ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики

«До захисту допущено»
Завідувачка кафедри ПМІ

_____ Ольга ДМИТРІЄВА
(підпис) (ім'я та прізвище)

«18» червня 2022 р.

Випускна кваліфікаційна робота
бакалавра
(освітньо-кваліфікаційний рівень)

на тему: «Забезпечення інформаційної безпеки банківських додатків» зі спецчастиною «Розробка модулю автентифікації й захищеного входу в систему засобами Java»

Виконав: студент 4 курсу, групи КІБ-18
(шифр групи)
Спеціальності (напряму підготовки) 125 «Кібербезпека»
(шифр і спеціальності)
Дудкін А.О.
(прізвище та ініціали) (підпис)

Керівник доц. кафедри ПМІ, к.т.н., доц. Наталія МАСЛОВА _____
(посада, науковий ступінь, вчене звання, ім'я та прізвище) (підпис)

Рецензент доц. кафедри КІ, к.т.н., доц. Віталій ШАМАЄВ _____
(посада, науковий ступінь, вчене звання, ім'я та прізвище) (підпис)

Нормоконтроль:
_____ к.т.н., доц. Ірина НАЗАРОВА
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Луцьк – 2022 р.

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітньо-кваліфікаційний рівень (освітній ступінь) _бакалавр
Напрям підготовки (спеціальність) 125 Кібербезпека
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувачка кафедри ПМІ

_____/Ольга ДМИТРІЄВА
"6".05.2022 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____Дудкіну Андрію Олексійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи: Забезпечення інформаційної безпеки банківських додатків_
Спецчастина: Розробка модулю автентифікації й захищеного входу в систему засобами Java_____

Керівник роботи Маслова Н.О., к.т.н., доц..
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом від "06" травня 2022 року №180
2. Строк подання студентом роботи_____17 червня 2022р._____
3. Вихідні дані до роботи_результати переддипломної практики; система захисту інформації в мобільних додатках; _язик програмування Java_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно_ розробити):
1) Аналіз предметної області та постановка задачі
2) Методологія побудови двофакторної автентифікації, як важливого фактору інформаційної безпеки банківських додатків;
3) Розробка удосконаленого методу захищеності банківських додатків;
4) Результати розробки програмного забезпечення;

5. Перелік графічного матеріалу
робота містить 15 рисунків, 15 слайдів презентації та інший графічний матеріал, у кількості, достатній для демонстрації сутності роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 6 травня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз літератури з предметної області	06.05.2022 – 10.05.2022	
2	Збір інформації і методичних відомостей щодо Захисту додатків	11.05.2022 – 15.05.2022	
3	Розробка структури програмного забезпечення, вибір даних для обробки та експериментів.	16.05.2022 – 22.05.2022_	
4	Тестування системи (модуля)	23.05.2022 – 27. 05.2022	
5	Оформлення пояснювальної записки	28.05.2022 – 08. 06.2022_	
6	Нормоконтроль пояснювальної записки, її перевірка антиплагіатною системою	09.06.2022- 10.06.2022_	
7	Оформлення супутніх матеріалів (розробка графічного матеріалу, написання доповіді, тощо) та рецензування роботи	11.06.2022- 23.06.2022	
8	Захист роботи	24.06.2022	

Студент _____ Дудкін А.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Маслова Н.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дудкін Андрій. Забезпечення інформаційної безпеки банківських додатків/ Випускна кваліфікаційна робота на здобуття освітньо-кваліфікаційного рівня «бакалавр» за напрямом підготовки 125 «Кібербезпека». – ДВНЗ ДонНТУ, Луцьк, 2022.

Об’єкт дослідження є захист інформації банківських додатків.

Предмет дослідження – методи захисту в мобільних додатках банку.

Метою роботи є аналіз захищеності інформації, розробка удосконаленого методу захищеності додатків

Виходячи з мети, завданням даної дипломної роботи є аналіз сучасних видів захищеності мобільних (застосунків) банків; розробка удосконаленого методу захищеності додатків; програмного модуля даного методу, з використанням мови програмування Java.

Проведені дослідження в даній роботі базуються на сучасних методах аналізу механізмів захисту та аутентифікації.

Практичне значення полягає в авторській розробці програмного забезпечення мобільних додатків банківської системи на програмній мові Java

В результаті роботи розроблено програмний код для захисту мобільних додатків, а також розглянуто загрози та методи захищеності мобільного додатку.

Ключові слова: безпека, аутентифікація, мобільні додатки, Java, програма.

ЗМІСТ

ВСТУП

Мобільні програми банків стали справжнім проривом у банківському обслуговуванні, а серед усіх каналів дистанційного банківського обслуговування є найперспективнішим напрямком розвитку.

З розвитком технологічного прогресу та загального рівня інформатизації виникають також нові загрози. Кіберзлочинність дозволяє зловмисникам вчиняти протиправні та незаконні дії, перебуваючи за тисячі кілометрів від мети їх атаки.

На сьогоднішній день важко уявити роботу підприємств та установ без комп'ютерних мереж та баз даних. Державні, комерційні та особисті секрети довіряються комп'ютерам, у зв'язку з чим виникає потреба у надійних засобах безпеки для захисту інформаційних даних.

Найбільш надійним способом є двофакторна автентифікація (2FA) – автентифікація, у процесі якої використовуються автентифікаційні фактори кількох типів. В цьому випадку зловмисник не зможе отримати доступ до даних, тому що йому доведеться не тільки підглянути пароль, але й пред'явити фізичний пристрій, крадіжка якого, на відміну від крадіжки пароля, завжди швидко виявляється.

Метою роботи є аналіз захищеності інформації, розробка удосконаленого методу захищеності додатків.

З урахування мети роботи потрібно виконати наступні завдання:

1) провести аналіз сучасних систем захисту інформації в банківській сфері;

2) проаналізувати загрози інформаційної безпеки банківських мобільних додатків;

3) розробити програмний модуль двофакторної автентифікації в мобільні застосування банківської системи.

Методи розробки базуються на положеннях теорії захисту інформації, методах та вимогах до створення програмних модулів захисту, сучасних застосуваннях в кібербезпеці та кібераналізі.

Обґрунтованість розробленого модуля повинно забезпечуватися відповідними схемами та програмами, що демонструють його працездатність.

Кваліфікаційна робота складається з 4 розділів, включає 15 рисунків, 2 таблиці, 29 джерел інформації, загалом 70 сторінок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасної системи захисту інформації в банківській сфері

Стратегія інформаційної безпеки банків дуже відрізняється від аналогічних стратегій інших компаній та організацій. Це обумовлено насамперед специфічним характером загроз, а також публічною діяльністю банків, які змушені робити доступ до рахунків досить легким для зручності для клієнтів [1].

Звичайна компанія будує свою інформаційну безпеку, виходячи лише з вузького кола потенційних загроз – головним чином захист інформації від конкурентів. Така інформація цікава лише вузькому колу зацікавлених осіб, і організацій рідко буває ліквідна, тобто перетворюється на грошову форму.

Інформаційна безпека банку має враховувати такі специфічні фактори [2].

Збережена і оброблювана у банківських системах інформація – це реальні гроші. З інформації комп'ютера можуть проводитися виплати, відкриватися кредити, переказуватися значні суми. Цілком зрозуміло, що незаконне маніпулювання з такою інформацією може призвести до серйозних збитків. Ця особливість різко розширює коло злочинців, які робили замах саме на банки (на відміну від, наприклад, промислових компаній, внутрішня інформація яких мало кому цікава).

Інформація у банківських системах зачіпає інтереси великої кількості громадян та організацій – клієнтів банку. Як правило, вона конфіденційна, і банк відповідає за забезпечення необхідного рівня таємності перед своїми клієнтами. Звичайно, клієнти вправі очікувати, що банк повинен дбати про їхні інтереси, інакше він ризикує своєю репутацією з усіма наслідками, що звідси випливають.

Конкурентоспроможність банку залежить від того, наскільки клієнту зручно працювати з банком, а також наскільки широкий спектр послуг,

включаючи послуги, пов'язані з віддаленим доступом. Тому клієнт повинен мати можливість швидко та без додаткових процедур розпоряджатися своїми грошима. Але така легкість доступу до грошей підвищує ймовірність злочинного проникнення у банківські системи.

Інформаційна безпека банку (на відміну від більшості компаній) повинна забезпечувати високу надійність роботи комп'ютерних систем навіть у разі позаштатних ситуацій, оскільки банк відповідає не тільки за власні кошти, а й за гроші клієнтів.

Банк зберігає важливу інформацію про своїх клієнтів, що розширює коло потенційних зловмисників, зацікавлених у крадіжці або псуванні такої інформації.

Найбільш небезпечними для будь-якого банку з економічних наслідків є дії, пов'язані з несанкціонованим доступом до банківських комп'ютерних мереж та систем. Це призвело до необхідності організувати захист банківських структур від спроб протиправного отримання конфіденційної інформації, що накопичується та обробляється у засобах інформатизації банків у вигляді інформаційних ресурсів – окремих фінансових та інших документів та масивів таких документів, що містять відомості, що належать до комерційної чи банківської таємниці. На практиці багато банків продовжували і продовжують здебільшого вдосконалювати методи охорони своїх матеріальних цінностей, а також системи інженерно-технічного та програмного захисту конфіденційної інформації, що становить комерційну таємницю самого банку та його клієнтів – юридичних осіб, або персональні дані його клієнтів – фізичних осіб. При цьому приділяється мало уваги вдосконаленню організаційних заходів щодо захисту інформації, зокрема, вдосконаленню роботи зі створення сукупності організаційно-розпорядчих документів, організаційних методів та заходів, що регламентують як аналітичну роботу щодо виявлення зовнішніх та внутрішніх загроз інформаційній безпеці банку, так і роботу зі створення комплексної системи захисту інформації у банку.

Для захисту комп'ютерної конфіденційної інформації в корпоративній мережі банку основним організаційним заходом є створення адміністративної служби захисту цієї інформації у складі загальної служби безпеки банку та визначені вимоги до адміністратора системи захисту комп'ютерної інформації в корпоративній мережі банку (адміністратору безпеки), а також сфера його відповідальності. Порядок роботи, права та обов'язки адміністратора безпеки викладено у спеціально розробленій посадовій інструкції [3].

Поряд з інтенсивним розвитком обчислювальних засобів та систем передачі інформації все більш актуальною стає проблема забезпечення її безпеки. Заходи безпеки спрямовані: на запобігання несанкціонованому одержанню інформації, фізичному знищенню або модифікації інформації, що захищається.

Під час аналізу захищеності мобільного додатка проводиться оцінка захищеності трьох основних компонентів: серверної частини, клієнтської частини та каналу зв'язку.

Методи оцінки безпеки клієнтської програми [4]:

- динамічний аналіз:
- налагодження запущеної програми (на емуляторі або пристрої);
- фазинг;
- аналіз мережного трафіку;
- аналіз взаємодії з файловою системою;
- аналіз пам'яті програми.

Статичний аналіз [4]:

- аналіз вихідного коду (якщо є);
- зворотне проектування (Reverse Engineering);
- дизасемблювання;
- декомпіляція;
- аналіз отриманого представлення на слабкі ділянки коду.

Опираючись на багаторічний досвід та експертизу в галузі автентифікації, захисту персональних даних, забезпечення конфіденційності

інформації та безпечної роботи в мережі Інтернет, компанії засобами захисту інформації реалізують розробку та постачання засобів захисту, що відповідають національним та міжнародним стандартам у галузі інформаційної безпеки [2].

Рішення для банківських систем мають ряд специфічних особливостей. По-перше, забезпечення інформаційної безпеки носить цілком явний прикладний характер – протидія шахрайству та захист фінансів. По-друге, необхідність суворої відповідності вимогам українського законодавства та галузевим стандартам. По-третє, масове застосування цих рішень зумовлює вимоги простоти і зручності використання, необов'язковості спеціальних знань та вмінь при забезпеченні безпеки роботи з віддаленими сервісами.

1.2 Загрози інформаційної безпеки банківських інформаційних систем

Сучасні інформаційні технології дозволяють надавати більшість послуг через системи дистанційного банківського обслуговування (ДБО). Традиційно постає питання захисту такого роду сервісів. Відповіддю служить використання рішень, що забезпечують двофакторну автентифікацію.

Діяльність банківської галузі регулюється українським законодавством та міжнародними стандартами. Рішення щодо шифрування даних з використанням українських алгоритмів та захисту баз даних повинні відповідати вимогам регулюючих норм та галузевих стандартів [1-3].

Розвиток технологічних стандартів галузі вимагає від банків забезпечення безпечного (у тому числі віддаленого) доступу працівників до корпоративної мережі та інформаційних ресурсів, а також розмежування та аудиту доступу до баз даних як захист від можливих дій інсайдерів.

Дистанційне банківське обслуговування (ДБО) – загальний термін для технологій надання банківських послуг на підставі розпоряджень, що передаються клієнтом віддаленим чином (тобто без його візиту до банку), найчастіше з використанням комп'ютерних та телефонних мереж.

Сучасний банк немислимий без Інтернет-банку та системи ДБО, тому що тут користувачі працюють не просто з паперами, а з реальними грошима та завдання щодо забезпечення захищеності рішення ДБО виходять на перший план [5].

Підтвердилася тенденція, відзначена експертами у традиційних щорічних звітах у сфері безпеки систем ДБО. Розробники мобільних банк-клієнтів не приділяють достатньої уваги питанням безпеки додатків, не дотримуються посібників з безпечної розробки. Найчастіше відсутні процеси розробки безпечного коду та архітектури. Виявилося, що всі розглянуті програми містять хоча б одну вразливість, що дозволяє або перехопити дані, що передаються між клієнтом і сервером, або безпосередньо експлуатувати вразливість пристрою та мобільного додатка.

Так, 35% мобільних банків для iOS та 15% мобільних банків для Android містять уразливості, пов'язані з некоректною роботою SSL, а це означає можливість перехоплення критичних платіжних даних за допомогою атаки «людина посередині». 22% додатків для iOS потенційно вразливі до SQL-ін'єкції, що створює ризик крадіжки всієї інформації про платежі за допомогою кількох нескладних запитів. 70% програм для iOS і 20% програм для Android потенційно вразливі до XSS - однієї з найпопулярніших атак, що дозволяє ввести в оману користувача мобільного банк-клієнта і таким чином, наприклад, вкрасти його автентифікаційні дані. 45% додатків для iOS потенційно вразливі до XXE-атак. Близько 22% додатків для Android неправильно використовують механізми міжпроцесної взаємодії, тим самим фактично дозволяючи стороннім програмам звертатися до критичних банківських даних [6].

Програми для мобільних пристроїв можна класифікувати за багатьма критеріями, але в контексті безпеки програм розглянемо наступні: за місцем розташування програми та за типом використовуваної технології передачі даних.

За місцем розташування програми [4]:

- SIM-програми – програма на SIM-карті, написана відповідно до стандарту SIM Application Toolkit (STK);
- Web-додатки – спеціальна версія Web-сайту;
- Мобільні програми – програми, розроблені для певної мобільної ОС з використанням спеціалізованого API, що встановлюється у смартфон.

За типом використовуваної технології взаємодії з сервером:

- мережеві програми – використовують власний протокол спілкування поверх TCP/IP, наприклад HTTP;
- SMS-програми – програми на основі SMS (Short Messaging Service).
- програма обмінюється зі сервером інформацією за допомогою коротких текстових повідомлень;
- USSD-програми – програми на основі USSD (Unstructured Supplementary Service Data). Сервіс ґрунтується на передачі коротких повідомлень, схожих на SMS, але має ряд відмінностей;
- IVR-програми – програми, що базуються на технології IVR (Interactive Voice Response). Система базується на заздалегідь записаних голосових повідомленнях та тональному наборі.

Саме програми, розроблені для певної мобільної ОС з використанням спеціалізованого API, що встановлюються в смартфон для взаємодії з відповідним банківським сервісом, зараз найбільш поширені, тому що повністю використовують можливості мобільного апарату і мають найбільш дружній інтерфейс користувача [7].

Загрози банківським інформаційним системам, їх конкретні прояви та причини виникнення представлені у таблиці 1.1.

Таблиця 1.1 – Загрози банківським інформаційним системам [7]

Загрози банківським інформаційним системам	**Прояви загроз банківським інформаційним системам**	**Причина виникнення загроз банківським інформаційним системам**

Навмисне розкрадання/зміна інформації всередині організації	Несанкціонований доступ до секретної чи конфіденційної інформації як банку, так і клієнта та її розкрадання чи спотворення, навмисне псування електронних даних та їх носіїв при їх зберіганні в банку, під час запису або перевезення	шахрайство працівників та низький рівень внутрішнього контролю
Недбалість співробітників, що допустили витік інформації	Впровадження в лінію зв'язку між іншими учасниками розрахунків у системі інтернет-банкінгу як активний таємний ретранслятор, здійсненням платежів за рахунками клієнтів на підставі сфальшованих доручень	низька кваліфікація персоналу та низький рівень внутрішнього контролю
Апаратні та програмні збої	зміна функцій програмного забезпечення поломка апаратури	старіння та знос програмно-технічних засобів, використання неякісного програмного забезпечення
Шкідливі програми	DDoS-атаки («зомбі»-мережа) розсилка вірусів електронною поштою фішингові атаки	старіння та знос програмно-технічних засобів, використання неякісного програмного забезпечення
Зовнішнє фінансове шахрайство	подання себе в якості іншої особи з метою ухилення від відповідальності, або відмови від зобов'язань, або використання прав третьої особи для: надсилання фальсифікованого електронного повідомлення на проведення банківської операції; фальшування або крадіжки ідентифікаційних даних або їх носіїв; фальсифікації авторизації трансакцій або їх підтвердження; зчитування інформації на банківських картах пристрою у банкоматах (скіммінгові пристрої)	високий рівень злочинності у суспільстві та низький рівень інформаційної безпеки банків
Хакерські атаки	провокування інших учасників взаємодії у системі інтернет-банкінгу на порушення правил обміну електронними повідомленнями; фальсифікація або крадіжка даних	високий рівень злочинності у суспільстві та низький рівень інформаційної безпеки банків
Стихійні лиха та катаклізми, вандалізм користувачів	поломка, вихід з робочого стану обладнання банку навмисне псування обладнання банку; недостатня пожежна та технічна безпека приміщень банку	низький культурний рівень у суспільстві

Численні спроби зломів та успішні напади на банківські обчислювальні структури та портали гостро ставлять проблему забезпечення інформаційної безпеки.

Для запобігання цим зловживанням використовуються такі засоби захисту [5]:

- Шифрування вмісту документа.
- Контроль за авторством документа.
- Контроль цілісності документа.
- Нумерація документів.
- Ведення сесій лише на рівні захисту інформації.
- Динамічна автентифікація.
- Забезпечення безпеки секретних ключів.
- Надійна процедура перевірки клієнта під час реєстрації у прикладній системі.
- Використання електронного сертифіката клієнта.
- Створення захищеного з'єднання клієнта із сервером.

1.3 Автентифікація користувача як засіб інформаційної безпеки банківських додатків

Автентифікація використовується для доступу до соціальних мереж, електронної пошти, інтернет магазинів, інтернет-банкінгу, платіжних систем тощо. Автентифікація користувача класифікується за типами представленими на рис. 1.1.

В даний час використання парольної автентифікації є доступним та поширеним через простоту застосування. Цей спосіб захищеності веде до підвищення міцності концепції захисту.

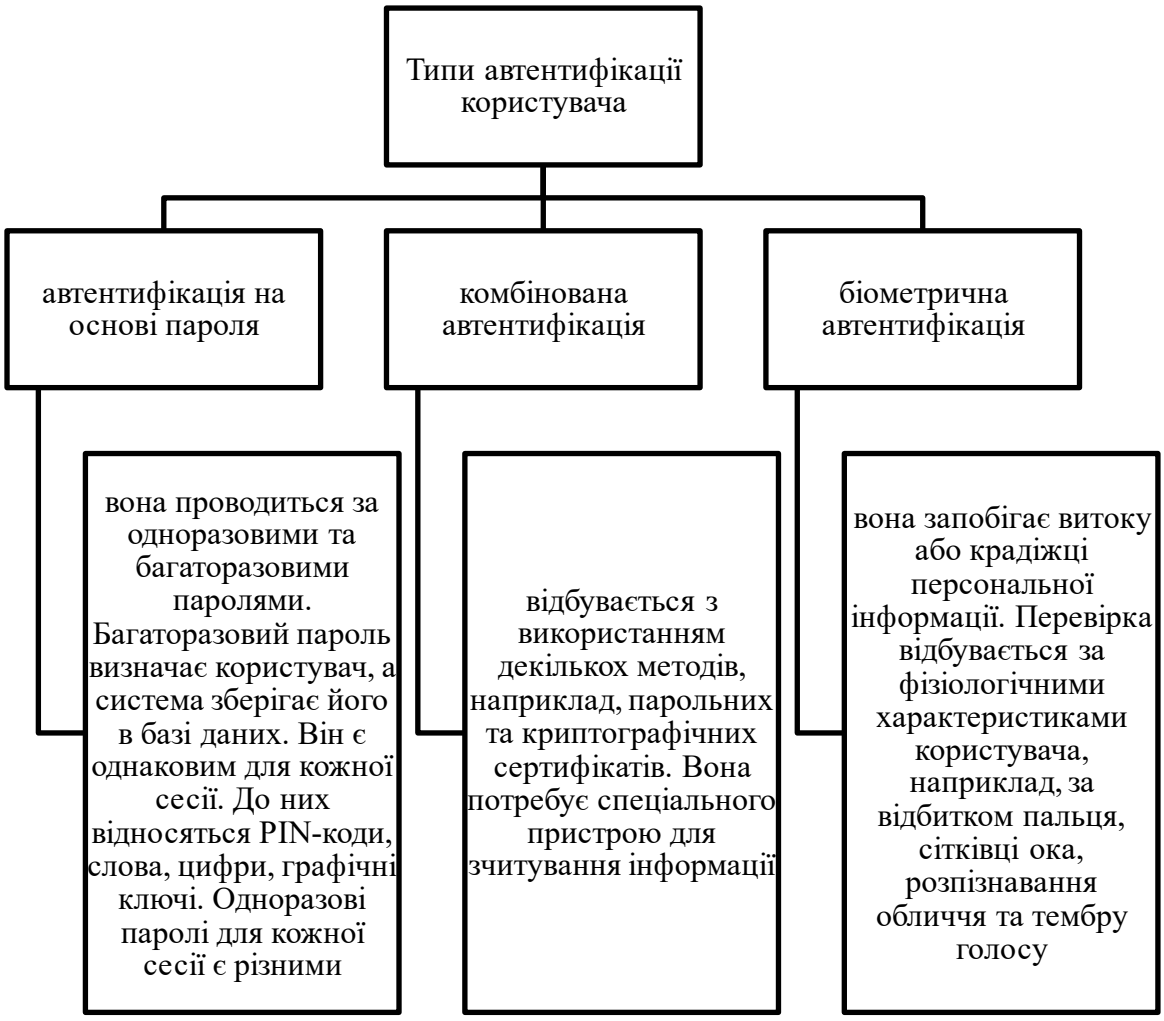


Рисунок 1.1 – Типи автентифікації користувача [8]

Одним з ефективних методів захисту є двофакторна автентифікація для входу в систему. Вона передбачає подвійний захист даних за допомогою прив'язки облікового запису до системи захисту. Після прив'язки користувачеві необхідно буде взаємодіяти з системою для верифікації даних.

На рис. 1.2 представлена схема двофакторної автентифікації, де описані типи, способи реалізації та стійкість [9].

Розглянемо найпоширеніші типи автентифікації користувачів інформаційних систем та відзначимо їх переваги та недоліки.

SMS-код. Після успішної авторизації користувача, на номер телефону надходить SMS з кодом, який діє певний проміжок часу і вводиться в акаунт системи. Повторний вхід можливий при надсиланні нового SMS з кодом.

Перевагою такого методу автентифікації є доступність, так як йде прив'язка до телефонного номера користувача, і сесія для нової генерації пароля повторюється.

Недоліком є відсутність мобільного зв'язку, а також заміна номера.

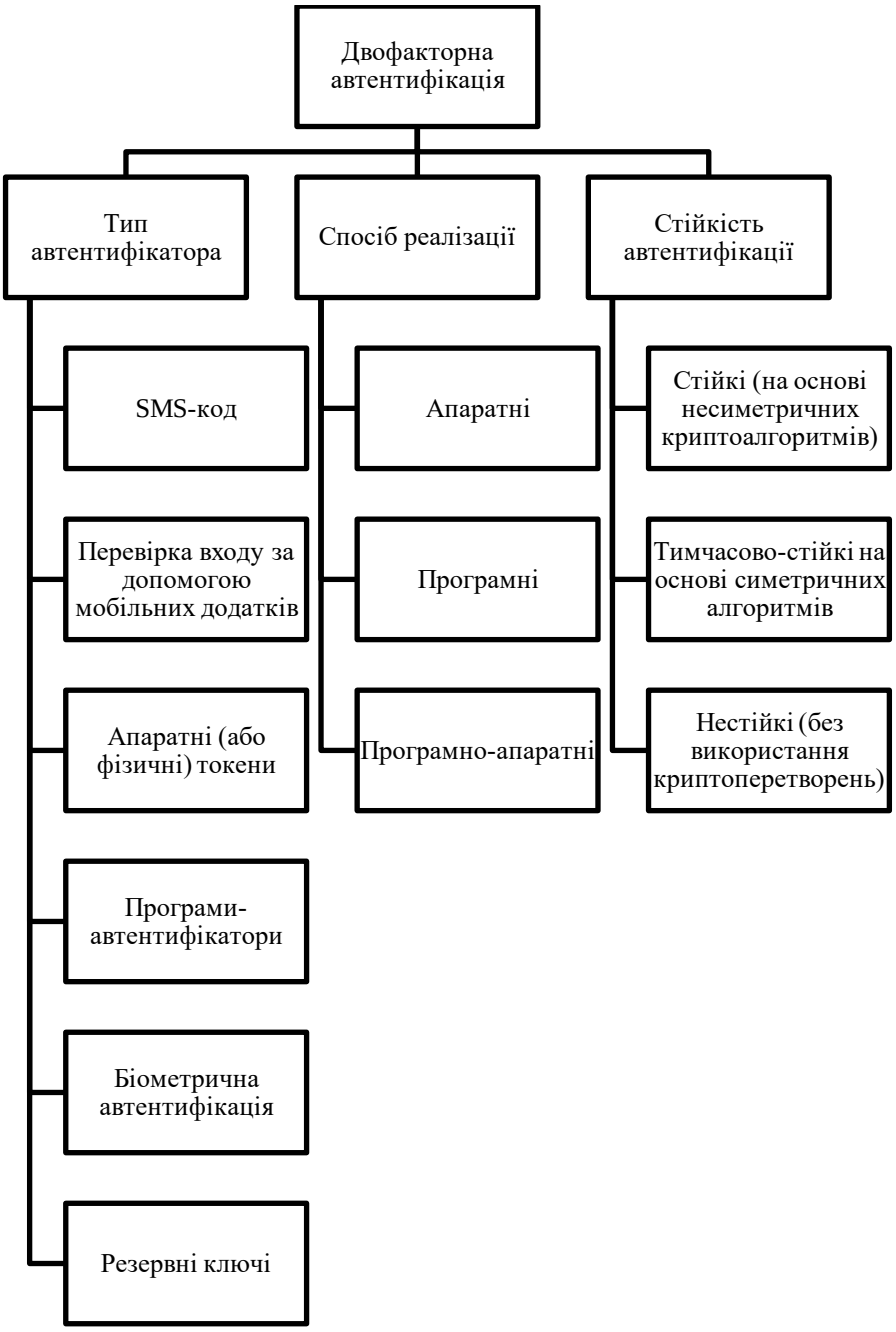


Рисунок 1.2 – Схема двофакторної автентифікації [9]

Відмова від використання SMS для обробки другого фактора полягає у небезпеці цього методу. Мобільні мережі використовують «SignalSystem 7» для взаємодії між собою. Але в цій системі виявлено серйозні вразливості, що дозволяють перехоплювати вхідні дзвінки та SMS абонентів.

Перевірка входу за допомогою мобільних програм. Вхід у систему здійснюється на смартфоні зі встановленим спеціальним додатком.

Смартфон зберігає ключ та забезпечує вхід до інформаційної системи.

Переваги: немає необхідності вводити пароль, не потрібен мобільний зв'язок для отримання SMS-повідомлень та Інтернет.

Недоліки: якщо пройде перехоплення приватного ключа, то можлива фальсифікація особистого номера.

Апаратні (фізичні) токени. Вважається найбільш міцним та надійним методом двофакторної автентифікації – USB ключ. Він має свій процесор, який здійснює генерацію одноразового пароля для автентифікації користувача при приєднанні до комп'ютера. Вибір ключа залежить від певної послуги.

Перевага: незалежний прилад не вимагає смартфона.

Недоліки:

- прилад береться окремо;
- не всі пристрої мають можливість використовувати цей спосіб;
- на один обліковий запис, один токен.

Програми – автентифікатори. Виробляється генерація пароля на пристрої за допомогою розробленої спеціальної програми. У період опції user набуває первинного ключа для генерації одноразового пароля за допомогою криптографічного алгоритму, з конкретним терміном часу, зазвичай до 1 хвилини.

Переваги: потрібен Інтернет з метою відкриття сесії.

Недоліки: ймовірність перехоплення первинного ключа, тоді правопорушник може генерувати подальші паролі.

Біометрична автентифікація. Автентифікація користувача за його унікальними біометричними характеристиками, такими як відбиток пальця,

структура сітківки ока, риси обличчя, голос та ін. При реєстрації біометричного автентифікатора відбувається зчитування та запис зразка відповідної біометричної характеристики користувача за допомогою спеціального пристрою для зчитування. Потім програмний алгоритм обробляє отриманий зразок, і система зберігає його як шаблон бази даних. При подальшій автентифікації, коли користувач пред'являє ідентифікатор біометричний, система порівнює наданий ідентифікатор з шаблоном за допомогою алгоритму зіставлення. Користувач визнається легітимним і отримує доступ тільки в тому випадку, якщо ступінь схожості наданого ідентифікатора зі збереженим у базі даних шаблоном відповідає встановленому пороговому значенню.

Переваги: ідентифікатор невіддільний від людини, його не можна забути, втратити, передати. Перевіривши ідентифікатор, можна говорити про те, що була ідентифікована саме ця людина.

Недоліки: необхідність наявності певних навколишніх умов для проведення біометричної ідентифікації можуть виникати ситуації, коли біометричні ідентифікатори пошкоджені або недоступні для зчитування, а також не дешева вартість сканерів.

Резервні ключі. Це запасний варіант, що використовується у разі втрати мобільного телефону, який служив елементом захисту, оскільки приймав одноразові підтвердження коди для входу в інформаційну систему. При роботі з двофакторною автентифікацією проводиться налаштування, що дозволяє отримувати резервні ключі для їх застосування у позапланових ситуаціях.

1.4 Висновки за розділом

Аналіз сучасних інформаційних систем підтверджує необхідність використання двофакторної автентифікації, як додаткового бар'єру захисту доступності, цілісності та конфіденційності збережених та оброблюваних даних в інформаційній системі. Методи застосування – автентифікатори та перевірка входу за допомогою мобільних додатків є більш захищеними,

практичними та зручними для програмної реалізації інформаційної системи двофакторної автентифікації [10]. Використання цих методів дозволить посилити захист інформації, що зберігається в інформаційній системі банку.

2 МЕТОДОЛОГІЯ ПОБУДОВИ ДВОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ, ЯК ВАЖЛИВОГО ФАКТОРУ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ БАНКІВСЬКИХ ДОДАТКІВ

2.1 Система автентифікації користувача на основі генерації одноразового паролю

Одноразові паролі OTP (One – Time Passwords) – генеруються для одноразового входу до інформаційної системи з використанням програмних чи апаратних методів захисту. Невразливість цих паролів полягає у використанні заданого інтервалу часу. У зв'язку з цим вирішується проблема перехоплення згенерованого одноразового паролю [11].

Навіть якщо зловмиснику вдасться перехопити одноразовий пароль, скористатися ним, у кращому разі, він зможе лише в дуже обмежений проміжок часу, оскільки пароль дійсний лише один раз протягом короткого проміжку часу.

В якості можливих пристроїв для генерації одноразових паролів використовуються різні програмні методики. У загальному випадку для застосування одноразових паролів користувач повинен володіти чимось (мобільний телефон, список паролів тощо), що є автентифікацією «на основі володіння чимось».

Для застосування автентифікації на основі одноразового пароля використовуються криптографічні алгоритми посилення його стійкості.

Дані автентифікації є секретним ключем користувача, що базується на використанні методу шифрування. Секретний ключ зберігається певний час на сервері і доступний клієнту за його успішної ініціалізації в інформаційній системі. Після закінчення часу функціонування пароля здійснюється його повторна генерація.

Для забезпечення двофакторної автентифікації в інформаційних системах одноразові паролі часто використовуються в групі зі збереженими

паролями. В загальному випадку цей алгоритм виглядає так, як продемонстровано на рис. 2.1.

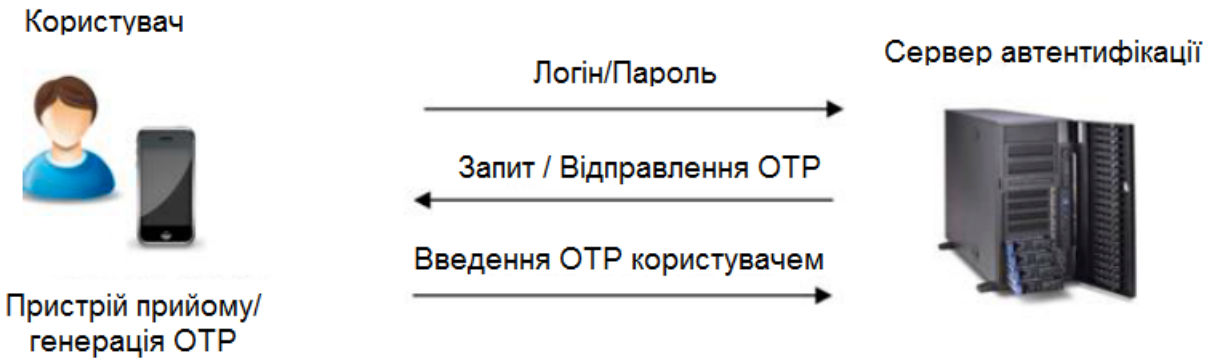


Рисунок 2.1 – Схема двофакторної автентифікації користувача з використанням одноразових паролів [12]

Використання одноразових паролів для проходження двофакторної автентифікації здійснюється таким чином:

- проводиться введення облікових даних користувача (логін/пароль);
- облікові дані надсилаються на сервер автентифікації;
- проводиться перевірка введених користувачем даних на сервері автентифікації;
- на стороні сервера формується запит, який генерує одноразовий код та відправляє його на пристрій;
- одноразовий пароль вводиться користувачем та передає його на сервер;
- сервер проводить перевірку подібності паролів;
- перевірка є задовільною, автентифікація вважається успішною.

Основними джерелами одноразових паролів, що використовуються в інформаційних системах є:

- програмні токени, які генерують одноразові паролі, використовуючи секретний ключ, та поточний момент часу (системний час ОС);
- коди, що генеруються випадковим чином, відображаються користувачеві або у вигляді SMS повідомлення або з використанням іншого каналу зв'язку;
- запис заздалегідь сформованих одноразових паролів або Scratch card.

Секретні ключі користувачів зазвичай зберігаються на сервері, але за необхідності можна використовувати додаткове сховище.

При кожному вході в систему потрібне введення нового одноразового згенерованого паролю з вказаним номером.

Найпоширенішими методами автентифікації є програмні токени, які більш затребувані та вимагають додаткових витрат.

2.2 Характеристика додатків автентифікаторів

Захист баз даних починається з ретельного аналізу неврахованих вразливостей під час здійснення. Часто фактом втрати або злому можуть бути невраховані або не функціонуючі на належному рівні стандартні засоби забезпечення безпеки інформації, що зберігається в базах даних, у зв'язку з тим, що СУБД встановлюється разом з іншими програмами (операційними системами та серверами).

Розглянемо деякі компанії, які надають послуги захисту інформації, що зберігається в БД.

IBM SecurityGuardium. Запобігання витоку інформації з баз даних, сховищ даних та серед великих даних, таких як Hadoop, забезпечує цілісність інформації та автоматизує контроль відповідності в гетерогенних середовищах. Рішення захищає структуровані та неструктуровані дані в базах даних, середовищах великих даних та файлових системах від загроз та забезпечує відповідність вимогам [13].

Також надає масштабовану платформу, яка забезпечує безперервний моніторинг структурованого та неструктурованого трафіку даних, а також застосування політик для доступу до конфіденційних даних у масштабах усього підприємства.

Компанія Imperva. Відома компанія, що займається захистом веб-додатків і систем управління базами даних. Основна мета роботи компанії полягає у проведенні моніторингу та контролю даних, що здійснює перевірку потоку інформації від сховища на сервері до застосування обробки [14].

Imperva Database Security Sphere. Основний вид діяльності – організація захисту інформації, що зберігається в базі даних, мета – забезпечення доступності та цілісності роботи з даними в інформаційній системі [15].

Компанія надає послуги з оптимізації рішень для забезпечення захисту бази даних за рахунок проведення аналізу та сканування мережної активності за допомогою агентів додатків.

McAfee DataCenter Security Suite for Databases. Надає можливість отримувати повну інформацію про загальний стан та рівень захищеності баз даних. Це програмне рішення для захисту баз даних не вимагає ні внесення змін до архітектури, ні встановлення дорогого обладнання. У режимі реального часу рішення виявляє та блокує спроби атак та вторгнень без необхідності відключати бази даних та тестувати програми [16].

McAfee Vulnerability Manager for Databases. Здійснює автоматичне виявлення бази даних у мережі, визначення установок новітніх пакетів виправлень, проведення перевірки на наявність поширених вразливостей [17].

Trustwave AppDetectivePRO. Сканер сховищ даних та великих даних, надає можливість виявити помилки в налаштуваннях, виявити труднощі контролю доступу та ідентифікації користувача, відсутні патчі та небезпечні композиції опцій, можуть бути основою для DoS–атак, розподілу привілеїв користувача та неприпустимої зміни інформації. Простота установки та інтуїтивно зрозумілий інтерфейс AppDetectivePRO не вимагає від користувача експертних знань у галузі баз даних. Усього за кілька хвилин можна миттєво

проаналізувати стан безпеки, отримати оцінку ризиків та скласти звіт про відповідність вимогам для будь-яких БД та сховищ BigData – як локальних, так і хмарних. AppDetectivePRO є чудовим доповненням до інших сканерів мереж та додатків [18].

DbProtect: платформа для забезпечення безпеки даних, що дозволяє миттєво виявляти помилки в конфігурації, проблеми ідентифікації та контролю доступу, відсутні патчі та небезпечні комбінації налаштувань, здатні призвести до атак типу «підвищення привілеїв» або «відмова в обслуговуванні» (DoS–атакам), витоків та несанкціонованої зміни даних. За рахунок використання розрахованої на багато користувачів рольової моделі доступу та розподіленої архітектури з інструментами аналітики та звітності корпоративного класу DbProtect захищає всі реляційні СУБД та сховища BigData в інфраструктурі компанії, розгорнуті як локально, так і в хмарі [19].

FUDO SECURITY – провідна польська компанія, постачальник інноваційних рішень у галузі IT-безпеки. Компанія спеціалізується на управлінні привілейованим доступом, автентифікації та авторизації користувачів, а також перевірці зашифрованого SSL/TLS трафіку [20].

FUDO PAM – ефективне управління та контроль привілейованих користувачів. FUDO PAM – передове рішення, доступне у вигляді фізичного чи віртуального пристрою. Запис сесії, проактивний моніторинг на основі штучного інтелекту, сучасне сховище паролів та бізнес-аналітика – все це в одному пристрої, що встановлюється за кілька годин.

Всі перелічені компанії надають недешеві послуги, у зв'язку з цим кожна компанія може собі дозволити їх використовувати. Для вирішення проблем, пов'язаних з безпекою даних компанії, зазвичай знаходять вигідніший і зручний варіант вирішення проблеми.

В даний час практично всі найбільш значущі ресурси та послуги використовують двофакторну автентифікацію. На першому етапі автентифікації користувач вводить свій логін та пароль, при успішному проходженні цього етапу необхідно пройти другий етап. В якості другого

етапу використовується OTP автентифікація за допомогою SMS або email розсилки, або з використанням програмного генератора паролів, встановленого на мобільний пристрій.

Незважаючи на те, що базова функція у всіх додатків одна і та ж – створення одноразових кодів по тому самому алгоритму, деякі автентифікатори мають додаткові функції або особливості інтерфейсу.

Google Authenticator. Підтримувані платформи: Android, iOS. Google Authenticator є програмою, створеною для автентифікації користувача на основі другого фактора. Google Authenticator не має налаштувань, що полегшує роботу цієї програми [21].

Duo Mobile. Підтримувані платформи: Android, iOS. Duo Mobile має ряд переваг: таких як простота у використанні, мінімалістичний та не вимагає додаткових налаштувань [22]. Для відображення коду потрібний токен.

Microsoft Authenticator. Підтримувані платформи: Android, iOS. Ця програма має можливість робити настройки токена, так що при запуску токен може бути прихований. Microsoft Authenticator – простий у використанні і функціональний [23].

FreeOTP. Підтримувані платформи: Android, iOS. Програмне забезпечення із відкритим кодом [24]. Розмір програми дорівнює 750 Кбайт для платформи iOS, мінімальний з усіх. Програма Google Authenticator має розмір рівний 14 Мбайт. Програма FreeOTP має можливість настроювати токен вручну, чого в інших додатках немає.

Authy. Працює на платформах Android, iOS, Windows, MacOS, Chrome. Ця програма дозволяє використовувати хмарні сервіси для зберігання токенів, що є перевагою. Хмарні послуги дозволяють з будь-яких пристроїв надати доступ до токену. Обов'язковим є те, що програма використовує обліковий запис, прив'язаний до телефонного номера, і без цього робота неможлива. Програма Authy має розмір 44 Мбайт [25].

Розглянемо основні характеристики додатків автентифікаторів.

Незалежність – передбачає існування каналу зв'язку, яким може бути інтернет чи оператор мобільного зв'язку

Потреба синхронізації – дана характеристика має на увазі потребу декодування з сервером автентифікації. Пристрій генерації одноразових паролів (OTP) функціонує самостійно, проте можливе порушення в часі, у цьому випадку отриманий пароль не буде достовірним і необхідно згенерувати його знову

Відновлення автоматизації – дозволяє визначити доступність поновлення до сервісу автентифікаційного приладу, не вдаючись до допомоги IT фахівців, це буде можливим у разі автоматичного коду відновлення

Допоміжний метод автентифікації – демонструє, чи є послуга беззбиткового способу автентифікації, крім застосування програми на смартфоні. Наприклад, не маючи мобільного телефону при собі, можна скористатися списком одноразових кодів для входу в сервіс певної інформаційної системи

Захист програми паролем – відображає, чи реалізований механізм парольного захисту для самого мобільного додатка

Введення даних з клавіатури – наявність цієї властивості говорить про те, що сервіс вимагає введення будь-яких даних за допомогою клавіатурного введення. Наприклад, для використання якогось сервісу необхідно спочатку ввести свій логін і пароль, а потім ввести код автентифікації, отриманий з мобільного додатка

Застосування до інших систем – цей пункт говорить про те, що одна програма може використовуватися для автентифікації на різних ресурсах. Також мається на увазі можливість додавання нових систем

Необхідність реєстрації – якщо є дана характеристика, то обслуговування, надає послугу, передбачає попередню реєстрацію у ньому. Іншими словами, створення облікового запису із завданням логіна та пароля і, можливо, із заповненням інших даних

Передача секретів сервісу – цей параметр свідчить, що сервіс зберігає секретну інформацію користувача, що використовується для його автентифікації

Характеристики додатків двофакторної автентифікації, описані вище, представлені в таблиці 2.1.

Таблиця 2.1 – Характеристики додатків двофакторної автентифікації

Автентифікатор	Google Authenticator	Duo Mobile	Microsoft Authenticator	Free OTP	Authy
Незалежність	+	+	+	+	+
Потреба синхронізації	+	+	+	+	+
Відновлення автоматизації	–	+	+	+	–
Допоміжний метод автентифікації	+	–	–	–	–
Захист програми паролем	–	–	–	+	–
Введення даних із клавіатури	+	+	+	+	+
Необхідність реєстрації	+	+	+	+	+
Передача секретів сервісу	+	+	+	+	+

Розглянувши готові рішення з двофакторної автентифікації, можна зробити висновок, що головним недоліком використання готових рішень є використання одного набору генерації для всіх підключень, а також можливість контролю та доступу до інформації фірмою розробником.

Використання додатку двофакторної автентифікації дозволить посилити захист даних у інформаційній системі. Більшість компаній, які мають доступ до конфіденційної інформації, особистим даним, особливо фінансовим, прагнуть максимально достовірно визначати особистість своїх користувачів.

2.2 Розробка моделі автентифікації користувача на основі другого фактора

Під час розробки системи захисту інформації пропонується модель двофакторної автентифікації (рис. 2.2).

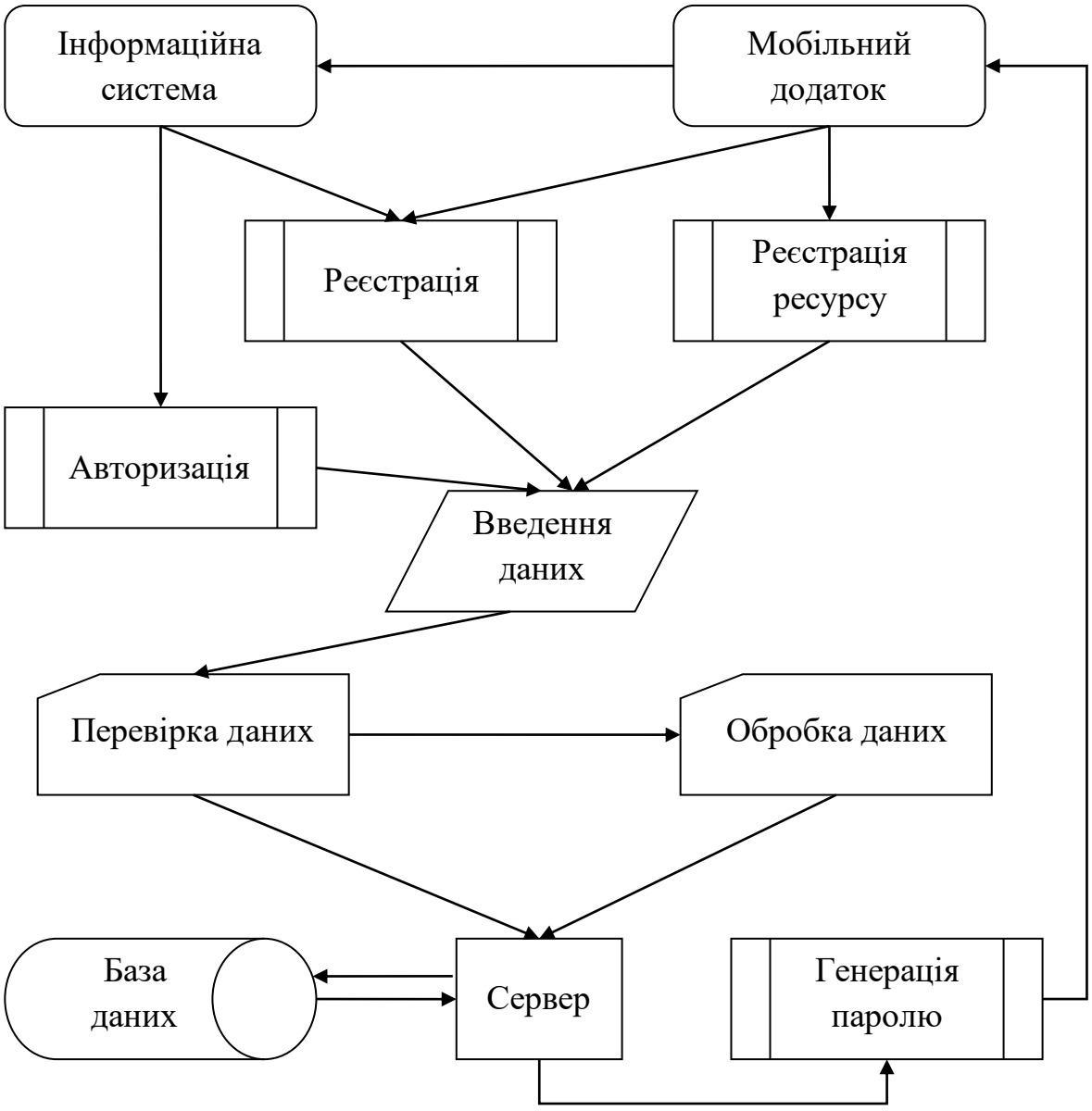


Рисунок 2.2 – Модель запропонованої двофакторної автентифікації

Розроблена модель заснована на двох типах двофакторної автентифікації: програма – автентифікатор та перевірка входу за допомогою мобільних додатків.

Описана модель та алгоритм запропонованого способу захисту інформації в інформаційній системі управління ґрунтуються на використанні комбінації двох факторів: постійного та тимчасового пароля.

Постійний пароль (перший фактор) користувач вибирає сам і використовує під час реєстрації облікового запису.

Перед авторизацією необхідно пройти реєстрацію в програмі. Після цього необхідно запустити програму для введення даних користувача (логіна та пароля), які повинні відповідати зареєстрованим даним. При успішному введенні даних необхідно увійти до програми на мобільному пристрої та ввести початкові дані для генерації тимчасового пароля.

Одноразові паролі, що використовуються як додатковий фактор, генеруються на сервері за описаним вище алгоритмом і діють протягом фіксованого відрізка часу, що дорівнює десяти секунд для одного сеансу автентифікації. Згенерований пароль повторно не використовується, що є бар'єром для зловмисника, який його перехопив. Зазначимо, що довжина пароля дорівнює 7 символам, і він змінюється кожні 20 секунд. У зв'язку з цим для його злому необхідно буде перебрати близько 1000000 паролів, що за вказаний проміжок часу є складним завданням.

2.4 Висновки за розділом

Одним із засобів захисту інформації в інформаційних системах є парольний захист з використанням другого фактора, який є актуальним на сьогоднішній день, оскільки весь банківський сектор та компанії, що займаються питаннями безпеки, використовують двофакторну автентифікацію як додатковий метод захисту. Описано організації, які надають можливість використання двофакторної автентифікації на основі одноразових кодів для захисту інформації. Розглянуто деякі наявні додатки

двофакторної автентифікації та наведено порівняльний аналіз готових додатків. Виділено основні характеристики додатків, що базуються на огляді засобів двофакторної автентифікації. Розроблена модель та алгоритм дозволяє посилити засоби захисту інформаційної системи на основі двофакторної автентифікації.

3 РОЗРОБКА УДОСКОНАЛЕНОГО МЕТОДУ ЗАХИЩЕНОСТІ БАНКІВСЬКИХ ДОДАТКІВ

3.1 Вибір мови програмування

Java – це широко використовувана об’єктно-орієнтована мова програмування та програмна платформа, яка працює на мільярдах пристроїв, включаючи ноутбуки, мобільні пристрої, ігрові консолі, медичні пристрої та багато інших. Правила та синтаксис Java засновані на мовах C і C++ [26].

Java – це:

- одночасно можливо виконувати багато операторів замість того, щоб виконувати їх послідовно;
- базована на класах та об’єктно-орієнтована мова програмування;
- незалежна мова програмування, яка дотримується логіки «Напишіть один раз, запускайте будь-де», тобто скомпільований код може працювати на всіх платформах, які підтримують Java.

Однією з основних переваг розробки програмного забезпечення на Java є його переносимість. Після того, як написаний код для програми Java на ноутбуці, дуже легко перенести код на мобільний пристрій. Коли мова була винайдена в 1991 році Джеймсом Гослінгом із Sun Microsystems (пізніше придбана Oracle), головною метою була можливість «писати один раз, запускати будь-де» [27].

Також важливо розуміти, що Java значно відрізняється від JavaScript. Javascript не потрібно компілювати, тоді як Java-код потрібно компілювати. Крім того, Javascript працює лише у веб-браузерах, тоді як Java можна запускати де завгодно.

Нові та вдосконалені інструменти розробки програмного забезпечення надходять на ринок з надзвичайною швидкістю, витісняючи існуючі продукти, які вважалися незамінними. У світлі цього постійного обороту довговічність Java вражає; більше ніж через два десятиліття після свого створення Java все

ще залишається найпопулярнішою мовою для розробки прикладного програмного забезпечення — розробники продовжують вибирати її замість таких мов, як Python, Ruby, PHP, Swift, C++ та ін.

Java — це технологія, що складається як з мови програмування, так і з програмної платформи. Щоб створити програму за допомогою Java, потрібно завантажити Java Development Kit (JDK), доступний для Windows, macOS та Linux. Після написання програми мовою програмування Java, компілятор перетворює програму в байт-код Java – набір інструкцій для віртуальної машини Java (JVM), які є частиною середовища виконання Java (JRE). Байт-код Java працює без змін у будь-якій системі, яка підтримує JVM, що дозволяє запускати ваш Java-код будь-де.

Програмна платформа Java складається з JVM, Java API та повного середовища розробки. JVM аналізує та запускає (інтерпретує) байт-код Java. Java API складається з великого набору бібліотек, включаючи основні об’єкти, мережеві функції та функції безпеки; генерацію розширюваної мови розмітки (XML); і веб-сервіси. У сукупності мова Java і програмна платформа Java створюють потужну, перевірену технологію для розробки корпоративного програмного забезпечення [28].

Коли справа доходить до вибору мови програмування та середовища для корпоративного додатка, є вагомі технічні причини для розгляду Java, включаючи сумісність, масштабованість та адаптивність.

Основна філософія його створення — сумісність між різними пристроями — залишається найсильнішим аргументом на користь Java для нових корпоративних програм. Об’єктно-орієнтована архітектура Java дозволяє створювати модульні програми та багаторазовий код, скорочуючи цикли розробки та подовжуючи термін служби корпоративних програм.

Масштабованість платформи є ключовим атрибутом Java. За допомогою Java є можливість використовувати одну систему в широкому діапазоні випадків використання. Існуючі настільні програми можна легко адаптувати для роботи на менших пристроях з обмеженими ресурсами. Також можливо

переносити програми з мобільних пристроїв на настільні комп'ютери, розробляючи бізнес-додатки для платформи Android, а потім інтегруючи їх у поточне програмне забезпечення для настільних ПК, минаючи тривалі й дорогі цикли розробки.

Java також обирають для розробки стратегічних планувальників за здатність адаптуватися до нових випадків використання. Наприклад, Java вважається ідеальною платформою для Інтернету речей (IoT). Типовий додаток IoT з'єднує велику кількість різних пристроїв, завдання, яке значно спрощується через те, що мільярди пристроїв працюють на Java. Крім того, розгалужена екосистема розробників Java постійно розробляє та ділиться новими бібліотеками з функціоналом, спеціально націленим на розробку додатків IoT [29].

Java можна використовувати для створення додатків для широкого кола платформ. Настільні комп'ютери, сервери, мобільні телефони, планшети, програвачі Blu-ray, телевізори та веб-браузери використовують Java і розробники можуть писати програми на основі Java для будь-якої з цих платформ. Оскільки Java відповідає вимогам WORA, той самий код можна запускати на всіх платформах з середовищем виконання Java (JRE) без перекомпіляції коду.

Java використовується для написання програм для різних платформ, які запускають JRE, і підтримує програми, які запускаються на одному пристрої, наприклад на настільному комп'ютері або мобільному телефоні. Java також можна використовувати для розробки програм, які працюють розподіленим способом. Це означає, що одна і та ж програма може бути розподілена між серверами або клієнтами в мережі і може виконуватися синхронно. Java також можна використовувати для написання програмних модулів або аплетів як частини веб-сторінок.

Java використовується для [28]:

- GUI програм;
- Веб-сервери та сервери програм;

- Додатки проміжного програмного забезпечення;
- Веб-додатки;
- Мобільні додатки;
- Вбудовані системи;
- Корпоративні програми.

Java розвивалася протягом багатьох років, оскільки Oracle підтримує цю мову та регулярно надає оновлення. Підтримка величезної спільноти розробників також є безсумнівною перевагою для нових програмістів Java. Оскільки Java існує більше двох десятиліть, вона має значну колекцію доступних бібліотек і функцій з відкритим кодом. Розглянемо деякі з ключових переваг мови програмування Java [27].

Проста і легка у навчанні. Java має спільний синтаксис з C і C++. Явні покажчики, перевантаження операторів, класи зберігання та інші елементи, які присутні в C++, недоступні в Java. Це робить її менш складною мовою для написання коду.

Об’єктно-орієнтована мова програмування. Усе в Java розглядається як об’єкт і має супутні функції, такі як клас, інкапсуляція, абстракція, успадкування та поліморфізм.

Багатопоточність підтримується Java. Великі програми можна конвертувати в кілька потоків і виконувати одночасно. Це зменшує ресурси та час, необхідні для виконання програми.

Платформно-агностична мова. Оскільки Java працює в пісочниці віртуальної машини, платформу та архітектуру комп’ютера не потрібно враховувати під час написання програм Java. Один і той самий код може виконуватися різними платформами без перекомпіляції для кожного пристрою, що спрощує керування проектами.

Безпечна платформа. Програми Java виконуються в середовищі виконання. Вона також надає завантажувач класів для завантаження класів у середовище виконання. Це забезпечує буфер і за своєю природою безпечно.

Тим не менш, плагіни для браузера Java вкрай небезпечні, і їх краще деактивувати, оскільки більшість Інтернету зараз працює на JavaScript.

Хоча використання Java має багато переваг, воно не позбавлене недоліків або можливості вдосконалення. Розглянемо деякі з недоліків Java.

Програми повинні запускатися на JRE. Пісочниця Java робить платформу програм агностичною, але це також означає, що програми можна запускати лише поверх JRE, що вимагає більше ресурсів. Споживання пам'яті велике, оскільки програми повинні працювати на віртуальній машині Java.

Інтерфейс користувача, створений за допомогою Java, менш привабливий. Існує кілька фреймворків Java для створення інтерфейсів програм, але жодна з них не є достатньо розвиненою, щоб обробляти складні елементи інтерфейсу користувача, які можна легко досягти за допомогою мов програмування, таких як JavaScript.

Немає резервного засобу. Java не забезпечує резервного копіювання і працює зі сховища.

Збірники сміття, надані з Java, працюють автоматично. Це може здатися перевагою, але це не дає програмістам можливості контролювати збір сміття. Це проблема, коли розширені функції кодуються на Java.

Хоча Java дуже безпечна, плагін для браузера Java вкрай небезпечний і в минулому був причиною інцидентів з кібербезпекою. Найкраще вимкнути плагін, навіть якщо використовується JRE.

Ось кілька додаткових практичних порад, яких слід дотримуватися під час використання Java [28]:

- код у безпечному середовищі;
- плануйте вимоги до об'єктів перед написанням коду;
- дотримуйтесь умов імен (це полегшує читання іншим програмістам у проекті);
- уникайте можливого витоку пам'яті;
- уникайте порожніх блоків;
- уникайте використання циклів з індексами;

- зарезервуйте достатню кількість пам’яті;
- завчасно перевірте Nulls, щоб уникнути винятків нульового покажчика;
- JSON має використовуватися для кодування з Java, а схема повинна бути відома перед декодуванням;
- повторно масштабуйте зображення, щоб використовувати менше ресурсів;
- буферизуйте вихідне зображення та використовуйте повторно масштабоване.

Java — популярна мова програмування, яку легко вивчити, не залежить від платформи та може використовуватися для широкого кола пристроїв. Oracle забезпечує чудову підтримку Java, а велика й зростаюча база програмістів Java також робить її привабливою мовою для використання та кодування.

Технічні аргументи на користь Java є переконливими, але бізнес-причини вибрати Java настільки ж вагомі: великий набір функцій, короткий час навчання та широкий спектр інтегрованих середовищ розробки (IDE).

JetBrains випустила IntelliJ IDEA 2022.1, який представляє аналізатор залежностей, щоб надати інформацію та покращити кодову базу. Покращення зручності та продуктивності включають підказки, запуск команд із файлів Markdown та покращене налагодження та профілювання.

IntelliJ IDEA 2022.1 представляє собою аналізатор залежностей для полегшення управління залежностями та вирішення конфліктів, оновлений майстер створення нових проектів для покращення запуску нових проектів і вікно інструмента сповіщень, яке пропонує новий, спрощений спосіб отримувати сповіщення з IDE. Він також включає ряд інших помітних покращень, які детально розглянуті нижче.

Аналізатор залежностей.

Нещодавно представлений аналізатор залежностей надає вичерпну інформацію про всі залежності Maven і Gradle, які використовуються у

проектах і підпроектах. Він допомагає виявляти та вирішувати конфліктні залежності, відфільтровувати ідентичні залежності та перевіряти, чи вони присутні в різних бібліотеках, а також легко переміщатися між залежностями, щоб виправити конфігурацію збірки.

Розширений майстер створення нового проекту.

Перероблений інтерфейс майстра, щоб спростити створення нових проектів. Є можливість швидко запустити порожній проект; використовувати попередньо налаштовані параметри для Java, Kotlin, Groovy і JavaScript; або використовувати генератори, якщо є більш складні проекти.

Archetype Maven у майстрі нового проекту.

Оновлений генератор проектів Maven Archetype у майстрі нового проекту вводить функціональні можливості пошуку під час перегляду архетипів, можливість керувати каталогом архетипів під час створення модуля та можливість вводити необхідні властивості за архетипом.

Покращені підказки.

Впроваджені покращені підказки для вставки Code Vision, які надають миттєву інформацію про код прямо в редакторі. Список відображених показників тепер включає спадкоємців, використання, авторів коду та пов’язані з ними проблеми. Усі показники тепер увімкнено за замовчуванням і їх можна змінити в налаштуваннях Inlay Hints. Ці налаштування також були оновлені та отримали новий інтерфейс конфігурації.

Рівномірно розділені вкладки.

Є можливість рівномірно розподілити робочий простір між вкладками редактора, щоб усі вони були однакової ширини. Щоб налаштувати це, потрібно перейти до Налаштування | Розширені налаштування | Вкладки редактора | Вирівнюйте пропорції у вкладках.

Можливість експорту діаграми UML в інші формати.

Тепер можна експортувати діаграми UML як файли yEd .graphml, JGraph .drawio, Graphviz .dot, Graphviz .dot, Mermaid .md, Plantuml і IntelliJ IDEA .uml, що робить їх сумісними зі сторонніми інструментами.

Нове діалогове вікно пошуку та заміни структури.

Діалогове вікно структурного пошуку та заміни тепер містить список усіх шаблонів, щоб полегшити навігацію між ними. Крім того, діалогове вікно містить піктограму діалогового вікна Pin та змінені прапорці Injected code та Match case.

Безпека.

IntelliJ IDEA 2022.1 тепер може виявляти вразливості залежностей Maven і Gradle, які використовуються у проектах, перевіряючи базу даних Checkmarx SCA і національну базу даних уразливостей, завдяки плагіну Package Checker, який постачається з IntelliJ IDEA Ultimate.

3.2 Розробка програмного забезпечення захисту банківських додатків

Насамперед необхідно підключитися до віддаленого сервера, був вказаний пароль та хост. Програмний код сервера представлений у Додатку А.

Далі було прописано код двофакторної автентифікації (Додаток Б).

Після цього було розроблено вікно авторизації (Додаток В) та вікно реєстрації (Додаток Г).

Реєстрація – це створення нового користувача чи підприємства.

Авторизація – це вхід для вже зареєстрованого користувача.

Авторизацію не слід плутати з автентифікацією – процедурою перевірки легальності користувача або даних, наприклад, перевірки відповідності введеного користувачем пароля до пароля облікового запису в базі даних, або перевірка цифрового підпису листа за ключом шифрування, або перевірка контрольної суми файлу на відповідність. Авторизація ж проводить контроль доступу до різних ресурсів системи в процесі роботи легальних користувачів після успішного проходження ними автентифікації.

Запуск сервера для роботи програми наведено в Додатку Д.

Після цього розроблено банківську програму з двофакторною автентифікацією за допомогою Authy (Додаток Е).

UML (з англійської абревіатура розшифровується як Unified Modeling Language – уніфікована мова моделювання) – це спосіб наочно описати архітектуру, проектування та реалізацію комплексних програмних систем. Код типового додатка включає тисячі рядків, за зв'язками та ієрархіями яких дуже непросто встежити. За допомогою діаграм UML структуру програми можна розділити на компоненти та підкомпоненти.

UML – стандартизована мова моделювання. Вона сумісна з різними мовами програмування та процесами розробки, а тому більшості програмістів не важко зрозуміти і застосувати її на практиці.

Діаграми UML допомагають розробникам:

- оперативно ввести у курс справи нових співробітників чи колег з інших відділів;
- легко зорієнтуватися у вихідному коді;
- ґрунтовно спланувати нові функції, перш ніж взятися за програмування;
- доступною мовою пояснити матеріал аудиторіям із різними рівнями технічної підготовки.

Діаграми розгортання використовуються для візуалізації апаратних процесорів/вузлів/пристроїв системи, каналів зв'язку між ними та розміщення програмних файлів на цьому апаратному забезпеченні.

Діаграма розгортання – це тип UML-діаграми, яка показує архітектуру виконання системи, включаючи такі вузли, як апаратні або програмні середовища виконання, а також проміжне програмне забезпечення, яке їх з'єднує.

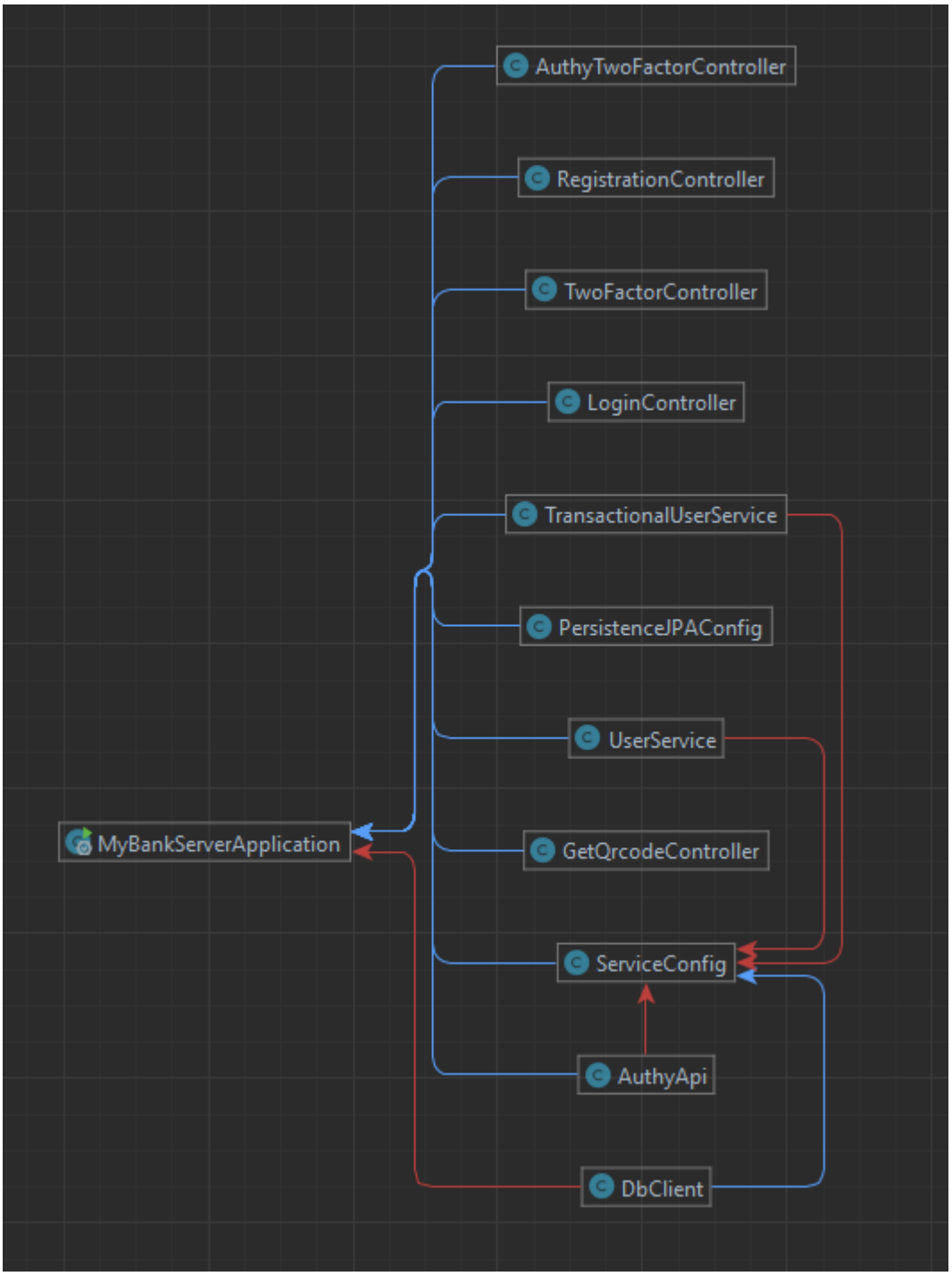


Рисунок 3.1 – UML діаграма серверу

Діаграми розгортання зазвичай використовуються для візуалізації фізичного апаратного та програмного забезпечення системи.

Використовуючи його, можна зрозуміти, як система буде фізично розгорнута на апаратному забезпеченні.

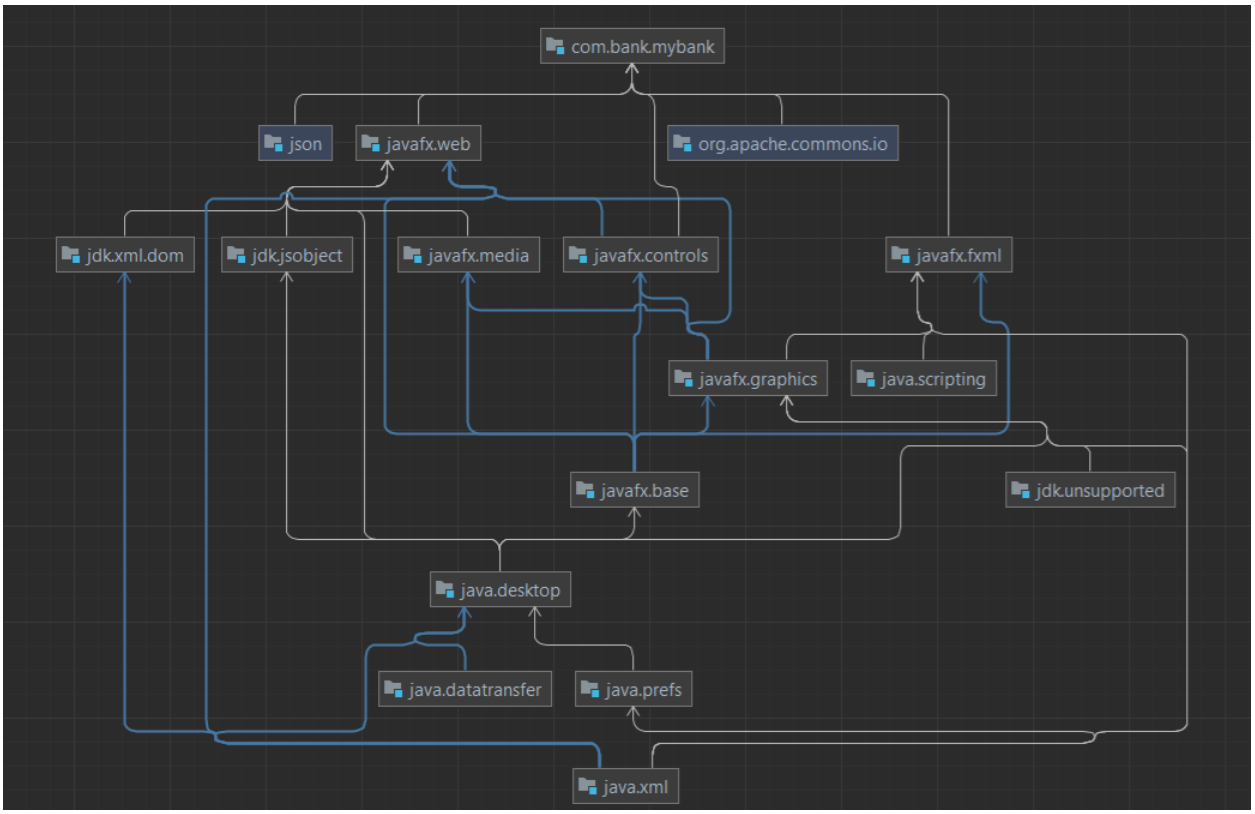


Рисунок 3.2 – UML діаграма програмного забезпечення системи

Діаграми допомагають моделювати апаратну топологію системи порівняно з іншими типами UML-діаграм, які здебільшого описують логічні компоненти системи.

3.2 Розробка бази даних

PostgreSQL — це передова система реляційних баз даних корпоративного класу з відкритим вихідним кодом. PostgreSQL підтримує як SQL (реляційні), так і JSON (нереляційні) запити.

PostgreSQL — це високостабільна база даних, що підтримується більш ніж 20-річною розробкою спільноти з відкритим кодом.

PostgreSQL використовується як основна база даних для багатьох веб-додатків, а також мобільних і аналітичних додатків.

Проект PostgreSQL розпочався в 1986 році на факультеті комп'ютерних наук Берклі Каліфорнійського університету.

Спочатку проект називався POSTGRES, посилаючись на стару базу даних Ingres, яка також була розроблена в Берклі. Метою проекту POSTGRES було додати мінімальні функції, необхідні для підтримки кількох типів даних.

У 1996 році проект POSTGRES був перейменований на PostgreSQL, щоб чітко проілюструвати його підтримку SQL. Сьогодні PostgreSQL зазвичай скорочується як Postgres.

З тих пір група глобальної розробки PostgreSQL, спеціальна спільнота учасників, продовжує випускати версії проекту з відкритим вихідним кодом і безкоштовною базою даних.

Спочатку PostgreSQL був розроблений для роботи на UNIX-подібних платформах. Потім PostgreSQL був розроблений і працював на різних платформах, таких як Windows, macOS і Solaris.

Нижче наведено поширені випадки використання PostgreSQL.

Надійна база даних у стеку LAPP.

LAPP означає Linux, Apache, PostgreSQL і PHP (або Python і Perl). PostgreSQL в основному використовується як надійна внутрішня база даних, яка забезпечує роботу багатьох динамічних веб-сайтів і веб-додатків.

База даних транзакцій загального призначення.

Великі корпорації та стартапи використовують PostgreSQL як основні бази даних для підтримки своїх програм і продуктів.

Геопросторова база даних.

PostgreSQL з розширенням PostGIS підтримує геопросторові бази даних для геоінформаційних систем (ГІС).

PostgreSQL підтримує більшість популярних мов програмування:

- Python;
- Java;
- C#;
- C/C+;
- Ruby;
- JavaScript (Node.js);

- Perl;
- Go;
- Tcl.

PostgreSQL має багато розширених функцій, які пропонують інші системи керування базами даних корпоративного класу, наприклад:

- типи, визначені користувачем;
- спадкування таблиці;
- складний механізм блокування;
- посилальна цілісність зовнішнього ключа;
- подання, правила, під запит;
- вкладені транзакції (точки збереження);
- багатоверсійний контроль паралельності (MVCC);
- асинхронна реплікація.

Останні версії PostgreSQL підтримують такі функції:

- власна версія Microsoft Windows Server;
- таблиці;
- відновлення в момент часу.

У кожному новому випуску додаються нові функції.

PostgreSQL призначений для розширення. PostgreSQL дозволяє визначати власні типи даних, типи індексів, функціональні мови тощо.

Якщо не подобається якась частина системи, завжди є можливість розробити власний плагін, щоб покращити його відповідно до вимог, наприклад, додавши новий оптимізатор.

Багато компаній створили продукти та рішення на основі PostgreSQL. Деякі представлені компанії: Apple, Fujitsu, Red Hat, Cisco, Juniper Network, Instagram тощо.

Для додатку була створена база даних, що складається з таблиці, представленої на рис.3.3.

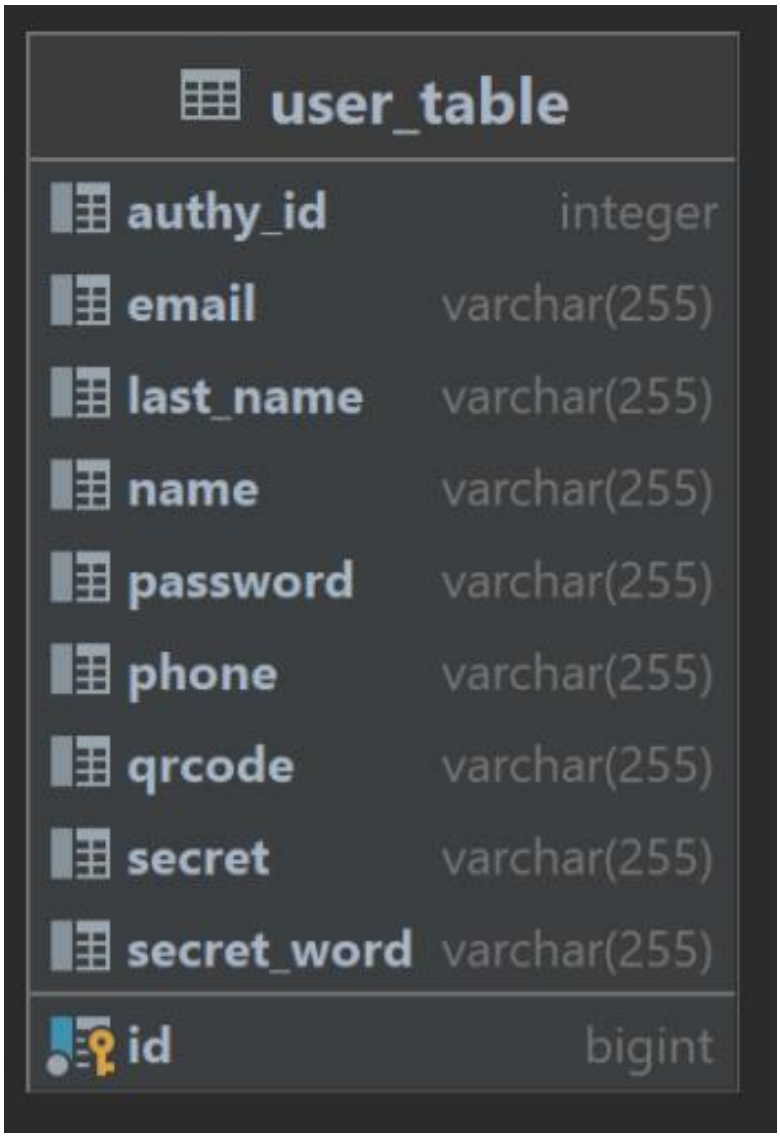


Рисунок 3.3 – База даних додатку

Після реєстрації дані користувача записуються у базу даних (рис. 3.4).

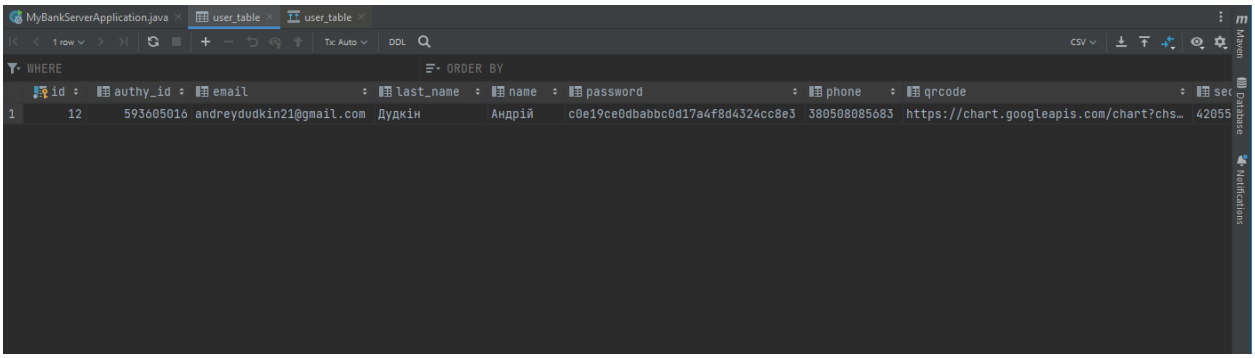


Рисунок 3.4 – Дані користувача в БД розробленого ПЗ

3.3 Висновки за розділом

У цьому розділі наведено результати програмної реалізації запропонованого алгоритму. Розроблений додаток є клієнт-серверною архітектурою та дозволяє повністю реалізувати розроблений алгоритм.

Розглянуто компоненти системи: користувач, мобільний додаток та серверна частина, описані використовувані файли, бібліотеки та фреймворки для реалізації цього алгоритму.

Вхід в банківський додаток виконується з двофакторною автентифікацією користувача: введення логіну та паролю; введення 7-значного коду з програми Authy.

База даних була розроблена за допомогою PostgreSQL. Складається з 1 таблиці, в яку вносяться особисті дані під час реєстрації користувача.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Інструкція користувача програмного забезпечення інформаційної безпеки банківських додатків

Запуск програми.

Вікно входу в банківський додаток представлено на рис. 3.5.

Окно входа.

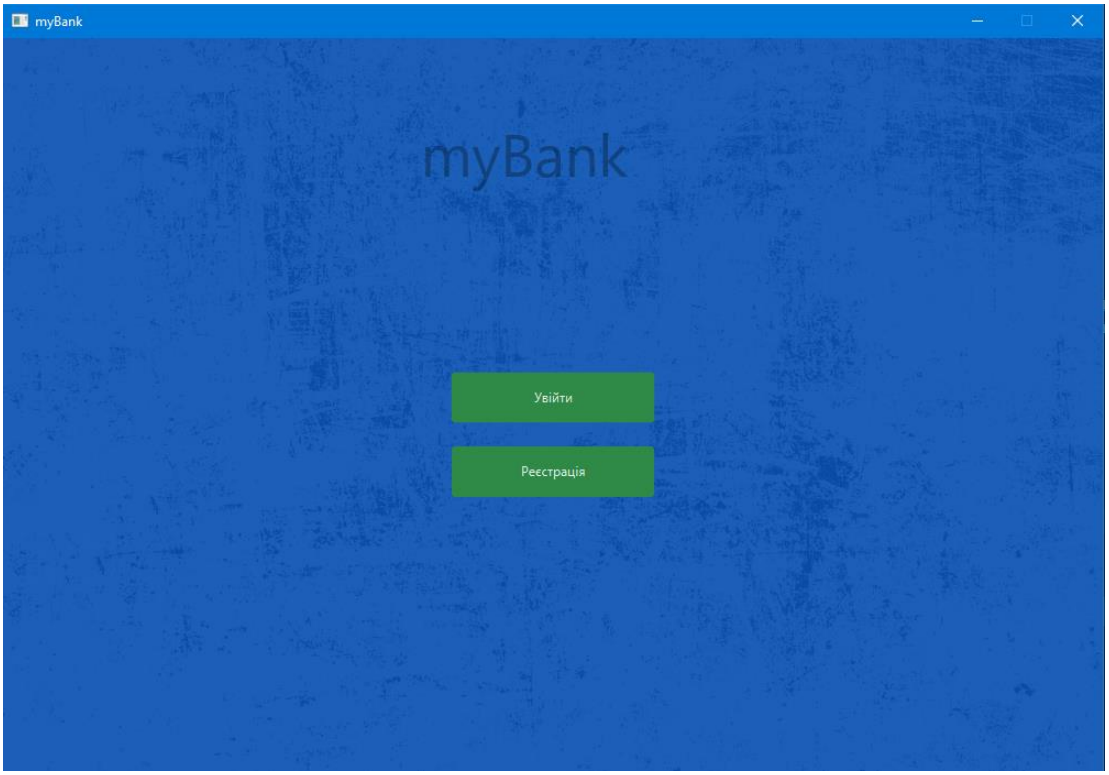
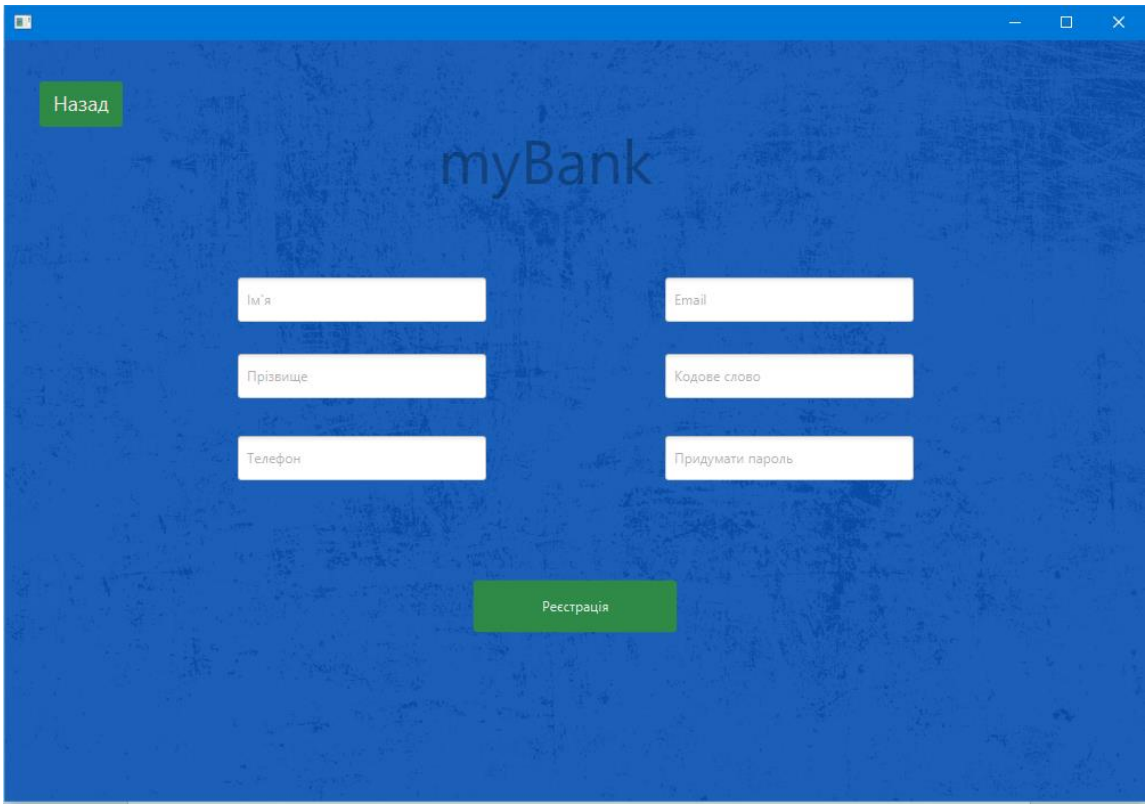


Рисунок 3.5 – Вікно входу в додаток

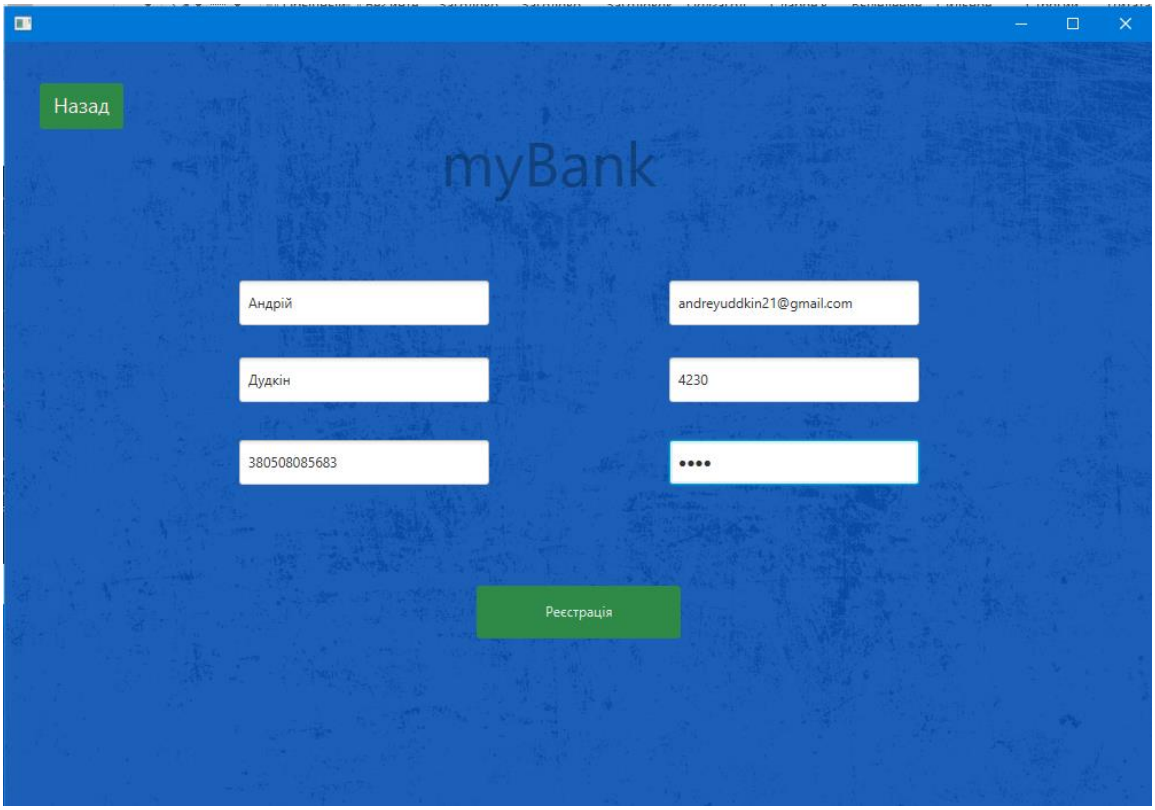
Реєстрація користувача.

При першому запуску програми користувачу потрібно зареєструватися, ввести особисті дані (рис. 3.6).

Важливо відзначити, що телефон вводиться без +, тільки 380 і важливо писати саме свій номер телефону, так як програма, яка скачується в Play Market або App Store, автоматично прив'язує код при введенні номеру.



а)



б)

Рисунок 3.6 – Вікно реєстрації а) пусті поля; б) з заповненими даними

Авторизація.

По-перше у вікні авторизації користувач вводить логін (електронна пошта) та пароль (рис. 3.7.).

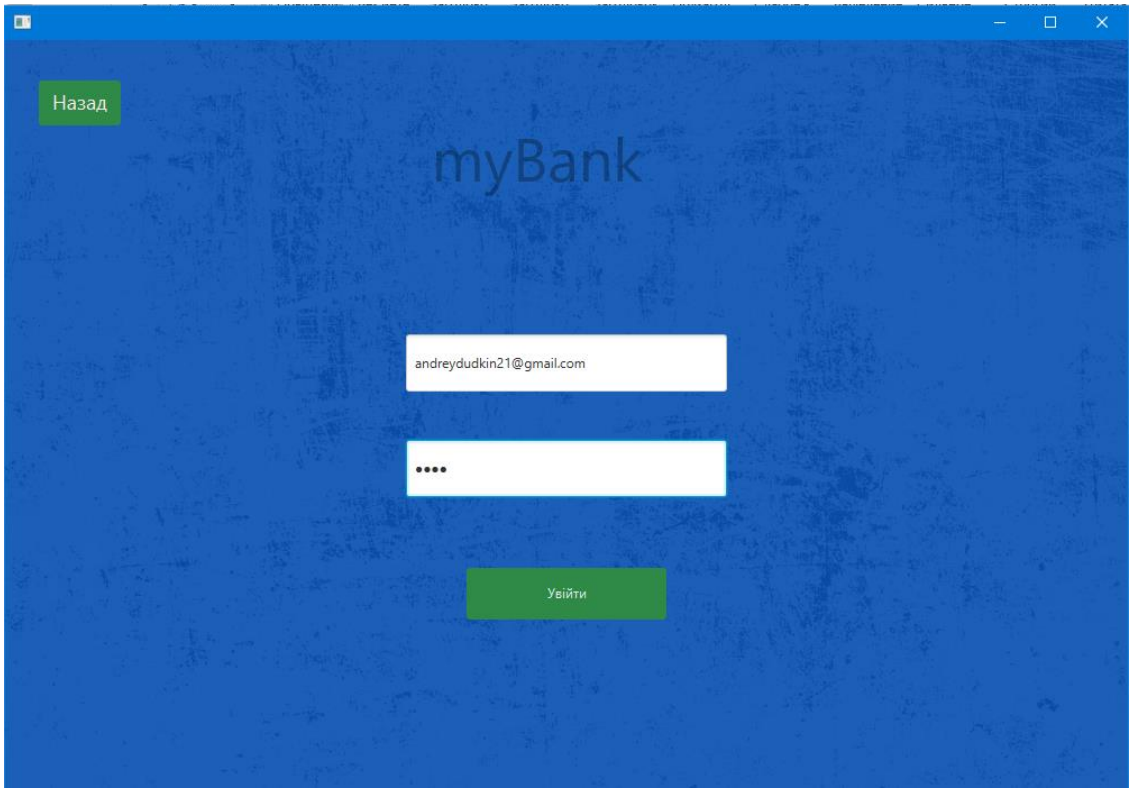


Рисунок 3.7 – Вікно авторизації (логін та пароль)

Після успішного введення логіну та паролю відкривається вікно для авторизації за допомогою Authy (рис. 3.8).

Для цього потрібно, щоб на мобільному телефоні був встановлений Twilio Authy. Найкращий спосіб керувати всіма своїми обліковими записами 2FA – це використовувати додаток Authy. Це дає змогу мати єдиний мобільний додаток для всіх облікових записів 2FA користувача, і він має змогу синхронізувати їх між кількома пристроями, навіть маючи доступ до них на комп’ютері.

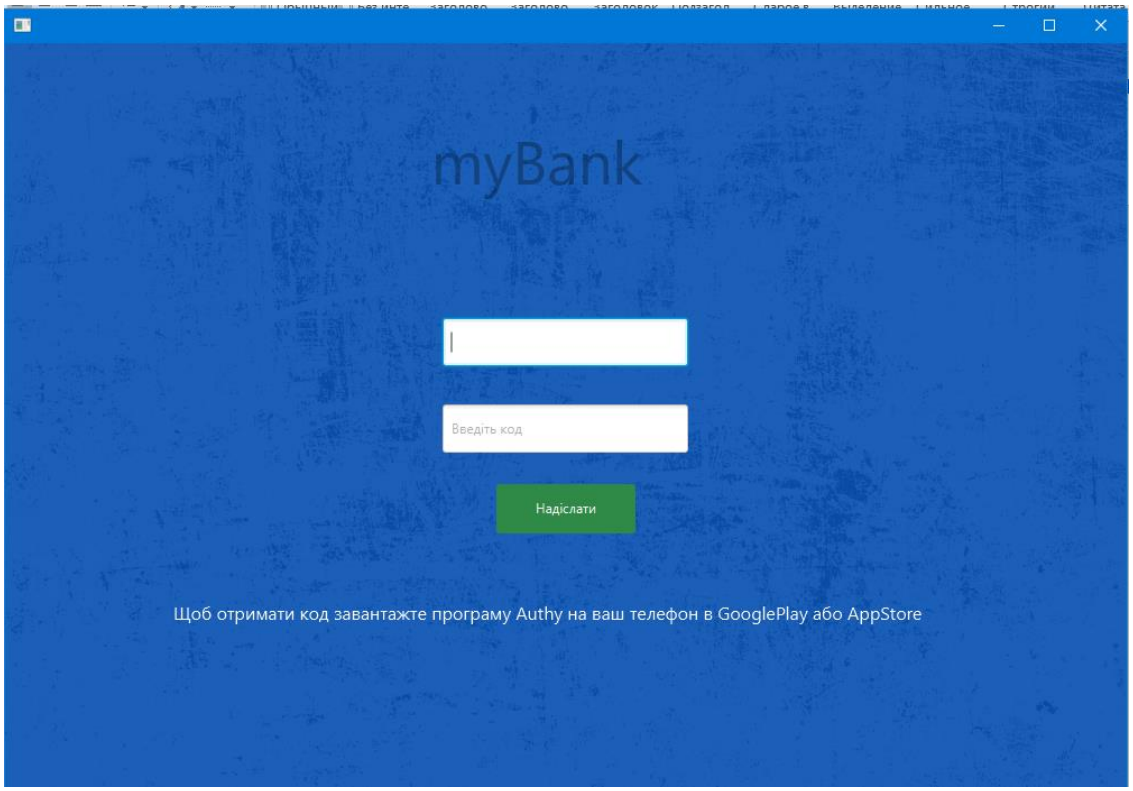


Рисунок 3.8 – Вікно авторизації за допомогою Authy

Використовуючи програму Authy, потрібно ввести код, який автоматично прив'язується за номером (рис. 3.9).

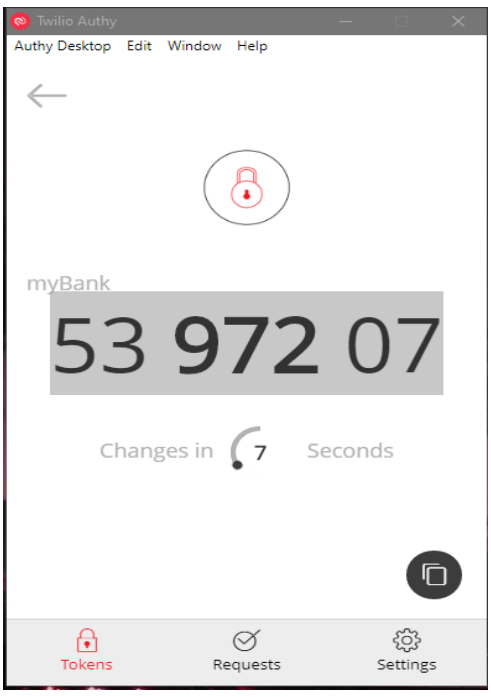


Рисунок 3.9 – Вікно програми Authy

Далі вводимо ці дані у вікно авторизації за допомогою Authy (рис. 3.10).

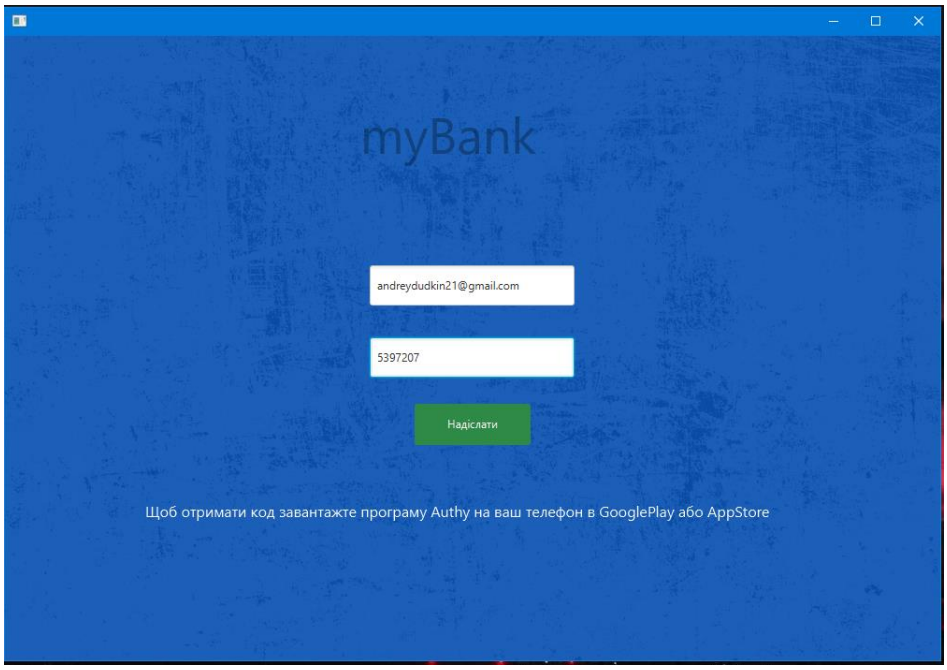


Рисунок 3.10 – Вікно авторизації за допомогою Authy з введеними даними

Вхід в банківський додаток.

Після успішної двофакторної автентифікації користувач входить в банківський додаток (рис. 3.11).

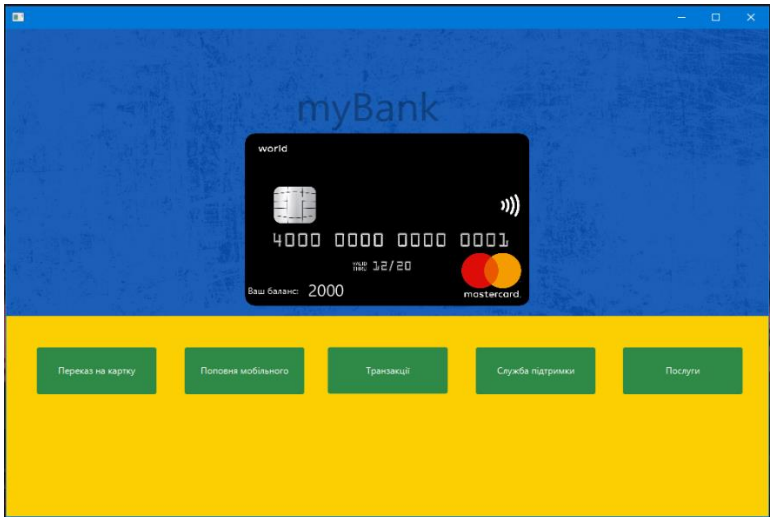


Рисунок 3.11 – Вікно банківського додатку

ВИСНОВКИ

Дане дослідження полягало у розробці та дослідженні алгоритму автентифікації користувачів банківських додатків на основі другого фактора.

Здійснено аналіз відомих алгоритмів та протоколів двофакторної автентифікації, криптографічних алгоритмів та додатків для захисту інформації в інформаційній системі.

Розроблено алгоритм двофакторної аутентифікації на основі програми – автентифікатора та мобільного телефону, що генерує ускладнений набір функцій одноразового пароля для кожної окремої банківської системи. Запропонована система може бути впроваджена також у закриту мережу.

Для розробки банківського додатку була використана мова програмування Java. На основі виконаної розробки були побудовані UML діаграми серверу та програмного забезпечення системи.

Вхід в банківський додаток виконується з двофакторною автентифікацією користувача: введення логіну та паролю; введення 7-значного коду з програми Authy.

База даних була розроблена за допомогою PostgreSQL. Складається з однієї таблиці, в яку вносяться особисті дані під час реєстрації користувача.

Отримані результати є новими, мають теоретичну та практичну значимість. Проведені науково-дослідні роботи з розробки та аналізу систем захисту інформації при ідентифікації та аутентифікації користувача на основі двофакторної аутентифікації спрямовані на вирішення завдань із забезпечення інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Положення про організацію заходів із забезпечення інформаційної безпеки в банківській системі України, затверджене Постановою Правління Національного банку України 28.09.2017 № 95 [Електронний ресурс]. – Режим доступу: http://zakon2.rada.gov.ua/laws/show/v0095500-17.

2. Національний банк України. СОУ Н НБУ 65.1 СУІБ 1.0:2010. Стандарт організації України. Настанова. Методи захисту в банківській діяльності. Система управління інформаційною безпекою. Вимоги (ISO/IEC 27001:2005, MOD). Видання офіційне. Київ. Національний банк України. 2010 СОУ Н НБУ 65.1 СУІБ 1.0:2010.

3. Національний банк України. СОУ Н НБУ 65.1 СУІБ 2.0:2010. Стандарт організації України. Настанова. Методи захисту в банківській діяльності. Звід правил для управління інформаційною безпекою. (ISO/IEC 27002:2005, MOD). Видання офіційне. Київ. Національний банк України. 2010 СОУ Н НБУ 65.1 СУІБ 1.0:2010.

4. Ляшенко І.О. Європейські критерії безпеки інформаційних технологій / І.О. Ляшенко // Сучасні інформаційні технології у сфері безпеки та оборони. – 2012. – № 1 (13). – С. 84–86.

5. Остапов С.Е. Технології захисту інформації: навчальний посібник / С.Е. Остапов, С.П. Євсеєв, О.Г. Король. – Х.: ХНЕУ, 2013. – 476 с.

6. Брижко В.М., Фурашев В.М. Інформаційне право та інформаційне законодавство: наукове видання / В.М. Брижко, В.М. Фурашев. – Київ: Видавничий дім “АртЕК”, 2020. 288 с.

7. Пилипчук В.Г., Брижко В.М., Баранов О.А., Мельник К.С. Становлення і розвиток правових основ та системи захисту персональних даних в Україні:Монографія / В.Г. Пилипчук, В.М. Брижко, О.А. Баранов, К.С. Мельник. – К. : ТОВ «Видавничий дім «АртЕк», 2017. – 226 с.

8. Sharma, Seema, "Location Based Authentication" (2005). University of New Orleans Theses and Dissertations. 141.

9. Frequently Asked Questions on FFIEC Guidance on Authentication in an Internet Banking Environment August 15, 2006. [Електронний ресурс]. – Режим доступу: https://www.ffiec.gov/pdf/authentication_faq.pdf.

10. Тарнавський Ю. А. Технології захисту інформації / Юрій Адамович Тарнавський. – Київ, 2018. – 162 с.

11. Конахович Г.Ф., Корченко О.Г., Юдін О.К., Захист інформації в мережах передачі даних: Підручник. – К.: Видавництво ТОВ НВП «ІНТЕРСЕРВІС», 2009. – 714 с.

12. Браіловський М.М., Головень СМ. та інші. - Технічний захист інформації на об'єктах інформаційної діяльності/ За ред. Проф. В.О. Хорошка. -К.: ДУІКТ, 2007. - 178с.

13. IBM Security Guardium. [Електронний ресурс]. – Режим доступу: https://www.ibm.com/security/data-security/guardium.

14. Imperva. [Електронний ресурс]. – Режим доступу: https://www.pacifica.kz/catalog/zashchita-bd/imperva-database-security.

15. Imperva Secure Sphere Data Security. [Електронний ресурс]. – Режим доступу: https://www.imperva.com/ resources/datasheets/DS_SecureSphere_Data-Security.pdf.

16. McAfee DataCenter Security Suite for Databases. [Електронний ресурс]. – Режим доступу:https://www.mcafee.com/enterprise/en-us/assets/data-sheets/ds-data-center-security-suite-databases.pdf.

17. McAfee Vulnerability Manager for Databases. [Електронний ресурс]. – Режим доступу: http://b2b-download.mcafee.com/products/evaluation/database_security/vulnerability_manager_for_databases/vmd_4.5.0/mcafee_vulnerability_manager_for_databases_product_guide_4_5.pdf.

18. Trustwave AppDetectivePRO. [Електронний ресурс]. – Режим доступу: https://www.trustwave.com/en-us/resources/library/documents/trustwave-appdetectivepro.

19. DbProtect. [Електронний ресурс]. – Режим доступу: https://www3.trustwave.com/software/Database-Security/DbProtectUserGuide_649.pdf.

20. FUDOSECURITY. [Електронний ресурс]. – Режим доступу: https://www.fudosecurity.com.

21. Install Google Authenticator. [Електронний ресурс]. – Режим доступу:https://support.google.com/accounts/answer/1066447?co=GENIE.Platform%3DAndroid&hl=en.

22. Secure Authentication With the Duo Mobile App. [Електронний ресурс]. – Режим доступу: https://duo.com/product/multi-factor-authentication-mfa/duo-mobile-app.

23. Microsoft Authenticator. [Електронний ресурс]. – Режим доступу: https:// support.microsoft.com/ru-kz/help/4026727/microsoft-account-how-to-use-the-microsoftauthenticator-app.

24. FreeOTP Authenticator. [Електронний ресурс]. – Режим доступу: https://www.securitylab.ru/software/498773.php.

25. Authy: 2FA and Password less Login. [Електронний ресурс]. – Режим доступу: https://www.twilio.com/docs/authy.

26. Arnold, Ken, et al. The Java Programming Language. 4th ed., Addison-Wesley, 2006.

27. Levenick, James. Simply Java : an Introduction to Java Programming. Charles River Media, 2006.

28. Shubham Singh, Hansraj Yadav, & Amarjeet Singh. (2020). Overview of Java Programming . International Journal of Progressive Research in Science and Engineering, 1(2), 75–77.

29. Java's 20 Years Of Innovation. Forbes https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/?sh=515cce8811d7.

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
С. 14, 16, 19, 23-24,33, 35, 37, …		Невірні переліки
С.14-15		Таблиця 1.1 - невірне продовження
ПЗ		Нещільне заповнення сторінок

Додаток Б

Додатки лістингу програми

Лістинг. Програмний код сервера. Клас А

```
package com.bank.mybankserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MyBankServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(MyBankServerApplication.class, args);
    }

/*    @Bean
    public RequestContextListener requestContextListener() {
        return new RequestContextListener();
    }*/


}
```

Лістинг. Код двофакторної автентифікації. Клас Б

```
package com.bank.mybankserver.web;

import com.bank.mybankserver.service.authy.AuthyApi;
import com.bank.mybankserver.service.db.DbClient;
import lombok.SneakyThrows;
import org.springframework.boot.configurationprocessor.json.JSONObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class AuthyTwoFactorController {

    @Autowired
    private DbClient dbClient;

    @Autowired
    private AuthyApi authyApi;

    @SneakyThrows
    @PostMapping("/factor")
    @ResponseBody
    public String twoFactorCode(@RequestBody String json) {
        JSONObject jsonObject = new JSONObject(json);
        JSONObject response = new JSONObject();

        if (dbClient.findByAuthyId(jsonObject.getString("email")) != null ) {
            response.put("code",
authyApi.verifyCode(dbClient.findByAuthyId(jsonObject.getString("email")),
                    jsonObject.getString("secret")));
        } else {
            response.put("code", "401");
        }

        return response.toString();
    }
}
```

Лістинг. Вікно авторизації. Клас B

```
package com.bank.mybankserver.web;

import com.bank.mybankserver.persistence.UserEntity;
import com.bank.mybankserver.service.db.DbClient;
import lombok.SneakyThrows;
import org.apache.commons.codec.digest.DigestUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.configurationprocessor.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class LoginController {

    @Autowired
    private DbClient dbClient;


    @SneakyThrows // аннатация которая позволяет обработать исключения
    @PostMapping("/login")
    @ResponseBody
    public String login(@RequestBody String json) { // метод, который
позволяет залогиниться пользователю
        JSONObject jsonObject = new JSONObject(json); //принимаем json с
данными для входа
        JSONObject response = new JSONObject();

        UserEntity user =
dbClient.findByEmail(jsonObject.getString("login")); // проверяем есть ли
такой пользователь в базе
        if (dbClient.findByEmail(jsonObject.getString("login")) != null) { //
проверяем пароль и логин
            if (jsonObject.getString("login").equals(user.getEmail()) &&
DigestUtils.md5Hex(jsonObject.getString("password")).equals(user.getPassword(
)) ) {
                response.put("code", 200); // если пользователь прошел
проверку, то формеруем ответ в виде json и возвращаем его
                response.put("status", "successful");
                response.put("qrcode", user.getQrcode());
            } else {
                response.put("status", "error");
                response.put("code", 405); // неправильный пароль или логин
                response.put("message", "Incorrect login or password");
            }
        } else {
            response.put("status", "error");
            response.put("code", 403); // ответ, который вернется в случаи
если пользователь с таким логин не зарегистрирован или же неправильно ввел
логин
            response.put("message", "Credentials entered incorrectly or user
with such credentials does not exist");
        }

        return response.toString();
    }
}
```

Лістинг. Вікно реєстрації. Клас Г

```
package com.bank.mybankserver.web;

import com.bank.mybankserver.persistence.UserEntity;
import com.bank.mybankserver.service.UserService;
import com.bank.mybankserver.service.authy.AuthyApi;
import com.bank.mybankserver.service.db.DbClient;
import lombok.SneakyThrows;
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.validator.EmailValidator;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.configurationprocessor.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class RegistrationController {

    @Autowired
    private DbClient dbClient;

    @Autowired
    private UserService userService;

    @Autowired
    private AuthyApi authyApi;

    @SneakyThrows
    @PostMapping("/registration")
    @ResponseBody
    public String registration(@RequestBody String json) { // контролер
который принимает запрос на регистрацию
        JSONObject jsonObject = new JSONObject(json);
        JSONObject result = new JSONObject();

        EmailValidator emailValidator = EmailValidator.getInstance();
        if (emailValidator.isValid(jsonObject.getString("email"))) { //
проверяем почту
            if (dbClient.findByEmail(jsonObject.getString("email")) == null)
{ // проверяем нет ли уже пользователя с такой почтой в базе
                UserEntity user = new UserEntity();  // создаем пользователя
если он прошел все проверки
                user.setName(jsonObject.getString("name"));
                user.setLastName(jsonObject.getString("lastName"));
                user.setPhone(jsonObject.getString("phone"));
                user.setEmail(jsonObject.getString("email"));

user.setPassword(DigestUtils.md5Hex(jsonObject.getString("password")));
                user.setSecret_word(jsonObject.getString("secret_world"));
                user.setQrcode(userService.generateQRUrl(user));

user.setAuthyId(Integer.parseInt(authyApi.create(user.getEmail(),
user.getPhone(), "380")));
                dbClient.createOrUpdate(user); // создаем его в базе
                result.put("status", "successful");
                result.put("code", 200);  // ответ, который получит
пользователь
            } else {
                result.put("status", "error");
                result.put("code", 402);
```

```
                result.put("message", "User with this address already
exists");
            }
        } else {
            result.put("status", "error");
            result.put("code", 401);
            result.put("message", "Check your address");
        }

        return result.toString();
    }
}
```

Запуск сервера. Клас Д

Лістинг. Банківська програма з двофакторною автентифікацією за допомогою Authy. Клас E

```
package com.bank.mybank;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class BankApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException { // метод который
запускает приложение
        FXMLLoader fxmlLoader = new
FXMLLoader(BankApplication.class.getResource("home.fxml")); // обращаемся к
файлу с дизайном
        Scene scene = new Scene(fxmlLoader.load(), 1000, 667); // установка
ширины и высоты окна программы
        stage.setTitle("myBank"); // смена заголовка окна
        stage.setScene(scene);
        stage.setResizable(false); // убираем возможность растягивать окно
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

```
package com.bank.mybank;

import com.bank.mybank.animation.Animation;
import com.bank.mybank.api.ServerInteraction;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class AuthController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private Button back_field;

    @FXML
    private URL location;
```

```
    @FXML
    private Button entrance;

    @FXML
    private TextField login;

    @FXML
    private PasswordField password_field;

    private final String loginUrl = "http://localhost:8811/login";

    @FXML
    void initialize() {

    }

    @FXML
    void buttonBackAction() { // метод который позволяет вернуться в главное
окно
        back_field.setOnAction(actionEvent -> { // обработка нажатия кнопки
войти
            back_field.getScene().getWindow().hide(); // скрывает текущие
окно
            FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/home.fxml"));
// путь к окну

            try { // обработка возможных исключений
                fxmlLoader.load();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            Parent root = fxmlLoader.getRoot();
            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.show();
        });
    }

    @FXML
    void loginUser() {
        ServerInteraction serverInteraction = new ServerInteraction();
        entrance.setOnAction(actionEvent -> {
            if (login.getText() != null & password_field.getText() != null) {

                JSONObject json = serverInteraction.sendRequest(loginUrl,
serverInteraction.jsonLogin(login.getText(),
                            password_field.getText()));

                try {
                    if (json.getString("status").equals("error")) {
                        getAnimation();
                    } else {
                        entrance.getScene().getWindow().hide(); // скрывает
текущие окно
                        FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/authy.fxml"))
; // путь к окну
```

```
                        fxmlLoader.load();

                        Parent root = fxmlLoader.getRoot();
                        Stage stage = new Stage();
                        stage.setScene(new Scene(root));
                        stage.show();
                    }
                } catch (JSONException | IOException e) {
                    throw new RuntimeException(e);
                }


            } else {
                getAnimation();
            }
        });
    }


    public void getAnimation() {
        Animation name = new Animation(login);
        Animation pass = new Animation(password_field);
        name.play();
        pass.play();
    }
}
```

```
package com.bank.mybank;

import com.bank.mybank.animation.Animation;
import com.bank.mybank.api.ServerInteraction;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import javafx.stage.Stage;
import org.json.JSONObject;

public class TwoFactorController { // контролер в котором обробатываются
действия двухфакторки

    @FXML
    private AnchorPane home_view;

    @FXML
    private Button load_qr;

    @FXML
    private Button next;

    @FXML
    private WebView web_view;

    @FXML
    private TextField world_secret;

    @FXML
```

```
    private TextField email_field;

    private WebEngine engine;

    private String getQrcode = "http://localhost:8811/get_qrcode";

    private String twoFactor = "http://localhost:8811/two_factor";



    public void getQr() { // запрашиваем qrcode на сервере по нажатии кнопки
запросить qr код
        ServerInteraction interaction = new ServerInteraction();
        load_qr.setOnAction(actionEvent -> {
            if (email_field.getText() != null) {
                try {
                    JSONObject json = interaction.sendRequest(getQrcode,
interaction.jsonGetQr(email_field.getText()));
                    if (json.getString("status").equals("error")) {
                        getAnimation();
                    } else {

                        loadPage(urlValid(json.getString("qr")));
                    }
                } catch (Exception e) {
                    throw new RuntimeException(e);
                }
            } else {
                getAnimation();
            }
        });
    }

    public void checkCode() { // проверяем введенный код как только будет
нажата кнопка со стрелкой
        ServerInteraction interaction = new ServerInteraction();
        next.setOnAction(actionEvent -> {
            if (email_field.getText() != null) {
                try {
                    JSONObject json = interaction.sendRequest(twoFactor,
interaction.jsonCheckCode(email_field.getText(), world_secret.getText()));
                    if (json.getString("status").equals("error")) {
                        Animation animation = new Animation(world_secret); //
вызываю анимацию если код неверный
                        animation.play();
                    } else {
                        // переходим на окно кабинета, если все хорошо
                        next.getScene().getWindow().hide(); // скрывает
тикущие окно
                        FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/final.fxml"))
; // путь к окну

                        fxmlLoader.load();

                        Parent root = fxmlLoader.getRoot();
                        Stage stage = new Stage();
                        stage.setScene(new Scene(root));
                        stage.show();
                    }
                } catch (Exception e) {
                    throw new RuntimeException(e);
```

```
                }
            } else {
                getAnimation();
            }
        });
    }

    public void loadPage (String url) { // загружаем страницу с qr кодом
        engine = web_view.getEngine();
        engine.load(url);
    }

    public void getAnimation() {

        Animation email = new Animation(email_field);

        email.play();

    }

    public static String urlValid(String url) { // метод для валидации ссылки
        char[] arr = url.toCharArray();
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] != '\\') {
                builder.append(arr[i]);
            }
        }
        return builder.toString();
    }
}
```

```
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import com.bank.mybank.animation.Animation;
import com.bank.mybank.api.ServerInteraction;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import org.json.JSONException;
import org.json.JSONObject;

public class AuthyController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextField email_field;

    @FXML
    private AnchorPane home_view;
```

```
    @FXML
    private TextField secret_field;

    @FXML
    private Button send;

    private String twoFactor = "http://localhost:8811/factor";


    @FXML
    void initialize() {
        ServerInteraction interaction = new ServerInteraction();
        send.setOnAction(actionEvent -> {
            if (email_field.getText() != null & secret_field != null) {
                JSONObject jsonObject = interaction.sendRequest(twoFactor,
interaction.sendRequestAuthy(email_field.getText(),
                        secret_field.getText()));
                try {
                    if (jsonObject.getString("code").equals("200")) {
                        // переходим на окно кабинета, если все хорошо
                        send.getScene().getWindow().hide(); // скрывает
тикущие окно
                        FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/final.fxml"))
; // путь к окну

                        fxmlLoader.load();

                        Parent root = fxmlLoader.getRoot();
                        Stage stage = new Stage();
                        stage.setScene(new Scene(root));
                        stage.show();
                    } else {
                        getAnimation();
                    }
                } catch (JSONException | IOException e) {
                    throw new RuntimeException(e);
                }
            } else {
                getAnimation();
            }
        });
    }

    private void getAnimation() {
        Animation email = new Animation(email_field);
        Animation secret = new Animation(secret_field);
        email.play();
        secret.play();
    }

}
```

Код регистрации

```
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import com.bank.mybank.animation.Animation;
import com.bank.mybank.api.ServerInteraction;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import org.json.JSONException;
import org.json.JSONObject;

public class RegistrationController {

    private final String registrationUrl =
"http://localhost:8811/registration";

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private PasswordField create_pass;

    @FXML
    private Button back_field;

    @FXML
    private TextField last_name_field;

    @FXML
    private TextField mail_field;

    @FXML
    private TextField name_field;

    @FXML
    private TextField phone_field;

    @FXML
    private TextField secret_field;

    @FXML
    private Button reg_field;

    @FXML
    void initialize() {


    }

    @FXML
    void buttonBackAction() { // метод который позволяет вернуться в главное
окно
        back_field.setOnAction(actionEvent -> { // обработка нажатия кнопки
войти
            back_field.getScene().getWindow().hide(); // скрывает тикущие
окно
            FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/home.fxml"));
// путь к окну

            try { // обработка возможных исключений
```

```
                fxmlLoader.load();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            Parent root = fxmlLoader.getRoot();
            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.show();
        });
    }

    @FXML
    void singUpUser() { // обробатываем кнопку регистрации
        ServerInteraction serverInteraction = new ServerInteraction();
        reg_field.setOnAction(actionEvent -> {
            if (name_field.getText() != null & last_name_field.getText() !=
null & phone_field.getText() != null &
            mail_field.getText() != null &  secret_field.getText() != null &
create_pass.getText() != null) {
                try {
                    JSONObject json =
serverInteraction.sendRequest(registrationUrl,
serverInteraction.jsonRegistration(name_field.getText(),
                            last_name_field.getText(), phone_field.getText(),
mail_field.getText(), create_pass.getText(), secret_field.getText()));

                    if (json.getString("status").equals("error")) { //
проверяем ответ сервера, если есть ошибка выводим анимацию
                        getAnimation();
                    } else {
                        reg_field.getScene().getWindow().hide(); // скрывает
тикущие окно
                        FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/bank-
view.fxml")); // путь к окну

                        fxmlLoader.load();

                        Parent root = fxmlLoader.getRoot();
                        Stage stage = new Stage();
                        stage.setScene(new Scene(root));
                        stage.show();
                    }
                } catch (IOException e) {
                    throw new RuntimeException(e);
                } catch (JSONException e) {
                    throw new RuntimeException(e);
                }
            } else {
                getAnimation();
            }
        });
    }


    public void getAnimation() { // получаем анимацию к указанным полям
        Animation name = new Animation(name_field);
        Animation lastName = new Animation(last_name_field);
        Animation phone = new Animation(phone_field);
        Animation email = new Animation(mail_field);
        Animation secret = new Animation(secret_field);
```

```
        Animation password = new Animation(create_pass);
        name.play();
        lastName.play();
        phone.play();
        email.play();
        secret.play();
        password.play();
    }
}
```

Клас Є

Лістинг. Кнопка «Вхід»

```
package com.bank.mybank;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class HomeController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button home_auth_butt;

    @FXML
    private Button home_reg_butt;

    @FXML
    private AnchorPane home_view;

    @FXML
    void initialize() {
        home_auth_butt.setOnAction(actionEvent -> { // обработка нажатия
кнопки войти
            home_auth_butt.getScene().getWindow().hide();
            FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/bank-
view.fxml")); // путь к окну

            try { // обработка возможных исключений
                fxmlLoader.load();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            Parent root = fxmlLoader.getRoot();
            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.showAndWait();
        });
    }

    @FXML
    void showRegistrationsView() { // метод который позволяет вернуться в
главное окно
        home_reg_butt.setOnAction(actionEvent -> { // обработка нажатия
кнопки войти
            home_reg_butt.getScene().getWindow().hide(); // скрывает тикущие
```

```
окно
            FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("/com/bank/mybank/registration.
fxml")); // путь к окну

            try { // обработка возможных исключений
                fxmlLoader.load();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            Parent root = fxmlLoader.getRoot();
            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.showAndWait();
        });
    }
}
```

Додаток В

Матеріали презентації

Тема: Забезпечення інформаційної безпеки банківських додатків

Спецчастина: Розробка модулю автентифікації й захищеного входу в систему засобами Java.

Автор: ст. гр. КІБ-18 Дудкін А.О.

Керівник: Маслова Н.О., к.т.н., доц.

Об’єкт та предмет дослідження

- **Актуальність:** На сьогоднішній день важко уявити роботу підприємств та установ без комп'ютерних мереж та баз даних. Державні, комерційні та особисті секрети довіряються комп'ютерам, у зв'язку з чим виникає потреба у надійних засобах безпеки для захисту інформаційних даних. Найбільш надійним способом є двофакторна автентифікація (2FA) – автентифікація, у процесі якої використовуються автентифікаційні фактори кількох типів. В цьому випадку зловмисник не зможе отримати доступ до даних, тому що йому доведеться не тільки підглянути пароль, але й пред'явити фізичний пристрій, крадіжка якого, на відміну від крадіжки пароля, завжди швидко виявляється.

- **Об’єкт дослідження:** захист інформації банківських додатків.

- **Предмет дослідження:** методи захисту в мобільних додатках банку.

Мета та задачі роботи

Мета кваліфікаційної роботи — аналіз захищеності інформації, розробка удосконаленого методу захисту додатків

Задачі роботи

- провести аналіз сучасних систем захисту інформації в банківській сфері;
- проаналізувати загрози інформаційної безпеки банківських мобільних додатків;
- розробити програмний модуль двофакторної автентифікації в мобільні застосування банківської системи.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Загрози банківським інформаційним системам	**Прояви загроз банківським інформаційним системам**	**Причина виникнення загроз банківським інформаційним системам**
Навмисне розкрадання/зміна інформації всередині організації	Несанкціонований доступ до секретної чи конфіденційної інформації як банку, так і клієнта та її розкрадання чи спотворення, навмисне псування електронних даних та їх носіїв при їх зберіганні в банку, під час запису або перевезення	шахрайство працівників та низький рівень внутрішнього контролю
Недбалість співробітників, що допустили витік інформації	Впровадження в лінію зв'язку між іншими учасниками розрахунків у системі інтернет-банкінгу як активний таємний ретранслятор, здійсненням платежів за рахунками клієнтів на підставі сфальшованих доручень	низька кваліфікація персоналу та низький рівень внутрішнього контролю

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Апаратні та програмні збої	зміна функцій програмного забезпечення поломка апаратури	старіння та знос програмно-технічних засобів, використання неякісного програмного забезпечення

Хакерські атаки	провокування інших учасників взаємодії у системі інтернет-банкінгу на порушення правил обміну електронними повідомленнями; фальсифікація або крадіжка даних	високий рівень злочинності у суспільстві та низький рівень інформаційної безпеки банків
Стихійні лиха та катаклізми, вандалізм користувачів	поломка, вихід з робочого стану обладнання банку навмисне псування обладнання банку; недостатня пожежна та технічна	низький культурний рівень у суспільстві

Типи автентифікації користувача

Схема двофакторної автентифікації

МЕТОДОЛОГІЯ ПОБУДОВИ ДВОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ, ЯК ВАЖЛИВОГО ФАКТОРУ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ БАНКІВСЬКИХ ДОДАТКІВ

Одноразові паролі OTP (One – Time Passwords) – генеруються для одноразового входу до інформаційної системи з використанням програмних чи апаратних методів захисту. Невразливість цих паролів полягає у використанні заданого інтервалу часу. У зв'язку з цим вирішується проблема перехоплення згенерованого одноразового паролю

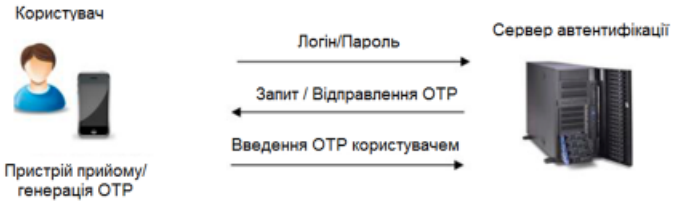
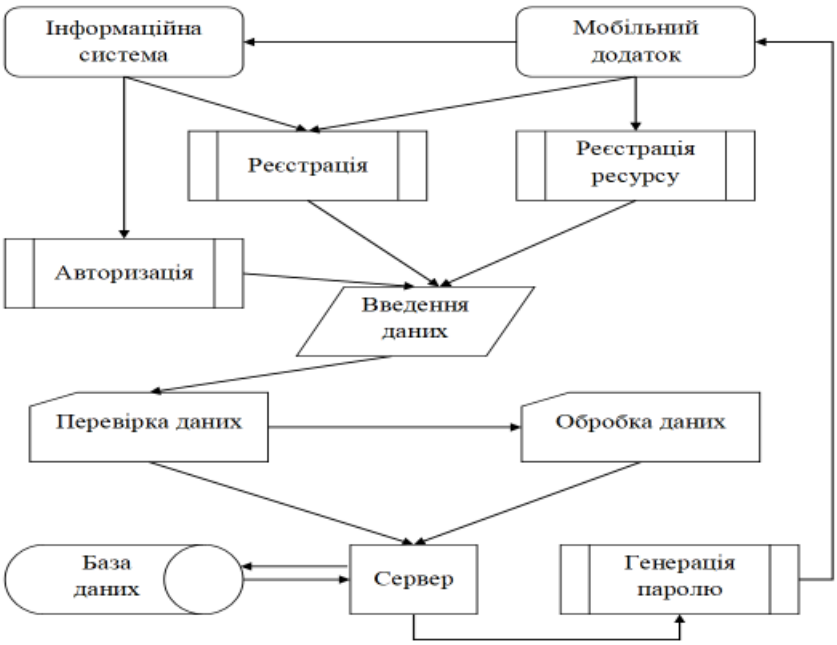


Схема двофакторної автентифікації користувача з використанням одноразових паролів

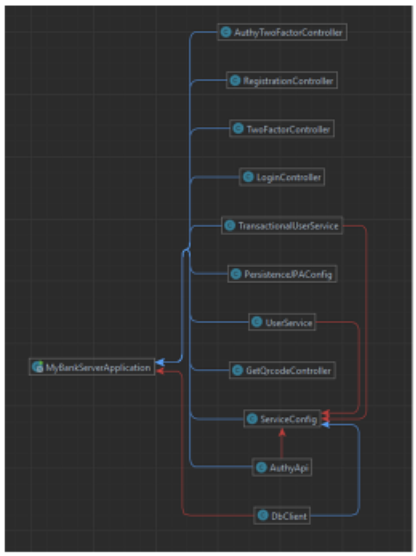
Характеристики додатків двофакторної автентифікації

Автентифікатор	Google Authenticator	Duo Mobile	Microsoft Authenticator	Free OTP	Authy
Незалежність	+	+	+	+	+
Потреба синхронізації	+	+	+	+	+
Відновлення автоматизації	–	+	+	+	–
Допоміжний метод автентифікації	+	–	–	–	–
Захист програми паролем	–	–	–	+	–
Введення даних із клавіатури	+	+	+	+	+
Необхідність реєстрації	+	+	+	+	+
Передача секретів сервісу	+	+	+	+	+

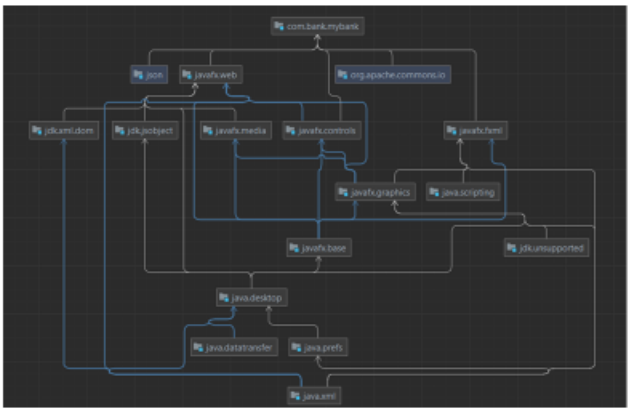
Модель запропонованої двофакторної автентифікації



РОЗРОБКА УДОСКОНАЛЕНОГО МЕТОДУ ЗАХИЩЕНОСТІ БАНКІВСЬКИХ ДОДАТКІВ

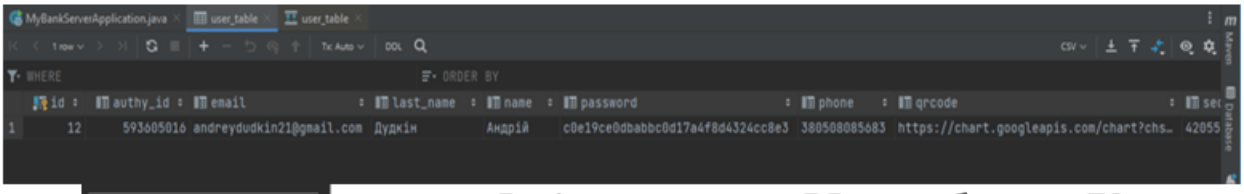


UML діаграма серверу

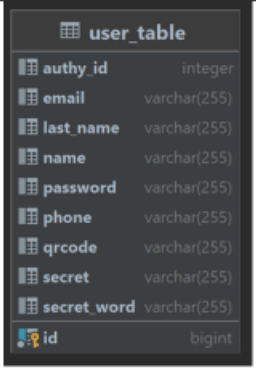


UML діаграма програмного забезпечення системи

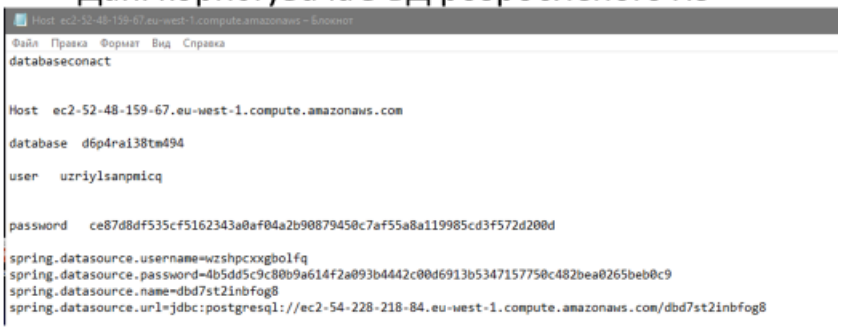
База даних



Дані користувача в БД розробленого ПЗ

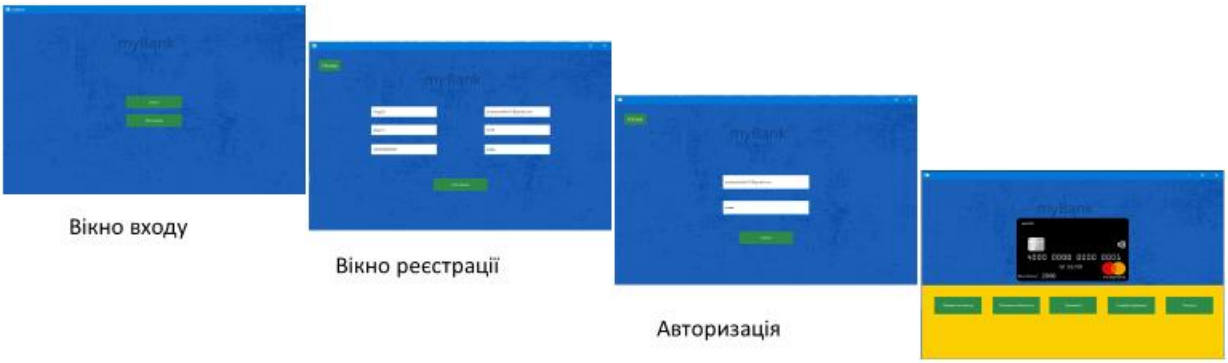


База даних додатку (PostgreSQL)



Дані для входу у базу даних

Тестування модулю



Вікно входу

Вікно реєстрації

Авторизація

Автентифікація успішна!

Отриманні результати

- Здійснено аналіз відомих алгоритмів та протоколів двофакторної автентифікації, криптографічних алгоритмів та додатків для захисту інформації в інформаційній системі.
- Розроблено алгоритм двофакторної аутентифікації на основі програми – автентифікатора та мобільного телефону, що генерує ускладнений набір функцій одноразового пароля для кожної окремої банківської системи.

Отримані результати є новими, мають теоретичну та практичну значимість. Проведені науково-дослідні роботи з розробки та аналізу систем захисту інформації при ідентифікації та аутентифікації користувача на основі двофакторної аутентифікації спрямовані на вирішення завдань із забезпечення інформаційної безпеки.