компіляції коду.

Синтаксис мови дещо схожий на сімейство мов C, зокрема основою є об’єктна модель, що підтримує поліморфізм, наслідування та статичну типізацію. У Java полегшений процес розробки об'єктно-орієнтованих програм, зокрема те, що усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста. Це вийшло реалізувати за рахунок того, що деякі дії програмістів виконує віртуальна машина.

Недоліком Java можна вважати те, що швидкодія програм значно поступається таким же, але написаних на сімействі C. Але, при необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Поточною версією є Java 12(актуальна на 1.03.2019).

Особливістю даної мови є те, що код спочатку транслюється в спеціальний байт-код, що має сумісність з різними платформами, потім спеціальна машина

(JVM) його виконує. Для кожної з платформ може бути своя реалізація віртуальної машини JVM, але кожна з них може виконувати один і той же код.

Для реалізації декстопної частини програмного модулю ідентифікації користувачів на основі біометричних даних було обрано мову програмування C#.

Синтаксис C# схожий до C++ і Java. Дана мова об’єктно-орієнтовна та підтримує поліморфізм, наслідування, перевантаження операторів. На від міну від C++, Delphi, Модула і Smalltalk – C#, виключає проблематичні моделі, що використовують при розробці.

Оскільки C# досить схожа з Java та C++, то їх можна порівняти.

<table>
<tr><th>Схожість з Java</th><th>Схожість з C++</th></tr>
<tr><td>компонентно-орієнтована</td><td rowspan="2">"перевантажені" оператори</td></tr>
<tr><td>інтерфейси</td></tr>
<tr><td rowspan="2">виключення</td><td rowspan="2">небезпечні арифметичні операції з плаваючою точкою</td></tr>
<tr><td>нитки (threads)</td></tr>
<tr><td rowspan="2">простір імен</td><td rowspan="3">деякі особливості синтаксиса</td></tr>
<tr><td>сильна (сувора) типізація</td></tr>
<tr><td>збір сміття</td></tr>
</table>

Рисунок 3.1 – Схожі риси C# з Java та C++

3.2 Вибір середовища розробки та обґрунтування його доцільності

Android Studio є офіційним інтегрованим середовищем розробки (IDE) для розробки додатків для Android, заснованим на IntelliJ IDEA. На додаток до потужного редактору та інструментам розробника IntelliJ, Android Studio пропонує ще більше можливостей, що підвищують продуктивність при створенні додатків для Android, таких як:

– гнучка система збірки на основі Gradle;

– швидкий і багатофункціональний емулятор;

– єдине середовище, у якому можна розробляти для всіх Android-пристроїв;

– миттєвий запуск для внесення змін у робочому додатку без створення нового APK;

– шаблони коду і інтеграція GitHub, щоб допомогти створювати спільні функції додатку та імпортувати приклад коду;

– обширні інструменти та засоби тестування;

– інструменти Lint для відстеження продуктивності, зручності використання, сумісності версій та інших проблем;

– підтримка C ++ та NDK;

– вбудована підтримка хмарної платформи Google, що дозволяє легко інтегрувати Google Cloud Messaging і App Engine [47].

Крім емулятора, SDK також включає безліч інших інструментальних засобів для налагодження та установки створюваних додатків. При розробці програми для Android за допомогою IDE Android Studio, багато інструментів командного рядка, що входять до складу SDK, вже використовуються при складанні та компіляції проекту.

Найбільш важливий з них цих інструментів – є емулятор мобільного пристрою, однак до складу SDK входять і інші інструменти для налагодження, упаковки та інсталяції додатків на емулятор.

– Головне вікно Android Studio складається з наступних логічних областей:

– панель інструментів, яка дозволяє виконувати широкий спектр дій, включаючи запуск додатку та інструментів Android;

– панель навігації, яка дозволяє переміщуватися по проекту і відкривати файли для редагування. Вона забезпечує компактніший вигляд структури, видимої у вікні проекту;

– у вікні редактора можна створювати та редагувати код. В залежності від поточного типу файлу редактор може змінитися;

– вікно панелі інструментів працює зовні вікна IDE і містить кнопки, які дозволяють розгорнути або згорнути окремі вікна інструментів.

– у вікні інструментів надається доступ до конкретних завдань (управління проектами, пошук, контроль версій і багато чому іншому).

– рядок стану відображає стан вашого проекту і саме IDE, а також будь-які попередження або повідомлення.

Стиль і форматування. При редагуванні Android Studio автоматично застосовує форматування і стилі, які зазначено в параметрах стилю коду, який можна налаштувати за допомогою мови програмування, включаючи вказівку умовностей для вкладок і відступів, прогалин, обгортання, фігурних дужок і порожніх рядків.

Основи управління версіями. Android Studio підтримує різні системи управління версіями (VCS), включаючи Git, GitHub, CVS, Mercurial, Subversion і Google Cloud Source Repositories. Після імпорту додатку в Android Studio використовуйте параметри меню Android Studio VCS, щоб включити підтримку VCS для системи управління необхідної версією, створити репозиторій, імпортувати нові файли в систему управління версіями та виконати інші операції.

Підтримка декількох APK. Дозволяє ефективно створювати кілька APK на основі щільності екрану або ABI. Наприклад, можна створювати окремі APK додатки для щільності екрану hdpi і mdpi, все ще розглядаючи їх як один варіант і дозволяючи їм ділитися параметрами APK, javac, dx і ProGuard.

Збірка сміття. При профілюванні використання пам'яті в Android Studio, можна одночасно ініціювати складання сміття і вивантажувати купу Java в моментальний знімок купи в двійковому файлі формату HPROF для Android. Засіб перегляду HPROF відображає класи, екземпляри кожного класу і дерево посилань, щоб допомогти відстежувати використання пам'яті та знаходити її витоки.

Перевірка коду. Кожного разу при компілюванні програми Android Studio автоматично запускає налаштовані перевірки Lint та інших IDE, щоб допомогти легко ідентифікувати та усунути проблеми зі структурною якістю коду. Інструмент

Lint перевіряє вихідні файли проекту Android на можливі помилки та покращує оптимізацію для правильності, безпеки, продуктивності, зручності використання, доступності та інтернаціоналізації. На додаток до перевірок Lint, Android Studio також виконує перевірки коду IntelliJ та перевіряє анотації для оптимізації робочого процесу кодування.

Анотації в Android Studio. Android Studio підтримує анотації для змінних, параметрів і значень, щоб допомогти зловити помилки, такі як виключення нульового покажчика та конфлікти типів ресурсів. Android SDK Manager упаковує бібліотеку Support-Annotations у репозиторий підтримки Android для використання з Android Studio. Android Studio перевіряє налаштовані анотації під час перевірки коду.

Журнальні повідомлення. При створенні та запуску додатку за допомогою Android Studio, можна переглядати повідомлення виходу adb і повідомлення журналу пристрою у вікні Logcat [48].

Для реалізації десктоп частини було обрано Visual Studio. Плюсами цього середовища програмування є: підтримка багатьох мов програмування,. Можливість створення Web-додатків на різних мовах, але об’єднувати їх у один проект. Оле недоліком цього є те, що кожну мову можна використовувати на різних веб-сторінках.

За рахунок того, що у Visual Studio включені шаблони стандартного коду (додавання веб-елемента управління, приєднання обробників подій і коригування форматування) ,то потребується менше коду для написання, зо значно зменшує роботу та прискорює розробку проекту.

У процесі написання коду Visual Studio форматує його автоматично, вставляючи всі необхідні відступи. Також зручно ще те, що текст виділяється кольором відповідно до його значення (класи – одним кольором, коментарі, код – іншими).

Visual Studio створена допомагати розробнику робити свою роботу набагато швидше. У середовище розробки включені такі функції: функція пошуку і заміни (яка дозволяє шукати за ключовими словами як в одному файлі, так і в усьому

проекті) і функції автоматичного додавання і видалення коментарів (яка може тимчасово приховувати блоки коду), дозволяють розробнику працювати швидко і ефективно.

Недоліком можна назвати те, що неможливо відстежити у відладчик у режимі ядра.

3.3 Розробка програмного модулю ідентифікації користувача за геометрією обличчя

Для реалізації програмної частини ідентифікації користувача за геометрією обличчя буде використано API Face Azure Cognitive Services, що надає алгоритми, які використовуються для виявлення, розпізнавання і аналізу людських обличь.

Щоб почати створювати додаток на Android потрібно підключити такі класи:

```
import java.io.*;
import java.lang.Object.*;
import android.app.*;
import android.content.*;
import android.net.*;
import android.os.*;
import android.view.*;
import android.graphics.*;
import android.widget.*;
import android.provider.*;
```

Далі створюється обробник подій на кнопці, який запускає нову дію, щоб дозволити користувачеві вибрати зображення, у даному випадку використовуючи камеру. Він відображає зображення в ImageView.

Далі потрібно підключитись до бібліотеки Face, як показано на рисунку 3.2.

```
20
21  dependencies {
22      implementation fileTree(dir: 'libs', include: ['*.jar'])
23      implementation 'com.android.support.constraint:constraint-layout:1.1.2'
24
25      implementation 'com.microsoft.projectoxford:face:1.4.3'
26
27      testImplementation 'junit:junit:4.12'
```

Рисунок 3.2 – Підключення до бібліотеки

Далі необхідно повернутись до MainActivity.java і додати import оператори:

```
import com.microsoft.projectoxford.face.*;
import com.microsoft.projectoxford.face.contract.*;
```

Наступні два методи будуть розпізнавати обличчя викликаючи метод faceClient.Face.DetectWithStreamAsync, яки забезпечує API-интерфейс Detect REST і повертає список екземплярів Face.

Потім дані відправляються на ПК, де щоб їх отримати необхідно визвати метод з параметром returnFaceId, зі встановленим значенням true.

```
IList<DetectedFace> faces = await faceClient.Face.DetectWithUrlAsync(imageUrl, true, false, null);
```

Щоб отримати дані про розташування маркерів (рис. 3.3), треба встановити для параметру returnFaceLandmarks значення true.

```
IList<DetectedFace> faces = await faceClient.Face.DetectWithUrlAsync(imageUrl, true, true, null);
```

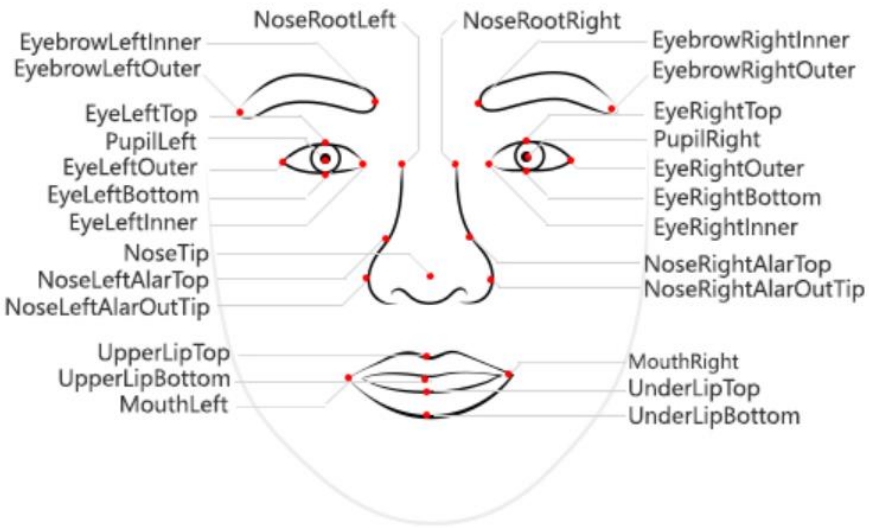


Рисунок 3.3 – Маркери визначення особливостей обличчя

Наступний код демонструє як отримуються розташування носа та зрачків:

```
foreach (var face in faces)
{
   var landmarks = face.FaceLandmarks;
   double noseX = landmarks.NoseTip.X;
   double noseY = landmarks.NoseTip.Y;
   double leftPupilX = landmarks.PupilLeft.X;
   double leftPupilY = landmarks.PupilLeft.Y;
   double rightPupilX = landmarks.PupilRight.X;
   double rightPupilY = landmarks.PupilRight.Y;
}
```

Взаємодію між програмам можна побачити на рисунку 3.4.

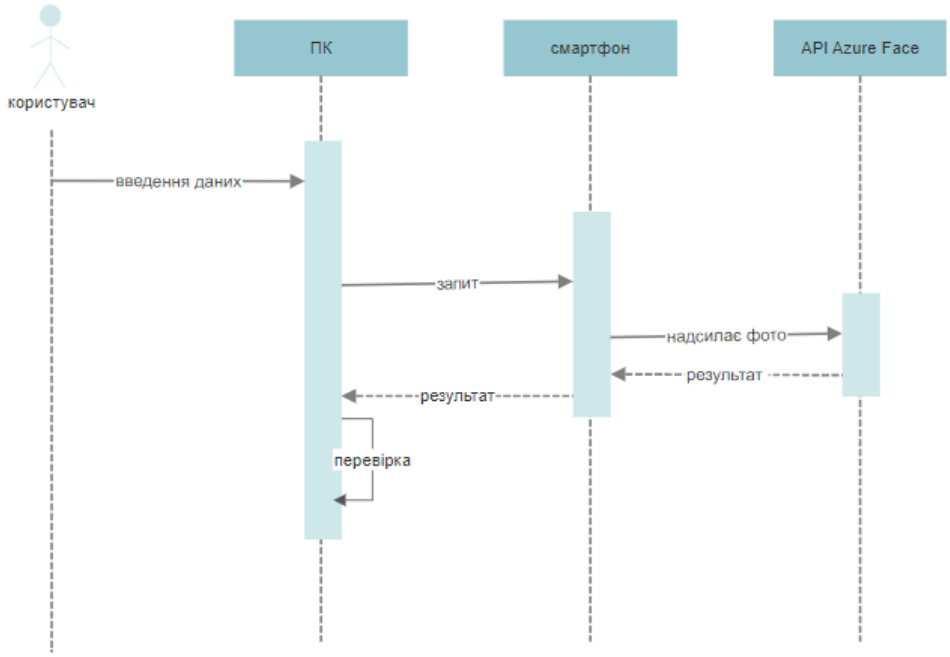


Рисунок 3.4 – Діаграма послідовності взаємодії ПК додатку з додатком на Android

3.4 Висновки до розділу

Результатом проведеного аналізу існуючих мов програмування оптимальною для розробки системи ідентифікації користувача за біометричними характеристиками було обрано Java та C#. Плюсами цих мов можна назвати те, що вони прості та універсальні. Об’єктно-орієнтовні та схожі між собою.

Середовищем розробки слугували Android Studio та Visual Studio. Переваги даних продукті: дані середовища розробки зручні у користуванні, мають перевірку коду, пошук по тексту та прибирання сміття. Також різний тип коду відображається різними кольорами.

Також було розроблено саму систему ідентифікації користувача за допомогою біометричних даних за допомогою алгоритму API Face Azure. Даний алгоритм підвищує точність вимірювань.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Технологічний аудит розробленої системи

Актуальною задачею розвитку інформаційних технологій на даному етапі є забезпечення надійного захисту інформації. Однією з важдивих та ще не вирішених проблем захисту інформації є ефективна ідентифікація користувача, що має доступ до конфіденційної інформації.

Все більше набувають популярності системи на основі методів біометричної ідентифікації для контролю доступу. Існує безліч біометричних пристроїв і відповідних програмних продуктів.

Для того щоб оцінити комерційний потенціал даної розробки необхідно провести технічнологічний аудит, що передбачає оцінювання наукового рівня розробки та визначення потенційних можливостей для її комерційного використання. Технологічний аудит був проведений експертним методом. Експертами аудиту були: Жуков С.О., Недовес О.О., Мовчан В.С.

Дані експерти є професіоналами у даній цій галузі досліджень та багато часу вивчають це питання.

Оцінювання комерційного потенціалу розробки було здійснено за 12-ю критеріями (додаток Г). Результати оцінювання експертами наведено у таблиці 4.1.

Таблиця 4.1 – Результати оцінювання наукового та комерційного потенціалу розробки.

Критерії	Прізвище, ініціали експерта		
	Жуков С.О.	Недовес О.О.	Мовчан В.С.
	Бали, виставлені експертами:		
0	*1*	*2*	*3*
1	2	3	3
2	3	4	3
3	3	3	4
4	4	2	2
5	3	3	2
6	2	4	2
7	3	3	4

продовження таблиці 4.1

0	*1*	*2*	*3*
8	4	3	3
9	4	4	3
10	3	4	4
11	4	4	4
12	4	4	4
Сума балів	$СБ_1 = 39$	$СБ_2 = 37$	$СБ_3 = 38$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{39+37+38}{3} = 38$		

Для визначення рівеня наукового та комерційного потенціалу системи ідентифыкації користувача за біометричними даниими, необхідно порівняти отриманий результат з даними, які наведено в таблиці 4.2.

Таблиця 4.2 – Рівні наукового та комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів оцнювання експертами складає 38 балів. За даними таблиці 4.2 можна зробити висновок, що дана розробка має рівень наукового та комерційного потенціалу „вище середнього”.

У таблиці 4.3 представлено зіставлення основних функціональних характеристик розробленої інформаційної системи ідентифікації користувача на основі біометричних даних з аналогічними показниками найближчого аналога наведемо в таблиці 4.3.

Таблиця 4.3– Порівняння функціональних характеристик з аналогом

Показники	Аналог	Розробка	
1. Швидкодія	85%	92%	1,08%
2. Точність оцінювання	70%	85%	1,21%
3. Адаптивність	68%	80%	1,18%

Шлях для реалізації розробки буде у вигляді розповсюдження ліцензійної угоди на певний термін дії з наданням технічної підтримки.

4.2 Розрахунок витрат на виконання даної розробки

Розрахунок витрат на розробку інформаційно системи ідентифікації користувачів на основі біометричних даних включає в себе:

– розрахунок витрат, на заробітню плату виконавцям;

– розрахунок загальних витрат на виконання (амортизація, витратні матеріали, інше);

– прогнозування витрат на впровадження даної роботи.

Розрахунок витрат, що відносяться до виконавців даного розділу роботи, включає в себе розрахунок таких статей:

1. Основна заробітна плата розробників (дослідників) $З_о$, які працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \ \text{[грн]}, \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника, грн.

У 2019 році величини окладів знаходиться в межах (4700…15000) грн. за місяць;

$Т_р$ – число робочих днів в місяці; $Т_р$ = 20 днів;

t – число робочих днів роботи розробника; t = 60 днів.

Зо = 5000/20*60 = 15000 грн

Розрахунки наведені у таблиці 4.4:

Таблиця 4.4 – Результати розрахунку витрат на виконання даної роботи

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.	Примітка
1. Керівник роботи	10000	500	3	1500	
2. Розробник	5000	250	60	15000	
Всього				$З_о$= 16500	

Додаткова заробітна плата $З_д$ виконавців розраховується як (10…12)% від величини основної заробітної плати, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о \ [грн] \qquad (4.2)$$

Для даного випадку:

$$З_д = 0{,}12 \cdot 16500 = 1980 \ (грн)$$

Нарахування на заробітну плату $Н_{зп}$ фахівців, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \ [грн], \qquad (4.3)$$

де $З_о$ – основна заробітна плата фахівців, грн.;

$З_д$ – додаткова заробітна плата фахівців, грн.;

β – ставка єдиного внеску на загальнообов’язкове державне соціальне страхування, яка дорівнює 22%.

Тоді:

$$Н_{зп} = (16500 + 1980) \cdot 0{,}22 = 4056 \ (грн.)$$

Наступним етапом необхідно обрахувати амортизацію обладнання, комп’ютерів, та приміщень, які використовуються під час виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому можна розрахувати за формулою:

де $Н_о$ – річна ставка орендної плати за рік для даного виду обладнання;

Ц – балансова вартість об’єкта, приміщень;

Т – термін корисного використання.

$$O = \frac{Ц \cdot H_O}{100} \cdot \frac{T}{12} \text{ [грн]}, \quad (4.4)$$

де H_o – річна ставка орендної плати за рік для даного виду обладнання;

Ц – балансова вартість об'єкта, приміщень;

Т – термін корисного використання.

Розрахунки амортизаційних нарахувань зведені у таблиці 4.5.

Таблиця 4.5 – Розрахунки амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Термін корисного використання, років	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
1.Комп'ютери, принтери, інша комп'ютерна техніка тощо	50000	8	1	520,8
2.Приміщення	80000	20	1	333,3
Всього				А = 854,2

Розрахунок витрат на допоміжні матеріали проводиться за формулою:

$$ВМ = \sum_1^n N_i Ц_i К_i \text{ [грн]}. \quad (4.5)$$

де n – кількість допоміжних матеріалів, N_i - кількість допоміжних матеріалів і-го виду, $Ц_i$ – покупна ціна допоміжних матеріалів і-го виду, грн, $К_i$ – коефіцієнт транспортних витрат ($К_i$ = 1,0).

Таблиця 4.6 - Витрати на допоміжні матеріали, що були використані для розробки ПЗ

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість комплектуючі, грн.
Папір	уп.	80	1	80
Ручка	шт.	10	2	20
Всього з урахуванням транспортних витрат				100

Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}. , \quad (4.6)$$

де В – вартість 1 кВт-год. Електроенергії (В ≈ 8 грн./кВт), П – установлена потужність обладнання (П = 0,9 кВт), Ф – фактична кількість годин роботи обладнання, орієнтовано 60 днів по 8 годин (Ф = 200 год), $К_п$ – коефіцієнт використання потужності ($К_п < 1$, $К_п = 0,7$).

Тоді витрати на електроенергію складуть:

$$В_е = 8 \cdot 0,9 \cdot 480 \cdot 0,7 = 2419,2 \text{ (грн)}$$

Інші витрати $В_{ін}$ можна прийняти як (120)% від суми основної заробітної плати фахівців, які виконували дану роботу, тобто:

$$В_{ін} = 1,2 \cdot 16500 = 19800 \text{ (грн)}$$

Сума всіх попередніх статей витрат дає витрати на виконання даного етапу роботи магістрантом – В.

Для даного випадку:

В = 16500 + 2419,2 + 1714,9 + 854,2 + 100 + 19800 = 41 388 (грн.)

Далі потрібно розрахувати загальні витрати на виконання даної роботи всіма виконавцями. Загальна вартість всієї роботи визначається за $В_{заг}$ формулою:

$$В_{заг} = \frac{В}{\alpha} \text{ [грн]}, \qquad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях. Для даного випадку $\alpha = 0,95$.

Тоді:

$$В_{заг} = \frac{41\,388}{0,95} = 43\,566 \text{ (грн)},$$

Розрахунок загальні витрати на виконання та можливе впровадження результатів виконаної роботи:

$$ЗВ = \frac{В_{заг}}{\beta} \text{ [грн]}, \qquad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0,1$;

- на стадії технічного проектування, то $\beta \approx 0{,}2$;

- на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;

- на стадії розробки технологій, то $\beta \approx 0{,}4$;

- на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;

- на стадії розробки промислового зразка, $\beta \approx 0{,}7$;

- на стадії впровадження, то $\beta \approx 0{,}85$. Для даного випадку $\beta \approx 0{,}85$.

Отриманий результат:

$$ЗВ = \frac{43566}{0{,}85} = 51\ 254{,}8(\text{грн}).$$

Тобто, прогнозовані витрати на виконання та можливе впровадження результатів даної роботи складають $51\ 254{,}8$ (грн).

4.3 Прогнозування комерційних ефектів від можливої реалізації результатів розробки

Для того щоб отримувати дохід від реалізованої системи і продавати її дорожче аналогів треба зробити прогнузування комерційних ефектів.

Проаналізувавши ринок України можна сказати що потенційних користувачів розробки приблизно 2200 шт. Але тих, якщо реаль но дивитись на ситуацію, то попит складе 15-25 позицій.

Порівнявши ціни конкурентів, вартість програми з подібним функціоналом приблизно 50 000 гривень. Але розроблена програма має кращі тенічні параметри та функціональні, що дозволяє реалізовувати її приблизно на 10-20% дорожче, при цьому збільшуючи попит.

Можна припустити, що інформаційна система ідентифікації корситувачів на основі біометричних даних буде актульною на ринку три роки після реалізації.

З 1 січня 2020 року уже можна презунтавати ринку дану програму, а результат продажу будуть виявлятись протягом 2020-го, 2021-го та 2022-го років. Прогноз попиту на розробку складає по роках:

- 1-й рік після впровадження (2020 р.) – приблизно 10 шт.;
- 2-й рік після впровадження (2021 р.) – приблизно 15 шт.;
- 3-й рік після впровадження (2022 р.) – приблизно 12 шт.

На 4-й рік (2023 р) отримання прибутків не планується, оскільки є висока ймовірність розробки нових систем, які будуть досконалішими.

Для того щоб розрахувати очікуване збільшення чистого прибутку $\Delta П_i$, що може отримати потенційний інвестор від впровадження даної програми, коли не має можливоті оцінити зростання чистого прибутку підприємства від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року після впровадження, необхідно скористатись формулою:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.9)$$

де, $\Delta Ц_o$ – покращення основного оціночного показника від впровадження результатів розробки у даному році. Для даного випадку $\Delta Ц_o$ = 18 тис. грн.;

N – основний кількісний показник, який визначає обсяг діяльності у даному році до впровадження результатів розробки;

ΔN – покращення основного кількісного показника від впровадження результатів розробки;

$Ц_o$ – основний оціночний показник, який визначає обсяг діяльності у даному році після впровадження результатів розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%;

λ = 0,8333. Ставка податку на додану вартість встановлена на рівні 17%, а коефіцієнт λ = 0,8547;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = 0,2...0,3; обрано ρ = 0,2;

υ – ставка податку на прибуток; υ = 18%.

Орієнтовно: реалізація продукції до виходу на ринок розробки складала 1 шт., а її ціна – 55000грн.

Збільшення чистого прибутку $\Delta П_i$ протягом першого року складе:

$$\Delta П_1 = [18000 \cdot 1 + 55000 \cdot 10] \cdot 0{,}8333 \cdot 0{,}2 \cdot \left(1 - \frac{18}{100}\right) = 77623{,}6 \text{ (грн)}$$

Збільшення чистого прибутку $\Delta П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів розробки) складе:

$$\Delta П_2 = [18000 \cdot 1 + 55000 \cdot (10 + 15)] \cdot 0{,}8333 \cdot 0{,}2 \cdot \left(1 - \frac{18}{100}\right) = 190369 \text{ (грн)}$$

Збільшення чистого прибутку $\Delta П_i$ протягом третього року (відносно базового року, тобто року до впровадження результатів розробки) складе:

$$\Delta П_3 = [18000 \cdot 1 + 55000 \cdot (10 + 15 + 12)] \cdot 0{,}8333 \cdot 0{,}2 \cdot \left(1 - \frac{18}{100}\right) = 280565{,}4 \text{ (грн)}$$

Далі необхідно розрахувати теперішню вартість інвестицій PV, що можуть бути вкладені в дану розробку. Можна припустити, що загальні витрати ЗВ на виконання та впровадження результатів роботи (або теперішня вартість інвестицій PV) дорівнює 51 254,8 грн. Результати вкладених у наукову розробку інвестицій почнуть виявлятися через три роки.

Такі результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 77623,6 грн відносно базового року, у другому році – збільшення чистого прибутку на 190369 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 280565,4 грн (відносно базового року). На рисунку 4.1, наведений рух платежів (інвестицій та додаткових прибутків).

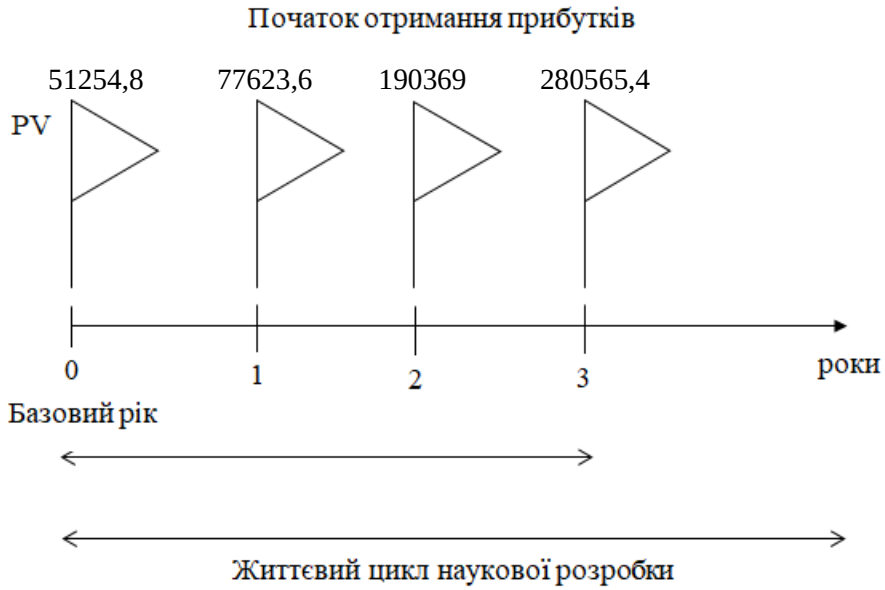


Рисунок 4.1 – Вісь часу з фіксацією платежів (у грн.), що мають місце під час розробки та впровадження результатів нашої роботи.

Далі необхідно розрахувати абсолютний ефект вкладених інвестицій $E_{абс}$. :

$$E_{абс} = (ПП – PV) \text{ [грн]}, \quad (4.10)$$

де ПП – приведена вартість всіх чистих прибутків від можливої реалізації результатів розробки, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою 5.11:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої наукової роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні.

Для України цей показник приймемо на рівні $\tau = 0{,}1$;

t – період часу (в роках) від моменту отримання прибутків до точки „0”.

Якщо $E_{абс} \leq 0$, то результат від впровадження розробки буде збитковим і вкладати кошти в проведення досліджень ніхто не буде.

Якщо $E_{абс} > 0$, то результат від впровадження розробки може принести прибуток і вкладати кошти в дану розробку в принципі можна.

Тоді приведена вартість всіх чистих прибутків ПП від можливої реалізації результатів нашої розробки складе:

$$ПП = \frac{77623,6}{(1+0,1)^1} + \frac{190369}{(1+0,1)^2} + \frac{280565,4}{(1+0,1)^3} = 438689,6(грн),$$

Абсолютний ефект від впровадження результатів нашої розробки протягом 3-х років складе:

$$E_{абс} = 438\,689,6 - 51\,254,8 = 387\,434,8$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів розробки може бути доцільним.

Щоб зацікавити фінансування розробки від інвестору необхідно визначити чи ефективність вкладень буде більше чим первний рівень.

Для цього розраховується відносна ефективність $E_в$ інвестицій, вкладених у розробку. Для цього необхідно скористатись формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.12)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = 51 254,8 грн;

$E_{абс}$ = 387 434,8 грн.

$T_ж$ – життєвий цикл наукової розробки, роки.

Для нашого випадку:

$$Ев = \sqrt[3]{1 + \frac{387\,434{,}8}{51\,254{,}8}} - 1 = \ 0{,}7104 \text{ або } 71{,}04\%.$$

Далі потрібно визначити ту мінімальну дохідність, нижче за яку кошти в розробку проекту вкладатися не будуть.

У загальному вигляді мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \qquad (4.13)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = (0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5), але може бути і значно більше.

Для нашого випадку отримаємо:

$$\tau_{мін} = 0{,}2 + 0{,}05 = 0{,}25 \text{ або } \tau_{мін} = 25\%.$$

Оскільки величина $E_в = 71{,}04\% > \tau_{мін} = 25\%$, то потенційний інвестор може бути зацікавлений у фінансуванні даної розробки.

Далі потрібно розраховати термін окупності вкладених у реалізацію проекту інвестицій:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.14)$$

Для нашого випадку термін окупності інвестицій $T_{ок}$, складе:

$$T_{ОК} = \frac{1}{0{,}7104} \approx 1{,}4 \text{ роки}.$$

Оскільки $T_{ок} < (3…5)$ років, то фінансування нашої розробки є доцільним.

4.4 Висновки до розділу

На основі зроблених підрахунків в економічній частині магістерської кваліфікаційної роботи досягнуті наступні результати:

– рівень комерційного потенціалу розробки – вище середнього;

– витрати на розробку склали 51 254,8 грн, що в межах норми, та не перевищує витрати у ТЗ;

– порівняно з витратами прибуток від розробки за три роки буде 438689,6 грн.;

– термін окупності даної системи складе 1,4 роки.

Розробка системи ідентифікації користувачів за біометричними даними є доцільною з комерційної точки зору.

ВИСНОВКИ

У результаті аналізу загроз інформаційній безпеці, та огляду сучасних методів ідентифікації користувачів комп'ютерних систем, можна зробити висновок, що у міру розвитку технологій все більш актуальним буде саме вживання систем біометричної ідентифікації, що дозволить значно підвищити рівень надійності систем ідентифікації.

Найпоширенішою біометричною технологією є ідентифікація за відбитками пальців. А найперспективнішою на майбутній рік – ідентифікація за геометрію обличчя. Отже, було обрано ці методи ідентифікації.

Проаналізувавши усі методи ідентифікації зроблено висновок, що сьогодні частка систем розпізнавання за відбитками пальців становить майже половину від усіх використовуваних у світі біометричних технологій, через свою точність, і за прогнозами обсяг продажів на наступний рік системи ідентифікації за геометрію обличчя складе конкуренцію. Отже, було обрано ці методи.

У результаті проведеного аналізу технологій розмежування доступу було виявлено низку характерних особливостей, переваг і недоліків існуючих методів управління доступом. Визначено, що одними з найбільш зручним та актуальним є використання технології рольового розмежування доступу.

З безлічі доступних мов програмування, за зезультатом проведеного аналізу, оптимальною для розробки системи ідентифікації користувача за біометричними характеристиками було обрано Java та C#. Ці мови прості та універсальні, обьектно-орієнтовні та подібні між собою.

Розробка системи здійснювалась у Android Studio та Visual Studio. Дані середовища зручні у користуванні, мають перевірку коду, пошук по тексту та прибирання сміття. Також різний тип коду відображається різними кольорами.

У економічній частині кваліфікаційної роботи було оцінено комерційниї потенціал розробки на основі розрахунків технологічного аудиту, витрат на виконання і реалізацію, та прогнозування комерційних ефектів від можливої

реалізації результатів розробки. Зроблено висновок, що розробка системи є доцільною.

Результатом даної магістерської роботи є розроблений програмний модуль ідентифікації користувача за біометричними характеристиками, а саме відбитками пальців та геометрією обличчя. Для роботи з інформаційною системою було створено інструкцію користувача.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Галатенко В.А. Основы информационной безопасности: учебное пособие / В. А. Галатенко; под ред. академика РАН В.Б. Бетелина, 4-е изд. – М.: Интернет-Университет Информационных технологий; БИНОМ. Лаборатория знаний, 2008. – 205 с.

2. Джхунян В.Л. Электронная идентификация / В.Л. Джхунян, В.Ф. Шаньгин. – М.: NT Press, 2004. – 695 с.

3. Голубев Г.А. Современное состояние и перспективы развития биометрических технологий / Г.А. Голубев, Б.А. Габриелян // Нейрокомпьютеры: разработка, применение. – 2004. – № 10. – С. 39-46.Кара-Мурза, С. Г. Манипуляция сознанием. - М.: Эксмо, 2006.-832 с.

4. Волошина В.А Біометрична ідентифікація користувачів Інформаційно-комп'ютерних систем / В.А. Волошина, С.О. Жуков [Електронний ресурс] // Інформаційні технології і автоматизація – 2019 : зб. доп. XII Міжнар. наук.-практ. конф., Одеса, 17–18 жовт. 2019. – Режим доступу до ресурсу: https://card-file.onaft.edu.ua/bitstream/123456789/10174/1/Information_technologies_and_automation_2019_%d0%a01.pdf

5. Кормич Б.А. Організаційно-правові засади політики інформаційної безпеки України [Текст]: монографія / Б.А.Кормич.– Одеса: Юридична література, 2007.– 471с.

6. Воронова В.А. Системы контроля и управления доступом / В.А. Воронова, В.А. Тихонов. – М.: «Горячая линия – Телеком», 2010. – 272 с.

7. Даклин Пол. Простые советы по более разумному выбору и использованию паролей / Пол Даклин. [Електронний ресурс]. – Режим доступу до ресурсу: http:// www.infosecurity.ru/_gazeta/content/060525/article01.shtml.

8. Безмалый В. Парольная защита: прошлое, насто- ящее, будущее / В. Безмалый // Журнал «КомпьютерПресс». – 2008. – №9. [Електронний ресурс]. – Режим дос- тупу до ресурсу: http://www.compress.ru/article.aspx?Id= 20509&iid=901.

9. Кухарев Г.А. Биометрические системы: методы и средства идентификации личности человека / Г.А. Кухарев. – СПб.: Политехника, 2001. – 240 с.

10. Горбулін В. П. Інформаційні операції та безпека суспільства: загрози, протидія, моделювання: монографія / В. П. Горбулін, О. Г. Додонов, Д. В. Ланде. – К.: Інтертехнологія, 2009. – 164 с.

11. Кухарев Г.А. Биометрические системы: Методы и средства идентификации личности человека / Г.А. Кухарев. – СПб.: Политехника, 2001. – 240 с.

12. Біометричні системи безпеки [Електронний ресурс]. – Режим доступу до ресурсу: http://uadoc.zavantag.com/text/23710/index-1.html.

13. Барабанова М. І. Інформаційні технології: відкриті системи, мережі, безпека в системах та мережах / М. І. Барабанова, В. І. Кіяев. // Навчальний посібник. – СПб. – 2010. – 267 с.

14. Задорожный В. В. Идентификация по отпечаткам пальцев / В. В. Задорожный. // PC Magazine. – 2004. – №1.

15. Вежневец В. Обнаружение и локализация лица на изображении / В. Вежневец, А. Дегтярева. // Компьютерная графика и мультимедиа. – 2003. – №1(3)

16. Коновалов Д.Н. Технология защиты информации на основе идентификации голоса / Д.Н. Коновалов, А.Г. Бояров // [Електронний ресурс]. – Режим доступу до ресурсу: http://www.fact.ru/archive/07/voice.shtml.

17. Шарипов Р.Р. Идентификация и аутентификация пользователей по клавиатурному почерку / Р.Р. Шарипов // Электронное приборостроение: Научно практический сборник. – Казань: ЗАО «Новое знание», 2005. – Вып. 3(44).

18. Завгородний В.И. Комплексная защита информа- ции в компьютерных системах: учебное пособие / В.И. Завгородний. – М.: Логос; ПБОЮЛ Н.А. Егоров, 2001. – 264 с.

19. Шрамко В.Н. Комбинированные системы иден- тификации и аутентификации / В.Н. Шрамко // PCWeek/RE. – 2004. – №45.

20. Десятчиков А.А. Синхронная биометрическая многофакторная идентификация / А.А. Десятчиков, А.Б. Мурынин, Ю.П. Тресков, В.Я. Чучупал // Труды ИСА РАН. Динами- ка неоднородных систем. – М.: УРСС. – 2005. – Вып. 9 (1). – С. 188-194.

21. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа [Текст] / А.Ю. Щеглов. – СПб.: Наука и техника, 2004. – 384 с.

22. Купін А. І. Перспективи застосування мультимодальних інформаційних технологій у задачах біометричного розпізнавання об’єктів / А. І. Купін, Ю. О. Кумченко // Гірничий вісник: Науково-технічний збірник. – 2014. – Вип. 97. – С. 165–168.

23. Хорошко В.А. Методы и средства защиты информации / В.А. Хорошко, А.А. Чекатков. – К.: Изд-во Юниор, 2003. – 504 с.

24. Романец Ю.В. Защита информации в компьютерных системах и сетях [Текст] / Ю.В. Романец, П.А. Ти- мофеев, В.Ф. Шаньгин; Под ред. В.Ф. Шаньгина. – 2-е изд., перераб. и доп. – М.: Радио и связь, 2001. – 376 с.

25. Конахович Г.Ф. Захист інформації в мережах передачі даних: підручник / Г.Ф. Конахович, О.Г. Корченко, О.К. Юдін. – К.: Видавництво ТОВ НВП «ІНТЕРСЕР- ВІС», 2009. – 714 с.

26. Єсін В.І, Кузнецов О.О., Сорока Л.С. Безпека інформаційних систем і технологій. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

27. Єсіна М.В., Горбенко І.Д. Багатофакторна автентифікація: використання механізмів двофакторної автентифікації для захисту від несанкціонованого доступу // Компьютерное моделирование в наукоемких технологиях (КМНТ-2014): Труды научнотехнической конференции с международным участием, 28-31 мая 2014 г. – Харьков: Харьковский национальный университет им. В.Н. Каразина, 2014. – С. 159–162. Симонс Г.Д. Обзор методов аутентификации информации. – Москва: ТИИЭР, 1988. – Т. 76, №5. – С. 105–125. Столлингс В.

28. Десятчиков А.А. Об объединении дистанционных биометрических методов распознавания человека / А.А. Десятчиков, В.В. Лобанцов, И.А. Матвеев,

А.Б. Мурынин // Современный экстремизм в Российской Федерации: особенности проявления и средства противодействия: Материалы всероссийской научно-практической конференции в Академии Управления МВД России. – М.: Академия управления МВД РФ, 2006. – С. 374-379.

29. Тархов Андрей Entrust IdentityGuard. Система многофакторной аутентификации [Електронний ресурс]. – Режим доступу: http://www.rnbo.ru/press-center/news228.php

30. Девянин П.Н. Модели безопасности компьютерных систем / П.Н. Девянин. – М.: Издательский центр «Академия», 2005. – 144 с.

31. Кузовкин К. Удаленный доступ к информационным ресурсам. Аутентификация [Електронний ресурс]/ Кузовкин К.// Директор информационной службы – 2003. - №9 . — Режим доступу до журн. : http://www.i-teco.ru/article33.html

32. Семенов С.Г. Методика настройки параметров распределения доступа и защиты информации в компьютерных системах критического применения / С.Г. Семенов // Системи озброєння і військова техніка. – Х.: ХУ ПС. – 2012. – Вип. 4(32). – С. 153-158.

33. Семенов С.Г. Методы и средства распределения доступа и защиты данных в компьютеризированных информационных управляющих системах критического применения / С.Г. Семенов. – Х.:НТУ «ХПИ», 2013. – 360 с.

34. Порошин С.М. Разработка и исследования математической модели компьютеризированной информационно-измерительной управляющей системы критического применения с учетом фактора внешних воздействий / С.М. Порошин, С.Г. Семенов // Системи обробки інформації. – Х.: ХУ ПС, 2013. – Вип. 2(110). – С. 208-210.

35. Барсуков В. С. Биоключ – путь к безопасности [Електронний ресурс] / Вячеслав Сергеевич Барсуков // Специальная техника. – 2007. – Режим доступу до ресурсу: http://www.vashdom.ru/articles/st_14.htm.

36. Bell D.E. Unified Exposition and Multics Interpretation MITRE Corporation / D.E. Bell, L.J. LaPadula // Secure Computer System: (1976). [Електронний ресурс]. – Режим доступа к ресурсу: http://csrc.nist.gov/ publications/history/bell76.pdf

37. Biba K. Integrity Considerations for Secure Computer Systems / K. Biba // Technical Report MTR-3153, MITRE Corporation, Bedford, MA (Apr. 1977).

38. Кумченко Ю. О. Інформаційна технологія авторизації користувача вебсайту з використанням біометричної характеристики / Ю. О. Кумченко, Р. С. Павлюк // VIII Всеукраїнська науково-практична WEB конференція аспірантів, студентів та молодих вчених «Комп'ютерні інтелектуальні системи та мережі»: 24–26 березня 2015 р.: матер. – Кривий Ріг, 2015. – С. 72–74

39. Кумченко Ю. О. Використання методів біометричної автентифікації користувача в сучасному ПЗ / Ю. О. Кумченко, А. І. Купін // Сталий розвиток промисловості та суспільства: матер. – Кривий Ріг, 2015. – Т. 1. – С. 280–281. 88. Інформаційна технологія ідентифікації персоналу на основі комплексу біометричних параметрів: звіт про НДР (закл.) 07.15 / ДВНЗ «Криворізький національний університет»; керівн. А. І. Купін; викон. Ю. О. Кумченко. – Кривий Ріг, 2015. – 34 с. – Інв. № 0715U007022.

40. Кумченко Ю. О. Інформаційна технологія ідентифікації студентів з використанням статичної біометричної характеристики для проходження електронного тестування / Ю. О. Кумченко, Р. С. Павлюк // IX Всеукраїнська науково-практична WEB конференція аспірантів, студентів та молодих вчених «Комп'ютерні інтелектуальні системи та мережі»: 22–24 березня 2016 р.: матер. – 128 Кривий Ріг, 2016. – С. 100–102.

41. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа. СПб: Наука и техника, 2004 г.- 384с.

42. Андреев О.О. Интеграция моделей логического разграничения доступа, описанных на специализированном языке / О.О. Андреев // Информационные технологии. – М.: Новые технологии. – 2009. – № 12. – С. 29-33.

43. Закон України «Про інформацію» [Редакція від 25.06.2016 р.]; [Електронний ресурс] // Сайт Верховної Ради України. – Режим доступу: http://zakon4.rada.gov.ua/laws/show/2657-12.

44. ДСТУ 3396.2-97 Захист інформації. Технічний захист інформації. Терміни та визначення. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.dstszi. gov.ua/dstszi/control/uk/publish/printable_article?art_id=38934.

45. Лапонина О.Р. Анализ возможностей языка XACML по управлению доступом / О.Р. Лапонина // Сборник трудов V Международной научно-практической конференции "Современные информационные технологии и ИТ-образование" (8–10 ноября 2010 г.). – М: Московский госуд. университет им. М.В. Ломоносова. – С. 473-484.

46. НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу. [Електронний ресурс]. – Режим доступу до ресурсу: http://tzi.com.ua/nd-tz-2.5-004-99.html.

47. Семенов С.Г. Методика настройки параметров распределения доступа и защиты информации в компьютерных системах критического применения / С.Г. Семенов // Системи озброєння і військова техніка. – Х.: ХУПС, 2012. – № 4(32). – С. 153-158.

48. Кормич Б.А. Організаційно-правові засади політики інформаційної безпеки України [Текст]: монографія / Б.А.Кормич.– Одеса: Юридична література, 2007.– 471с.

ДОДАТКИ

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики

ЗАТВЕРДЖУЮ
Завідувач кафедри САКМІГ
_____ д. т. н., проф. В.Б. Мокін
(підпис)
"__" _____ 2019

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу

ІНФОРМАЦІЙНА СИСТЕМА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ БІОМЕТРИЧНИХ ДАНИХ

08-53.МКР.011.01.000 ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доц.
_____С.О. Жуков
(підпис)
"__" _____ 2019 р.
Розробила студентка гр. ІСТ-18м
_____В.А. Волошина
(підпис)
"__" _____ 2019 р.

Вінниця 2019

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 201_ р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САКМІГ від «__» _____ 201_ р.

2. Джерела розробки

1) Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа

2) Системы управления доступом к информационным ресурсам [Електронний ресурс]

3) НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. [Електронний ресурс].

4) Технологии построения эффективной системы управления доступом к информационным ресурсам [Електронний ресурс]

3. Мета і призначення роботи

Розробка інформаційної системи ідентифікації користувачів на основі біометричних даних.

4. Вихідні дані для проведення робіт

1) Відбиток пальця

2) Геометрія обличчя

5. Методи дослідження

1) Розмежування доступу на основі ролей

2) Azure Face API

6. Етапи роботи і терміни їх виконання

1) Огляд технічних та програмних засобів..……..__. – __

Розробка інформаційної системи ідентифікації користувачів на основі біометричних даних……..……..__. – __

2) Програмна реалізація інформаційної ідентифікації користувачів на основі біометричних даних ……..…….……..__. – __

7. Очікувані результати та порядок реалізації

Розробка інформаційної ідентифікації користувачів на основі біометричних даних, яка на відмінну від існуючих, забезпечує кращий захист даних.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист…….__.__.201_ р.

Початок розробки «___» _____ 201_ р.

Граничні терміни виконання МКР «___» _____201_ р.

Розробив студент групи ІСТ-18м _____ Волошина В.А.

Додаток Б. Інструкція користувача

Робота з програмою де вбудований модуль ідентифікації користувача за біометричними починається з реєстрації у системі. Для того, зареєструватись необхідно звернутись до користувача з правами адміністратора щодо даного питання. Під час реєстрації у базу даних вносять дані про користувача, його логін, пароль та за допомогою камери на смартфоні вносяться дані про геометрію обличчя.

Для першого наступного входу у програму користувачу достатньо запустити модуль на комп’ютері та натиснути кнопку увійти(рис Б.1). Потім потрібно ввести логін та пароль.

Рисунок Б.1 – Стартове вікно програми

Далі необхідно запустити додаток на смартфоні, та натиснути кнопку «Увійти» (рис. Б.2).

Рисунок Б.2 – Стартове вікно мобільного додатку

Наступною дією є ввід логіна та паролю (рис Б.3).

Рисунок Б.3 – Вікно вводу логіну та паролю

Далі треба обрати зручний спосіб ідентифікації. Якщо натиснути увійти за допомого відбитку пальців, то додаток запропонує піднести палець до сканера (рис Б.4).

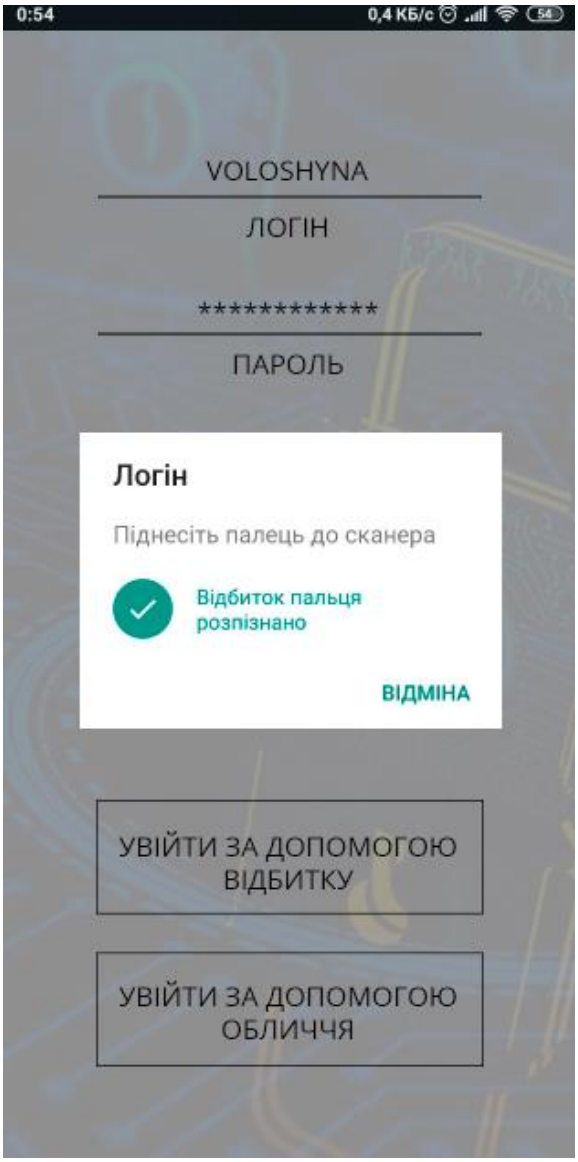


Рисунок Б.4 – Вікно ідентифікації за відбитком

Якщо ж обрати увійти за допомогою обличчя необхідно дозволити використання камери (рис Б.5). Після даних дій при успішній ідентифікації захищеною програмою можна користуватись.

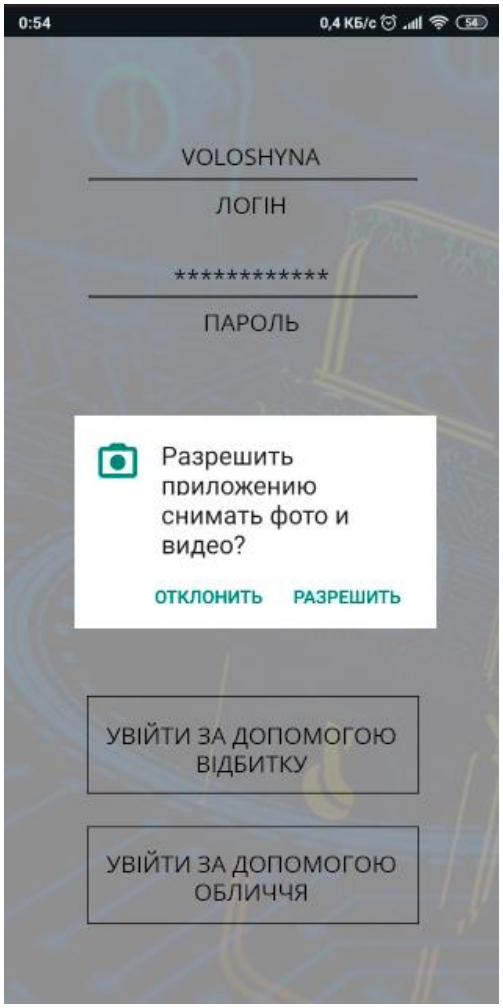


Рисунок Б.4 – Вікно ідентифікації за геометрією обличчя

Після даних дій при успішній ідентифікації захищеною програмою можна користуватись.

Додаток В. Лістинг програми

```
CREATE DATABASE UsrAuthorization;
CREATE TABLE UsrAuthorization.secured_users_data (
  id INT NOT NULL,
  entity_id INT NULL,
  first_name VARCHAR(255) NULL,
  last_name VARCHAR(255) NULL,
  entity_role VARCHAR(255) NULL,
  PRIMARY KEY (id),
  UNIQUE INDEX id_UNIQUE (id ASC),
  UNIQUE INDEX entity_id_UNIQUE (entity_id ASC));
CREATE TABLE UsrAuthorization.encrypted_fingerprints (
  id INT NOT NULL,
  entity_id VARCHAR(255) NULL,
  entity_encrypted_fingerprint VARCHAR(255) NULL,
  PRIMARY KEY (id),
  UNIQUE INDEX id_UNIQUE (id ASC),
  UNIQUE INDEX entity_id_UNIQUE (entity_id ASC),
  UNIQUE INDEX entity_encrypted_fingerprint_UNIQUE (entity_encrypted_fingerprint ASC));
CREATE TABLE UsrAuthorization.users_access_control (
  id INT NOT NULL,
  entity_id INT NULL,
  entity_access_level INT NULL,
  PRIMARY KEY (id),
  UNIQUE INDEX id_UNIQUE (id ASC),
  UNIQUE INDEX entity_id_UNIQUE (entity_id ASC));
CREATE TABLE UsrAuthorization.denied_access_blacklist (
  id INT NOT NULL,
  entity_id INT NULL,
  entity_encrypted_fingerprint VARCHAR(255) NULL,
  privacy_leak_risk_level INT NULL,
  access_denied_reason VARCHAR(255) NULL,
  PRIMARY KEY (id),
  UNIQUE INDEX id_UNIQUE (id ASC),
  UNIQUE INDEX entity_id_UNIQUE (entity_id ASC),
  UNIQUE INDEX entity_encrypted_fingerprint_UNIQUE (entity_encrypted_fingerprint ASC));

package com.auth.fingerprintauth;

import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.os.Build;
import android.security.keystore.KeyGenParameterSpec;
import android.security.keystore.KeyPermanentlyInvalidatedException;
import android.security.keystore.KeyProperties;
import android.support.annotation.Nullable;
import android.support.v4.hardware.fingerprint.FingerprintManagerCompat;
```

```
import android.util.Base64;

import java.io.IOException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.KeyPairGenerator;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.MGF1ParameterSpec;
import java.security.spec.X509EncodedKeySpec;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.OAEPParameterSpec;
import javax.crypto.spec.PSource;


@TargetApi(Build.VERSION_CODES.M)
public final class CryptoUtils {
   private static final String TAG = CryptoUtils.class.getSimpleName();

   private static final String KEY_ALIAS = "key_for_pin";
   private static final String KEY_STORE = "AndroidKeyStore";
   private static final String TRANSFORMATION = "RSA/ECB/OAEPWithSHA-
256AndMGF1Padding";

   private static KeyStore sKeyStore;
   private static KeyPairGenerator sKeyPairGenerator;
   private static Cipher sCipher;

   private CryptoUtils() {
   }

   public static String encode(String inputString) {
      try {
         if (prepare() && initCipher(Cipher.ENCRYPT_MODE)) {
            byte[] bytes = sCipher.doFinal(inputString.getBytes());
            return Base64.encodeToString(bytes, Base64.NO_WRAP);
```

```
            }
        } catch (IllegalBlockSizeException | BadPaddingException exception) {
            exception.printStackTrace();
        }
        return null;
    }


    public static String decode(String encodedString, Cipher cipher) {
        try {
            byte[] bytes = Base64.decode(encodedString, Base64.NO_WRAP);
            return new String(cipher.doFinal(bytes));
        } catch (IllegalBlockSizeException | BadPaddingException exception) {
            exception.printStackTrace();
        }
        return null;
    }

    private static boolean prepare() {
        return getKeyStore() && getCipher() && getKey();
    }


    private static boolean getKeyStore() {
        try {
            sKeyStore = KeyStore.getInstance(KEY_STORE);
            sKeyStore.load(null);
            return true;
        } catch (KeyStoreException | IOException | NoSuchAlgorithmException |
CertificateException e) {
            e.printStackTrace();
        }
        return false;
    }


    @TargetApi(Build.VERSION_CODES.M)
    private static boolean getKeyPairGenerator() {
        try {
            sKeyPairGenerator =
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_RSA, KEY_STORE);
            return true;
        } catch (NoSuchAlgorithmException | NoSuchProviderException e) {
            e.printStackTrace();
        }
        return false;
    }
```

```
    @SuppressLint("GetInstance")
    private static boolean getCipher() {
        try {
            sCipher = Cipher.getInstance(TRANSFORMATION);
            return true;
        } catch (NoSuchAlgorithmException | NoSuchPaddingException e) {
            e.printStackTrace();
        }
        return false;
    }

    private static boolean getKey() {
        try {
            return sKeyStore.containsAlias(KEY_ALIAS) || generateNewKey();
        } catch (KeyStoreException e) {
            e.printStackTrace();
        }
        return false;

    }


    @TargetApi(Build.VERSION_CODES.M)
    private static boolean generateNewKey() {

        if (getKeyPairGenerator()) {

            try {
                sKeyPairGenerator.initialize(
                        new KeyGenParameterSpec.Builder(KEY_ALIAS,
KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT)
                                .setDigests(KeyProperties.DIGEST_SHA256, KeyProperties.DIGEST_SHA512)
                                .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_RSA_OAEP)
                                .setUserAuthenticationRequired(true)
                                .build());
                sKeyPairGenerator.generateKeyPair();
                return true;
            } catch (InvalidAlgorithmParameterException e) {
                e.printStackTrace();
            }
        }
        return false;
    }


    private static boolean initCipher(int mode) {
        try {
```

```
            sKeyStore.load(null);

            switch (mode) {
                case Cipher.ENCRYPT_MODE:
                    initEncodeCipher(mode);
                    break;

                case Cipher.DECRYPT_MODE:
                    initDecodeCipher(mode);
                    break;
                default:
                    return false; //this cipher is only for encode\decode
            }
            return true;

        } catch (KeyPermanentlyInvalidatedException exception) {
            deleteInvalidKey();

        } catch (KeyStoreException | CertificateException | UnrecoverableKeyException |
IOException |
                NoSuchAlgorithmException | InvalidKeyException | InvalidKeySpecException |
InvalidAlgorithmParameterException e) {
            e.printStackTrace();
        }
        return false;
    }

    private static void initDecodeCipher(int mode) throws KeyStoreException,
NoSuchAlgorithmException, UnrecoverableKeyException, InvalidKeyException {
        PrivateKey key = (PrivateKey) sKeyStore.getKey(KEY_ALIAS, null);
        sCipher.init(mode, key);
    }

    private static void initEncodeCipher(int mode) throws KeyStoreException,
InvalidKeySpecException, NoSuchAlgorithmException, InvalidKeyException,
InvalidAlgorithmParameterException {
        PublicKey key = sKeyStore.getCertificate(KEY_ALIAS).getPublicKey();

        // workaround for using public key
        // from
https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.
html
        PublicKey unrestricted = KeyFactory.getInstance(key.getAlgorithm()).generatePublic(new
X509EncodedKeySpec(key.getEncoded()));
        // from https://code.google.com/p/android/issues/detail?id=197719
        OAEPParameterSpec spec = new OAEPParameterSpec("SHA-256", "MGF1",
MGF1ParameterSpec.SHA1, PSource.PSpecified.DEFAULT);
```

```
        sCipher.init(mode, unrestricted, spec);
    }

    public static void deleteInvalidKey() {
        if (getKeyStore()) {
            try {
                sKeyStore.deleteEntry(KEY_ALIAS);
            } catch (KeyStoreException e) {
                e.printStackTrace();
            }
        }
    }

    @Nullable
    public static FingerprintManagerCompat.CryptoObject getCryptoObject() {
        if (prepare() && initCipher(Cipher.DECRYPT_MODE)) {
            return new FingerprintManagerCompat.CryptoObject(sCipher);
        }
        return null;
    }


}
package com.auth.fingerprintauth;

import android.app.Application;
import android.content.Context;

public class FingerprintAuthApp extends Application {

    private static Context context;

    public void onCreate() {
        super.onCreate();
        FingerprintAuthApp.context = getApplicationContext();
    }

    public static Context getAppContext() {
        return FingerprintAuthApp.context;
    }

}


package com.auth.fingerprintauth;
import android.content.Context;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v4.app.DialogFragment;
```

```
import android.support.v4.hardware.fingerprint.FingerprintManagerCompat;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

public class FingerprintAuthenticationDialogFragment extends DialogFragment
      implements FingerprintUiHelper.Callback {

   private  FingerprintManagerCompat.CryptoObject mCryptoObject;
   private FingerprintUiHelper mFingerprintUiHelper;
   private MainActivity mActivity;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setRetainInstance(true);
      setStyle(DialogFragment.STYLE_NORMAL, android.R.style.Theme_Material_Light_Dialog);
   }

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      getDialog().setTitle(getString(R.string.sign_in));
      View v = inflater.inflate(R.layout.fingerprint_dialog_container, container, false);
      Button mCancelButton = v.findViewById(R.id.cancel_button);
      mCancelButton.setOnClickListener(view -> dismiss());

      mFingerprintUiHelper = new FingerprintUiHelper(
            FingerprintManagerCompat.from(getActivity()),
            PreferenceManager.getDefaultSharedPreferences(getActivity()),
            v.findViewById(R.id.fingerprint_description),
            v.findViewById(R.id.fingerprint_icon),
            v.findViewById(R.id.fingerprint_status), this);
      mCancelButton.setText(R.string.cancel);
      if (!mFingerprintUiHelper.isFingerprintAuthAvailable() && getActivity() != null) {
         mActivity.isFingerprintAuthDisable();
      }
      return v;
   }

   @Override
   public void onResume() {
      super.onResume();
      mFingerprintUiHelper.startListening(mCryptoObject);
   }
```

```
    @Override
    public void onPause() {
        super.onPause();
        mFingerprintUiHelper.stopListening();
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mActivity = (MainActivity) getActivity();
    }

    public void setCryptoObject( FingerprintManagerCompat.CryptoObject cryptoObject) {
        mCryptoObject = cryptoObject;
    }

    @Override
    public void onAuthenticated(FingerprintManagerCompat.AuthenticationResult result) {
        if (getActivity() != null) {
            mActivity.onAuthenticated(result);
        }
        dismiss();
    }

    @Override
    public void onNeedDeletePass() {
        if (getContext() != null) {
            Toast.makeText(getContext(), getString(R.string.fingerprint_exceeded_trying_count),
Toast.LENGTH_SHORT).show();
        }
        if (getActivity() != null) {
            mActivity.onNeedDeletePass();
        }
        dismiss();
    }

    @Override
    public void onError() {

    }

}

package com.auth.fingerprintauth;

import android.content.SharedPreferences;
import android.support.v4.hardware.fingerprint.FingerprintManagerCompat;
import android.support.v4.os.CancellationSignal;
```

```
import android.widget.ImageView;
import android.widget.TextView;

public class FingerprintUiHelper extends FingerprintManagerCompat.AuthenticationCallback {

    private static final String TRYING_COUNT = "trying_count";
    private static final long ERROR_TIMEOUT_MILLIS = 1600;
    private static final long SUCCESS_DELAY_MILLIS = 1300;

    private final FingerprintManagerCompat mFingerprintManager;
    private final SharedPreferences mSharedPreferences;
    private final TextView mDescriptionTextView;
    private final ImageView mIcon;
    private final TextView mErrorTextView;
    private final Callback mCallback;
    private CancellationSignal mCancellationSignal;

    private boolean mSelfCancelled;

    FingerprintUiHelper(FingerprintManagerCompat fingerprintManager, SharedPreferences
sharedPreferences,
                        TextView descriptionTextView, ImageView icon, TextView errorTextView,
Callback callback) {
        mFingerprintManager = fingerprintManager;
        mSharedPreferences = sharedPreferences;
        mDescriptionTextView = descriptionTextView;
        mIcon = icon;
        mErrorTextView = errorTextView;
        mCallback = callback;
    }

    public boolean isFingerprintAuthAvailable() {
        return mFingerprintManager.isHardwareDetected()
                && mFingerprintManager.hasEnrolledFingerprints();
    }

    public void startListening(FingerprintManagerCompat.CryptoObject cryptoObject) {
        if (!isFingerprintAuthAvailable()) {
            return;
        }
        mCancellationSignal = new CancellationSignal();
        mSelfCancelled = false;

        mFingerprintManager
                .authenticate(cryptoObject, 0, mCancellationSignal, this, null);
        mIcon.setImageResource(R.drawable.ic_fp_40px);
        setDescription();
    }
```

```
private void setDescription() {
    int tryingCount = mSharedPreferences.getInt(TRYING_COUNT, 5);
    if (tryingCount < 5) {
        mDescriptionTextView.setText(mDescriptionTextView.getContext()
                .getString(R.string.fingerprint_description_trying_count, tryingCount));
        if (tryingCount < 1) {
            mSharedPreferences.edit().putInt(TRYING_COUNT, 5).apply();
            mCallback.onNeedDeletePass();
        }
    }
}

public void stopListening() {
    if (mCancellationSignal != null) {
        mSelfCancelled = true;
        mCancellationSignal.cancel();
        mCancellationSignal = null;
    }
}

@Override
public void onAuthenticationError(int errMsgId, CharSequence errString) {
    if (!mSelfCancelled) {
        showError(errString);
        mIcon.postDelayed(mCallback::onError, ERROR_TIMEOUT_MILLIS);
    }
}

@Override
public void onAuthenticationHelp(int helpMsgId, CharSequence helpString) {
    showError(helpString);
}

@Override
public void onAuthenticationFailed() {
    showError(mIcon.getResources().getString(
            R.string.fingerprint_not_recognized));

    int tryingCount = mSharedPreferences.getInt(TRYING_COUNT, 5);
    if (tryingCount <= 1) {
        mSharedPreferences.edit().putInt(TRYING_COUNT, 5).apply();
        mIcon.postDelayed(mCallback::onNeedDeletePass, ERROR_TIMEOUT_MILLIS);
    }
    mSharedPreferences.edit().putInt(TRYING_COUNT, tryingCount - 1).apply();
    setDescription();
}
```

```
    @Override
    public void onAuthenticationSucceeded(final
FingerprintManagerCompat.AuthenticationResult result) {
        mErrorTextView.removeCallbacks(mResetErrorTextRunnable);
        mIcon.setImageResource(R.drawable.ic_fingerprint_success);
        mErrorTextView.setTextColor(
                mErrorTextView.getResources().getColor(R.color.success_color, null));
        mErrorTextView.setText(
                mErrorTextView.getResources().getString(R.string.fingerprint_success));

        mSharedPreferences.edit().putInt(TRYING_COUNT, 5).apply();
        mIcon.postDelayed(() -> mCallback.onAuthenticated(result), SUCCESS_DELAY_MILLIS);
    }

    private void showError(CharSequence error) {
        mIcon.setImageResource(R.drawable.ic_fingerprint_error);
        mErrorTextView.setText(error);
        mErrorTextView.setTextColor(
                mErrorTextView.getResources().getColor(R.color.warning_color, null));
        mErrorTextView.removeCallbacks(mResetErrorTextRunnable);
        mErrorTextView.postDelayed(mResetErrorTextRunnable, ERROR_TIMEOUT_MILLIS);
    }

    private Runnable mResetErrorTextRunnable = new Runnable() {
        @Override
        public void run() {
            mErrorTextView.setTextColor(
                    mErrorTextView.getResources().getColor(R.color.hint_color, null));
            mErrorTextView.setText(
                    mErrorTextView.getResources().getString(R.string.fingerprint_hint));
            mIcon.setImageResource(R.drawable.ic_fp_40px);
        }
    };

    public interface Callback {

        void onAuthenticated(FingerprintManagerCompat.AuthenticationResult result);

        void onNeedDeletePass();

        void onError();
    }
}

package com.auth.fingerprintauth;
import android.annotation.TargetApi;
import android.app.KeyguardManager;
import android.content.Context;
```

```
import android.os.Build;
import android.support.annotation.NonNull;
import android.support.v4.hardware.fingerprint.FingerprintManagerCompat;

public final class FingerprintUtils {
   private FingerprintUtils() {
   }
   public enum mSensorState {
      NOT_SUPPORTED,
      NOT_BLOCKED,
      NO_FINGERPRINTS,
      READY
   }
   public static boolean checkFingerprintCompatibility(@NonNull Context context) {
      return FingerprintManagerCompat.from(context).isHardwareDetected();
   }

   @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
   public static mSensorState checkSensorState(@NonNull Context context) {
      if (checkFingerprintCompatibility(context)) {

         KeyguardManager keyguardManager = (KeyguardManager)
context.getSystemService(Context.KEYGUARD_SERVICE);
         if (!keyguardManager.isKeyguardSecure()) {
            return mSensorState.NOT_BLOCKED;
         }

         if (!FingerprintManagerCompat.from(context).hasEnrolledFingerprints()) {
            return mSensorState.NO_FINGERPRINTS;
         }
         return mSensorState.READY;
      } else {
         return mSensorState.NOT_SUPPORTED;
      }
   }

   @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
   public static boolean isSensorStateAt(@NonNull mSensorState state, @NonNull Context
context) {
      return checkSensorState(context) == state;
   }
}

package com.auth.fingerprintauth;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
```

```
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.widget.EditText;
import android.widget.Toast;

public class LoginActivity extends AppCompatActivity {


   public static final String IP = "ip";
   public static final String LOGIN = "login";
   public static final String ENCODED_PASS = "pass";

   private EditText mEditTextIp;
   private EditText mEditTextLogin;
   private EditText mEditTextPass;
   private SharedPreferences mPreferences;


   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_login);
      mPreferences = PreferenceManager.getDefaultSharedPreferences(this);

      mEditTextIp = findViewById(R.id.ip_edt);
      mEditTextLogin = findViewById(R.id.login_edt);
      mEditTextPass = findViewById(R.id.pass_edt);

      findViewById(R.id.login_btn).setOnClickListener(view -> prepareLogin());
   }


   private void prepareLogin() {
      final String ip = mEditTextIp.getText().toString();
      final String login = mEditTextLogin.getText().toString();
      final String pass = mEditTextPass.getText().toString();
      if (isValidData(ip, login, pass)) {
         saveAuthData(ip, login, pass);
         finish();
         startActivity(new Intent(this, MainActivity.class));
      }
   }

   private boolean isValidData(String ip, String login, String pass) {
      if (ip.isEmpty()){
         Toast.makeText(this, getString(R.string.ip_empty_error),
Toast.LENGTH_SHORT).show();
         return false;
```

```
        }
        if (login.isEmpty()){
            Toast.makeText(this, getString(R.string.login_empty_error),
Toast.LENGTH_SHORT).show();
            return false;
        }
        if (pass.isEmpty()){
            Toast.makeText(this, getString(R.string.pass_empty_error),
Toast.LENGTH_SHORT).show();
            return false;
        }

        return true;
    }

    private void saveAuthData(String ip, String login, String pass) {
        if (FingerprintUtils.isSensorStateAt(FingerprintUtils.mSensorState.READY, this)) {
            String encodedPass = CryptoUtils.encode(pass);
            mPreferences.edit()
                    .putString(IP, ip)
                    .putString(LOGIN, login)
                    .putString(ENCODED_PASS, encodedPass)
                    .apply();
        }
    }
}

package com.auth.fingerprintauth;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.preference.PreferenceManager;
import android.support.v4.hardware.fingerprint.FingerprintManagerCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;
import java.io.*;
import java.lang.Object.*;
import android.app.*;
import android.content.*;
import android.net.*;
import android.os.*;
import android.view.*;
```

```
import android.graphics.*;
import android.widget.*;
import android.provider.*;

import com.auth.fingerprintauth.model.ApiFactory;

import javax.crypto.Cipher;

import okhttp3.ResponseBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

import static com.auth.fingerprintauth.LoginActivity.ENCODED_PASS;
import static com.auth.fingerprintauth.LoginActivity.IP;
import static com.auth.fingerprintauth.LoginActivity.LOGIN;

public class MainActivity extends AppCompatActivity {

   private FingerprintAuthenticationDialogFragment fingerprintDialogFragment;
   private SharedPreferences mPreferences;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      mPreferences = PreferenceManager.getDefaultSharedPreferences(this);
      if (!mPreferences.contains(ENCODED_PASS)) {
         onNeedDeletePass();
         return;
      }

      setContentView(R.layout.activity_main);

      fingerprintDialogFragment = new FingerprintAuthenticationDialogFragment();

      ((TextView) findViewById(R.id.login_txt))
            .setText(getString(R.string.login_temp, mPreferences.getString(LOGIN, "Bac")));
      findViewById(R.id.fingerprint_btn).setOnClickListener(view -> showFingerprintDialog());
      new Handler().postDelayed(this::showFingerprintDialog, 300);

   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.menu_main, menu);
      return true;
   }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()){
            case R.id.action_logout:
                onNeedDeletePass();
                break;
        }
        return super.onOptionsItemSelected(item);
    }


    private void showFingerprintDialog() {
        if (FingerprintUtils.isSensorStateAt(FingerprintUtils.mSensorState.READY, this)) {
            FingerprintManagerCompat.CryptoObject cryptoObject =
CryptoUtils.getCryptoObject();
            if (cryptoObject != null) {
                if
(getFragmentManager().findFragmentByTag("FingerprintAuthenticationDialogFragmentTag")
== null) {
                    fingerprintDialogFragment.setCryptoObject(cryptoObject);
                    fingerprintDialogFragment.show(getSupportFragmentManager(),
"FingerprintAuthenticationDialogFragmentTag");
                }

            } else {
                mPreferences.edit().remove(LOGIN).apply();
                mPreferences.edit().remove(ENCODED_PASS).apply();
                Toast.makeText(this, getString(R.string.new_fingerprint_enrolled_description),
Toast.LENGTH_SHORT).show();
            }

        }
    }

    public void isFingerprintAuthDisable() {

    }


    public void onAuthenticated(FingerprintManagerCompat.AuthenticationResult result) {
        Cipher cipher = result.getCryptoObject().getCipher();
        String encodedFingerprint = mPreferences.getString(ENCODED_PASS, null);
//        String decodedFingerprint = CryptoUtils.decode(encodedPass, cipher);
        new Handler().postDelayed(() -> startAuthRequest(encodedFingerprint), 500);
    }

    private void startAuthRequest(String encodedFingerprint) {
        ApiFactory.getApiRequestService()
```

```
            .signUp(mPreferences.getString(LOGIN, null), encodedFingerprint)
            .enqueue(new Callback<ResponseBody>() {
               @Override
               public void onResponse(Call<ResponseBody> call, Response<ResponseBody>
response) {
                  if (response.isSuccessful()) {
                     Toast.makeText(MainActivity.this, getString(R.string.sign_in_successful),
Toast.LENGTH_LONG).show();
                  } else {
                     Toast.makeText(MainActivity.this, getString(R.string.request_error),
Toast.LENGTH_SHORT).show();
                     onNeedDeletePass();
                  }
               }

               @Override
               public void onFailure(Call<ResponseBody> call, Throwable t) {
                  Log.e("TAG", call.request().toString());
                  Toast.makeText(MainActivity.this, "Помилка: " + t.getMessage(),
Toast.LENGTH_SHORT).show();

               }
            });

   }


   public void onNeedDeletePass() {
      mPreferences.edit().remove(IP).apply();
      mPreferences.edit().remove(LOGIN).apply();
      mPreferences.edit().remove(ENCODED_PASS).apply();

      finish();
      startActivity(new Intent(this, LoginActivity.class));
   }
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Button button1 = findViewById(R.id.button1);
   button1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
         intent.setType("image/*");
         startActivityForResult(Intent.createChooser(
               intent, "Select Picture"), PICK_IMAGE);
      }
```

```
    });

    detectionProgressDialog = new ProgressDialog(this);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PICK_IMAGE && resultCode == RESULT_OK &&
            data != null && data.getData() != null) {
        Uri uri = data.getData();
        try {
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(
                    getContentResolver(), uri);
            ImageView imageView = findViewById(R.id.imageView1);
            imageView.setImageBitmap(bitmap);

            // Comment out for tutorial
            detectAndFrame(bitmap);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
import com.microsoft.projectoxford.face.*;
import com.microsoft.projectoxford.face.contract.*;
// Add your Face endpoint to your environment variables.
private final String apiEndpoint = System.getenv("FACE_ENDPOINT");
// Add your Face subscription key to your environment variables.
private final String subscriptionKey = System.getenv("FACE_SUBSCRIPTION_KEY");

private final FaceServiceClient faceServiceClient =
        new FaceServiceRestClient(apiEndpoint, subscriptionKey);

private final int PICK_IMAGE = 1;
private ProgressDialog detectionProgressDialog;
private void detectAndFrame(final Bitmap imageBitmap) {
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    imageBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
    ByteArrayInputStream inputStream =
            new ByteArrayInputStream(outputStream.toByteArray());

    AsyncTask<InputStream, String, Face[]> detectTask =
            new AsyncTask<InputStream, String, Face[]>() {
                String exceptionMessage = "";

                @Override
                protected Face[] doInBackground(InputStream... params) {
                    try {
```

```
        publishProgress("Detecting...");
        Face[] result = faceServiceClient.detect(
                params[0],
                true,         // returnFaceId
                false,        // returnFaceLandmarks
                null          // returnFaceAttributes:
                /* new FaceServiceClient.FaceAttributeType[] {
                   FaceServiceClient.FaceAttributeType.Age,
                   FaceServiceClient.FaceAttributeType.Gender }
                */
        );
        if (result == null){
           publishProgress(
                   "Detection Finished. Nothing detected");
           return null;
        }
        publishProgress(String.format(
                "Detection Finished. %d face(s) detected",
                result.length));
        return result;
    } catch (Exception e) {
        exceptionMessage = String.format(
                "Detection failed: %s", e.getMessage());
        return null;
    }
}

@Override
protected void onPreExecute() {
    //TODO: show progress dialog
    detectionProgressDialog.show();
}
@Override
protected void onProgressUpdate(String... progress) {
    //TODO: update progress
    detectionProgressDialog.setMessage(progress[0]);
}
@Override
protected void onPostExecute(Face[] result) {
    //TODO: update face frames
    detectionProgressDialog.dismiss();

    if(!exceptionMessage.equals("")){
        showError(exceptionMessage);
    }
    if (result == null) return;

    ImageView imageView = findViewById(R.id.imageView1);
```

```
                imageView.setImageBitmap(
                        drawFaceRectanglesOnBitmap(imageBitmap, result));
                imageBitmap.recycle();
            }
        };

    detectTask.execute(inputStream);
}

private void showError(String message) {
    new AlertDialog.Builder(this)
        .setTitle("Error")
        .setMessage(message)
        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
            }})
        .create().show();
}
private static Bitmap drawFaceRectanglesOnBitmap(
      Bitmap originalBitmap, Face[] faces) {
    Bitmap bitmap = originalBitmap.copy(Bitmap.Config.ARGB_8888, true);
    Canvas canvas = new Canvas(bitmap);
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setStyle(Paint.Style.STROKE);
    paint.setColor(Color.RED);
    paint.setStrokeWidth(10);
    if (faces != null) {
        for (Face face : faces) {
            FaceRectangle faceRectangle = face.faceRectangle;
            canvas.drawRect(
                    faceRectangle.left,
                    faceRectangle.top,
                    faceRectangle.left + faceRectangle.width,
                    faceRectangle.top + faceRectangle.height,
                    paint);
        }
    }
    return bitmap;
}

}
import java.io.*;
import java.lang.Object.*;
import android.app.*;
import android.content.*;
import android.net.*;
import android.os.*;
```

```
import android.view.*;
import android.graphics.*;
import android.widget.*;
import android.provider.*;
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Button button1 = findViewById(R.id.button1);
   button1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
         intent.setType("image/*");
         startActivityForResult(Intent.createChooser(
               intent, "Select Picture"), PICK_IMAGE);
      }
   });

   detectionProgressDialog = new ProgressDialog(this);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   super.onActivityResult(requestCode, resultCode, data);
   if (requestCode == PICK_IMAGE && resultCode == RESULT_OK &&
         data != null && data.getData() != null) {
      Uri uri = data.getData();
      try {
         Bitmap bitmap = MediaStore.Images.Media.getBitmap(
               getContentResolver(), uri);
         ImageView imageView = findViewById(R.id.imageView1);
         imageView.setImageBitmap(bitmap);

         // Comment out for tutorial
         detectAndFrame(bitmap);
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
import com.microsoft.projectoxford.face.*;
import com.microsoft.projectoxford.face.contract.*;
private final String apiEndpoint = System.getenv("FACE_ENDPOINT");
private final String subscriptionKey = System.getenv("FACE_SUBSCRIPTION_KEY");
private final FaceServiceClient faceServiceClient =
      new FaceServiceRestClient(apiEndpoint, subscriptionKey);
private final int PICK_IMAGE = 1;
private ProgressDialog detectionProgressDialog;
```

```
private void detectAndFrame(final Bitmap imageBitmap) {
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    imageBitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
    ByteArrayInputStream inputStream =
            new ByteArrayInputStream(outputStream.toByteArray());

    AsyncTask<InputStream, String, Face[]> detectTask =
            new AsyncTask<InputStream, String, Face[]>() {
                String exceptionMessage = "";

                @Override
                protected Face[] doInBackground(InputStream... params) {
                    try {
                        publishProgress("Detecting...");
                        Face[] result = faceServiceClient.detect(
                                params[0],
                                true,         // returnFaceId
                                false,        // returnFaceLandmarks
                                null          // returnFaceAttributes:
                                /* new FaceServiceClient.FaceAttributeType[] {
                                    FaceServiceClient.FaceAttributeType.Age,
                                    FaceServiceClient.FaceAttributeType.Gender }
                                */
                        );
                        if (result == null){
                            publishProgress(
                                    "Detection Finished. Nothing detected");
                            return null;
```

```
        }
        publishProgress(String.format(
                "Detection Finished. %d face(s) detected",
                result.length));
        return result;
    } catch (Exception e) {
        exceptionMessage = String.format(
                "Detection failed: %s", e.getMessage());
        return null;
    }
}

@Override
protected void onPreExecute() {
    //TODO: show progress dialog
    detectionProgressDialog.show();
}
@Override
protected void onProgressUpdate(String... progress) {
    //TODO: update progress
    detectionProgressDialog.setMessage(progress[0]);
}
@Override
protected void onPostExecute(Face[] result) {
    //TODO: update face frames
    detectionProgressDialog.dismiss();

    if(!exceptionMessage.equals("")){
```

```
                showError(exceptionMessage);
            }
            if (result == null) return;

            ImageView imageView = findViewById(R.id.imageView1);
            imageView.setImageBitmap(
                    drawFaceRectanglesOnBitmap(imageBitmap, result));
            imageBitmap.recycle();
        }
    };
    detectTask.execute(inputStream);
}
private void showError(String message) {
    new AlertDialog.Builder(this)
        .setTitle("Error")
        .setMessage(message)
        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
            }})
        .create().show();
}
private static Bitmap drawFaceRectanglesOnBitmap(
        Bitmap originalBitmap, Face[] faces) {
    Bitmap bitmap = originalBitmap.copy(Bitmap.Config.ARGB_8888, true);
    Canvas canvas = new Canvas(bitmap);
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setStyle(Paint.Style.STROKE);
```

```
    paint.setColor(Color.RED);
    paint.setStrokeWidth(10);
    if (faces != null) {
        for (Face face : faces) {
            FaceRectangle faceRectangle = face.faceRectangle;
            canvas.drawRect(
                    faceRectangle.left,
                    faceRectangle.top,
                    faceRectangle.left + faceRectangle.width,
                    faceRectangle.top + faceRectangle.height,
                    paint);
        }
    }
    return bitmap;
}
```

Додаток Г. Критерії оцінювання комерційного потенціалу

Таблиця Г.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатністьпродукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатацій-ні витрати на рівні експлуатацій-них витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна Конкурен-ція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

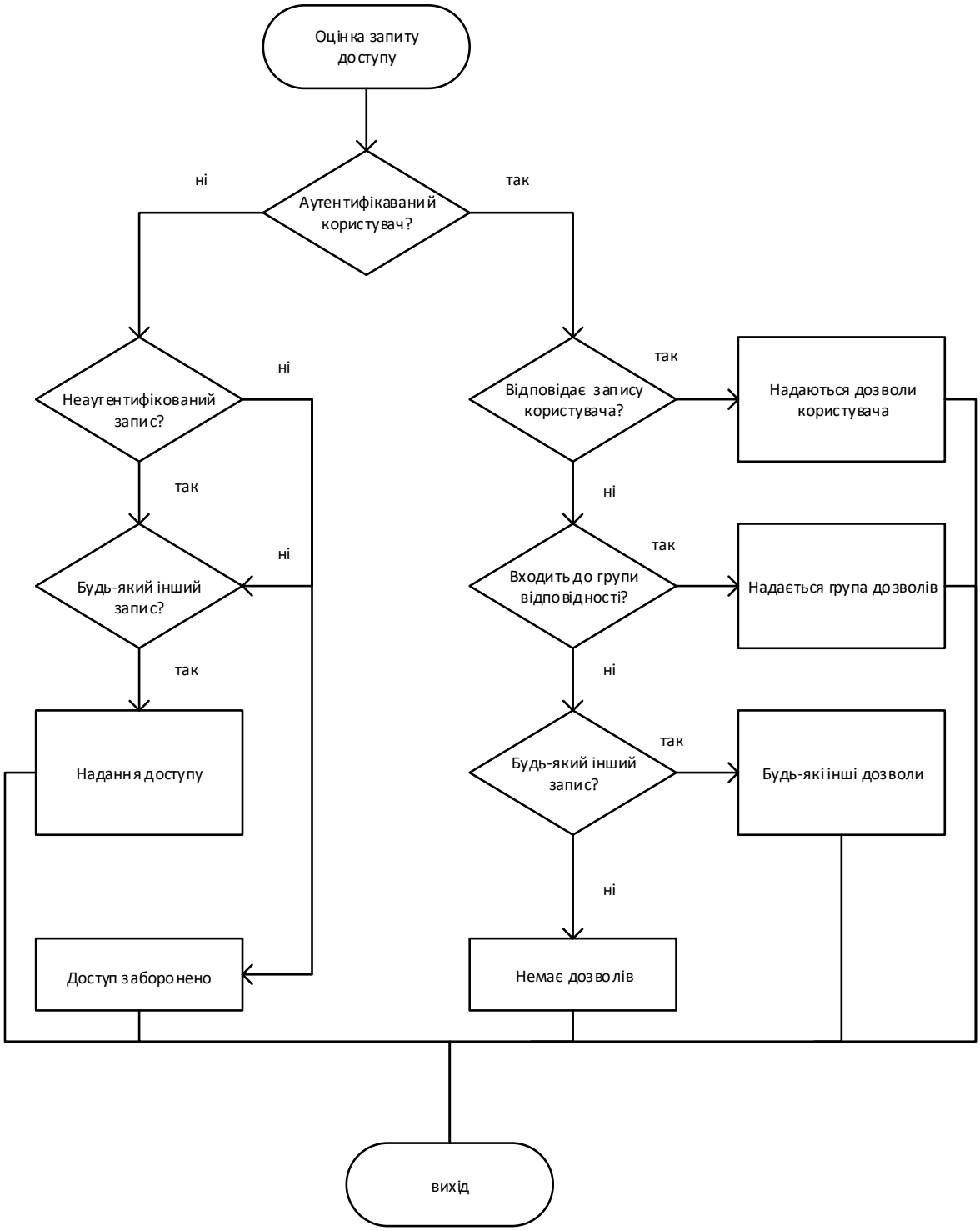
Продовження таблиці Г.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх Штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років.Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

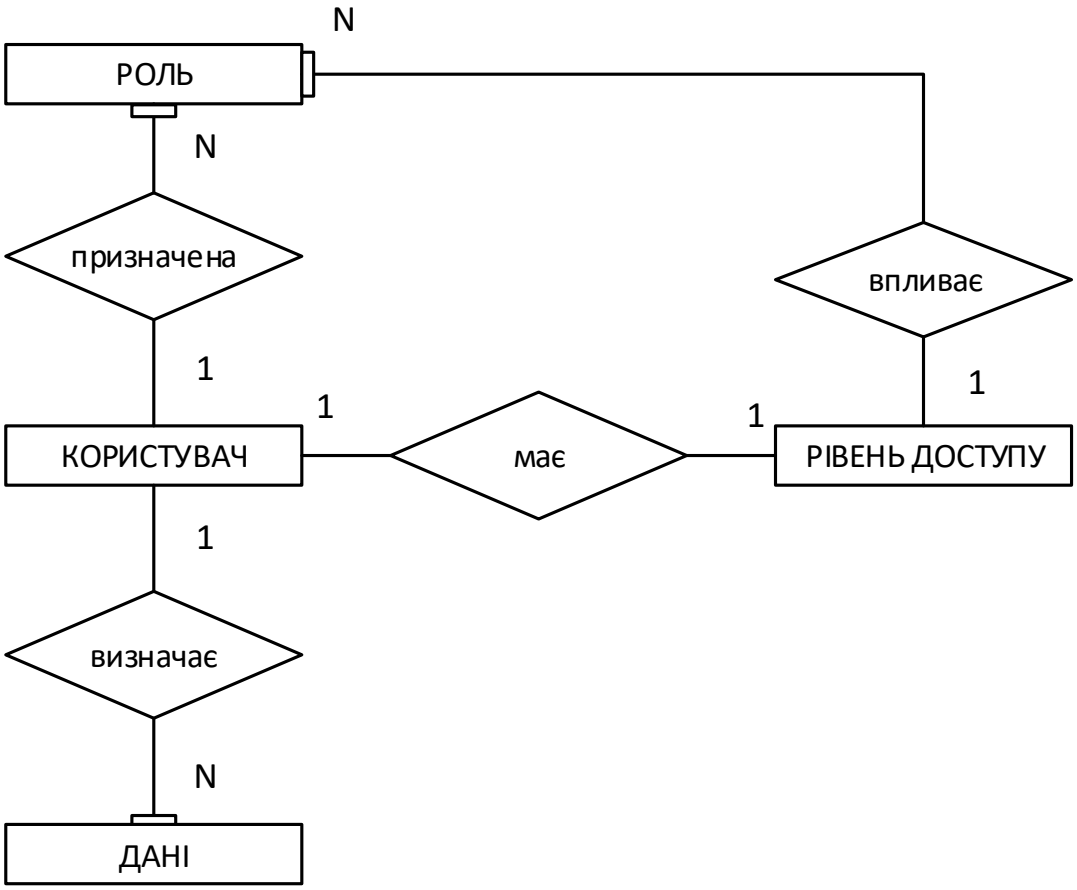
Додаток Д. Графічна частина

Технології розмежування доступу в комп'ютерних системах

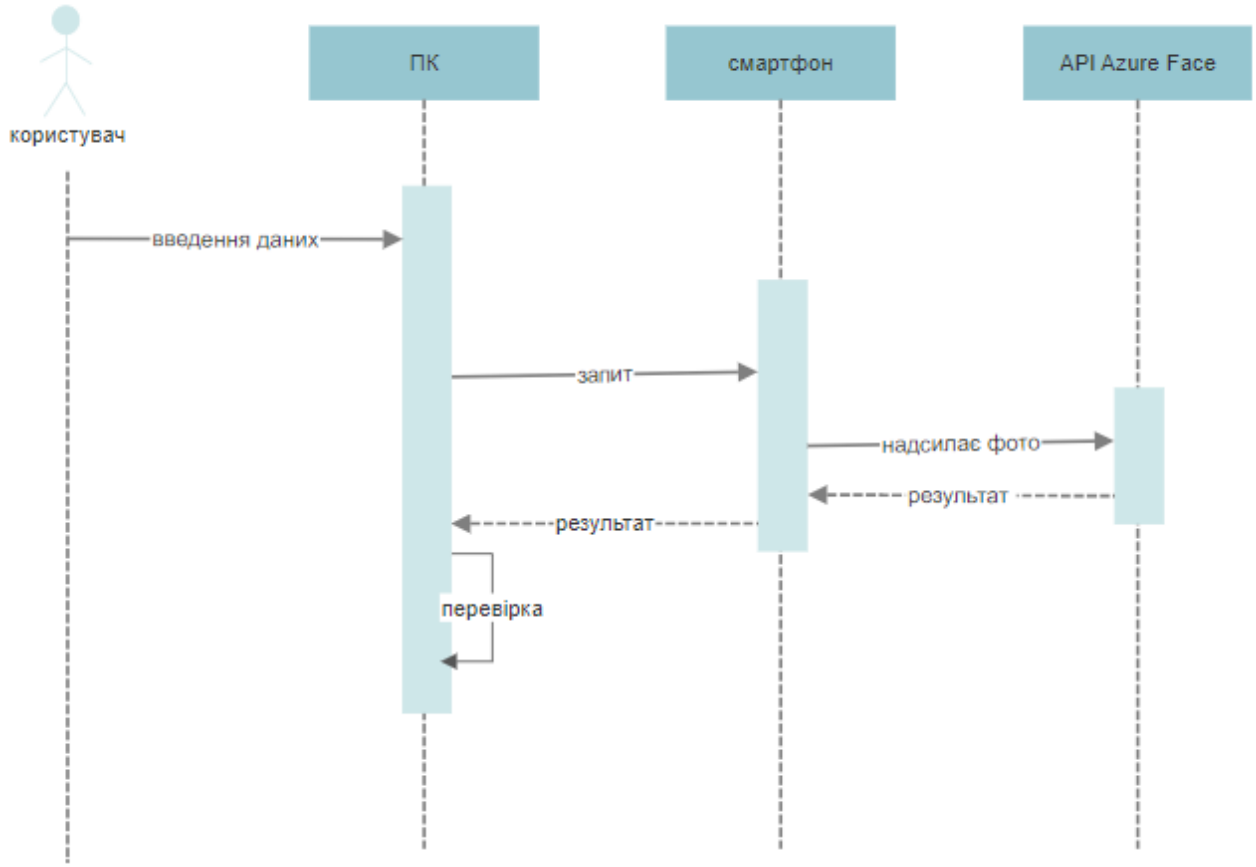
Оцінка контролю доступу

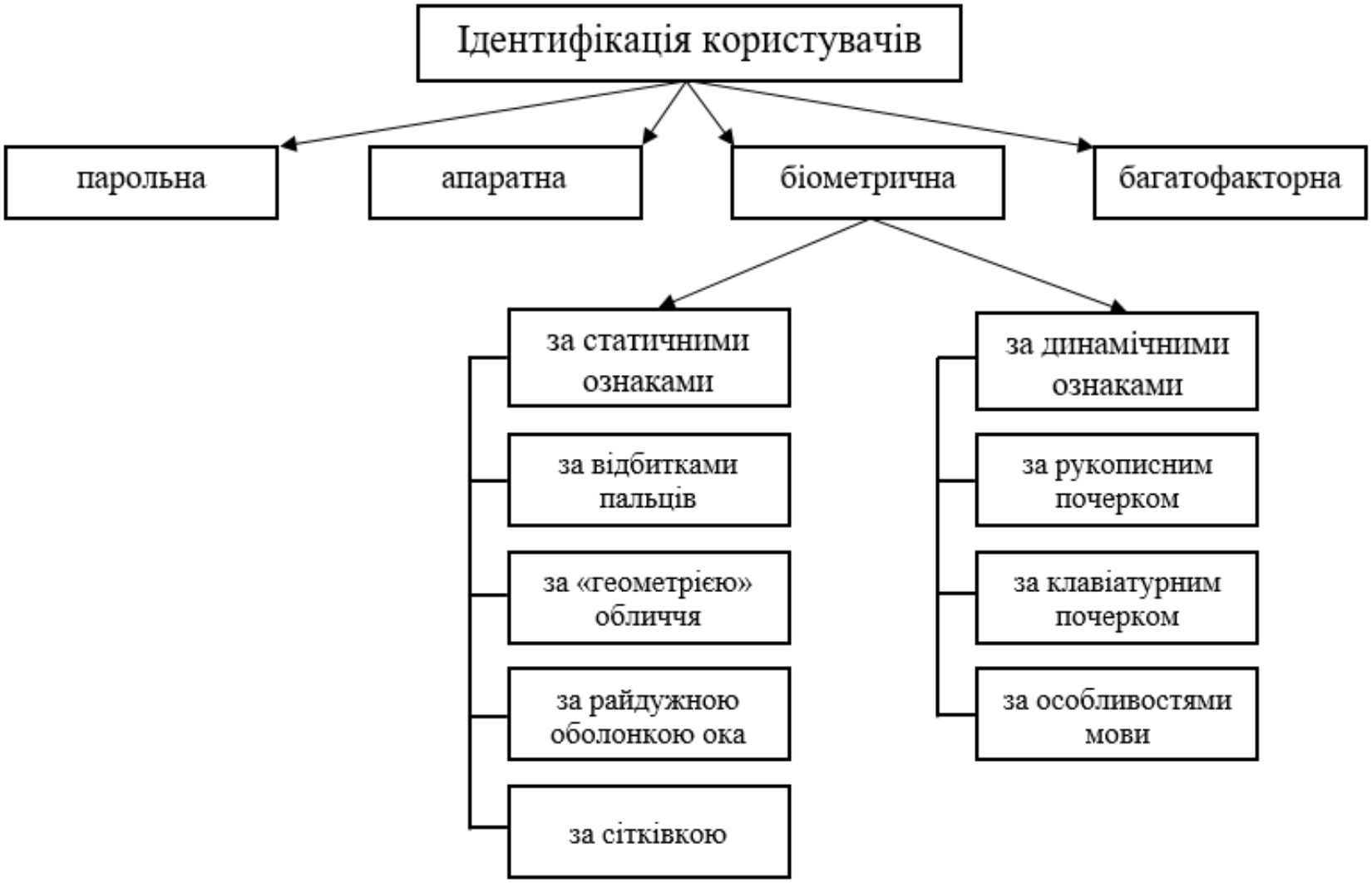


ER-модель


N
РОЛЬ
N
призначена
1
КОРИСТУВАЧ
1
має
1
РІВЕНЬ ДОСТУПУ
впливає
1
1
визначає
N
ДАНІ

Діаграма послідовностей

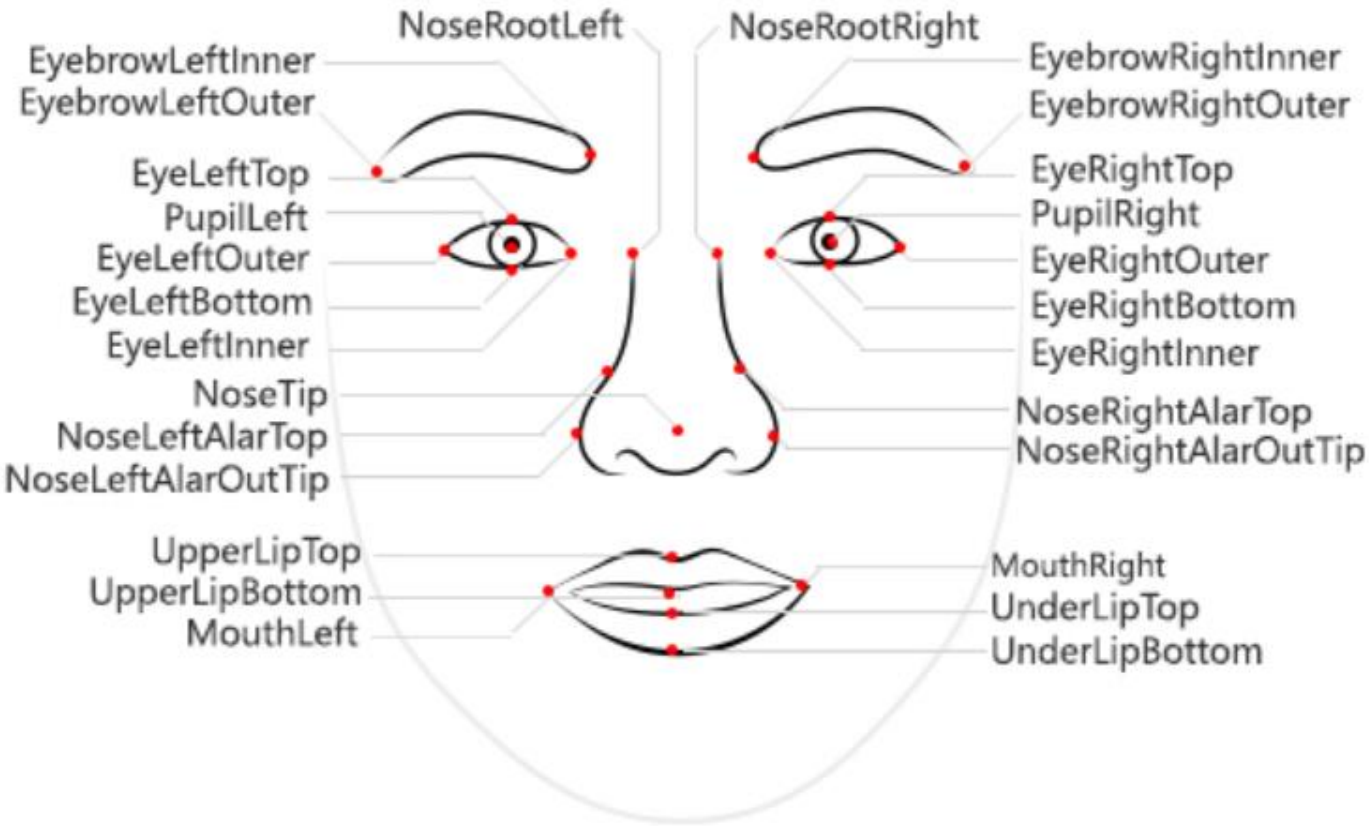




Способи ідентифікації користувачів

Основні характеристики методів біометричної ідентифікації

Метод отримання біометричних параметрів	Ймовірність відмови у доступі, %	Ймовірність помилкової ідентифікації «чужого» (без використання муляжу), %	Ймовірність помилкової ідентифікації «чужого» (з використанням муляжу), %	Збереження таємниці образу у процесі ідентифікації абонента	Вартість технічної реалізації в грошовому еквіваленті, у.о.
Геометрична будова руки	0,2…4	0,2….1	10…75	неможливо приховати	Від 600 до 3000
Відбитки пальців	2...6	0,0001	10...70	неможливо приховати	Від 60 до 600
Особливості малюнка сітківки ока	0,4	6...10	_____	неможливо приховати	приблизно 4000
Райдужна оболонка ока	0,2…2	0,0001	_____	неможливо приховати	Від 500 до 6000
Портрет обличчя	0,8…1	_____	_____	неможливо приховати	55000
Рукописний почерк	0,5…5	0,5…5	0,5...5	8-10…10-40	_____
Клавіатурний та комп'ютерний почерк	3…9	3...9	_____	6-10…10-12	_____
Характеристики і особливості мови	0,5...5	0,5...5	25...90 (запис)	10-16…10-30	1...60



Маркери визначення особливостей обличчя

Вигляд вікна програми мобільного модулю