НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв»

Виконав:
студент IV курсу, групи КП-91
Максименко Дмитро Юрійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Цуркан Василь Васильович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ІСТ ФІОТ, д.т.н, доцент,
Дорогий Ярослав Юрійович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Максименку Дмитру Юрійовичу

1. Тема проєкту «Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв», керівник проєкту Цуркан Василь Васильович, к.т.н., доцент, затверджені наказом по університету від 31 травня 2023 р. № 2107-с.

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - специфікація вимог до вебзастосунку формування моделі загроз безпеці мобільних пристроїв;
   - концептуальна модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв;
   - об'єктно-орієнтована модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв;
   - робочий проєкт вебзастосунку формування моделі загроз безпеці мобільних пристроїв.

5. Перелік обов'язкового графічного матеріалу:
   - статична логічна структура вебзастосунку (креслення);
   - динамічна логічна структура вебзастосунку (креслення);
   - фізична структура компонентів вебзастосунку (креслення);

- фізична структура вузлів вебзастосунку (креслення);
- вимоги до вебзастосунку (плакат);
- варіанти використання вебзастосунку (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «26» грудня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Специфікування вимог до вебзастосунку формування моделі загроз безпеці мобільних пристроїв	10.02.2023	
2.	Розроблення концептуальної моделі вебзастосунку формування моделі загроз безпеці мобільних пристроїв	31.03.2023	
3.	Розроблення об’єктно-орієнтованої моделі вебзастосунку формування моделі загроз безпеці мобільних пристроїв	15.04.2023	
4.	Розроблення робочого проєкту вебзастосунку формування моделі загроз безпеці мобільних пристроїв	25.05.2023	
5.	Оформлення додатків до дипломного проєкту	03.06.2023	

Студент Дмитро МАКСИМЕНКО

Керівник проєкту Василь ЦУРКАН

АНОТАЦІЯ

Дипломний проєкт присвячено програмній реалізації вебзастосунку формування моделі загроз безпеці мобільних пристроїв.

Проналізовано процес формування моделі загроз безпеці мобільних пристроїв. За його результатами виокремлено існуючі рішення і зіставлено їх відповідно до встановлених критеріїв. Унаслідок такого зіставлення визначено їхні переваги та недоліки. Це дозволило специфікувати вимоги як з точки зору функціонального, так і нефункціонального аспектів.

Розроблено концептуальну модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв. Для цього даний процес інтерпретованої як діяльність і формалізовано за допомогою діаграми в графічній нотації IDEF0. Тоді як процеси, потоки даних і зв’язки між ними діаграмами в графічних нотаціях IDEF3 та DFD відповідно.

Розроблено об’єктно-орієнтовану модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв. Визначено його варіанти використання відповідно до специфікованих вимог. Для їх задоволення розроблено статичну та динамічну логічну структуру, а також фізичну за допомогою діаграм UML.

Розроблено робочий проєкт вебзастосунку формування моделі загроз безпеці мобільних пристроїв. Це досягнуто завдяки аналізуванню і обиранню відповідних засобів програмування, а саме: Java, Spring Framework, TypeScript, Angular, Project Lombok, RxJS та PostgreSQL. Отриманий вебзастосунок протестовано для перевіряння стабільності та належності його функціонування. Насамкінець продемонстровано використання вебзастосунку для формування моделі загроз безпеці мобільних пристроїв.

ABSTRACT

Diploma project is dedicated to creation of a threat model for mobile devices generation web application.

An analysis of the process of threat model for mobile devices generation process itself was done. Based on this analysis, some existing solutions were found and compared according to defined criteria. Based on the performed comparison their advantages and disadvantages were defined. It allowed to specify the requirements, both in terms of functional requirements and non-functional requirements.

Conceptual model was developed for the mobile device threat model for mobile devices generation web application. For this purpose, the process was interpreted as activity and formalized as an IDEF0 diagram. Processes, data flow and connections between them were formalized via IDEF3 and DFD notations respectively.

Object-oriented model was developed for the mobile device threat model for mobile devices generation web application as well. Use cases were defined according to specified requirements. To satisfy them static and dynamic logic structures was developed, as well as physical one using UML diagrams.

A working project of the mobile device threat model for mobile devices generation web application was developed. This was achieved thanks to analyzing and choosing fitting development tools, them being: Java, Spring Framework, TypeScript, Angular, Project Lombok, RxJS and PostgreSQL. Resulting web application was tested to check stability and correctness of its functioning. Lastly, mobile devices generation web application usage was demonstrated.

ДП.045440-01-90 Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для формування	5	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Технічне завдання		
ДП.045440-03-81	Вебзастосунок для формування	100	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Пояснювальна записка		
ДП.045440-04-51	Вебзастосунок для формування	4	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Програма та методика		
ДП.045440-05-34	Вебзастосунок для формування	17	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Керівництво користувача		
ДП.045440-06-99	Вебзастосунок для формування	1	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Статична логічна структура		
	вебзастосунку. Діаграма класів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Вебзастосунок для формування	1	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Динамічна логічна		
	структура вебзастосунку.		
	Діаграма діяльності		
ДП.045440-08-99	Вебзастосунок для формування	1	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Фізична структура компонентів		
	вебзастосунку. Діаграма		
	компонентів		
ДП.045440-09-99	Вебзастосунок для формування	1	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Фізична структура вузлів		
	вебзастосунку. Діаграма		
	розгортання		
ДП.045440-10-98	Вебзастосунок для формування	1	
	моделі загроз безпеці		
	мобільних пристроїв.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖЕНО

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

Технічне завдання

ДП.045440-02-91

ПОГОДЖЕНО

Керівник проєкту:

_____ Василь ЦУРКАН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро МАКСИМЕНКО

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Нині невід’ємною частиною нашого життя стали мобільні пристрої. З кожним роком вони ускладнюються все більше й більше як апаратно, так і програмно. З ускладненням зростає і кількість атак з боку зловмисників, аби викрасти інформацію, завдати шкоди тощо. З огляду на це обумовлюється важливість розуміння потенційних загроз безпеці мобільних пристроїв, а також шляхів зменшення ризиків, що пов’язані з їхнім реалізуванням. Тож розроблення вебзастосунку формування моделі загроз є актуальним.

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для формування моделі загроз безпеці мобільних пристроїв на основі введеної моделі пристрою та створення чек-листа на основі сформованої моделі. В ньому можна буде проставляти відмітки, а також надавати особам з відповідною роллю можливість керувати інформацією в системі задля підтримання безпечності її використання, аби сформовані моделі відповідали актуальному стану в галузі інформаційних технологій і кібербезпеки зокрема.

4. ВИМОГИ ДО ВЕБЗАСТОСУНКУ

Вебзастосунок, що розробляється, має відповідати таким вимогам:

1. Для звичайних користувачів має бути наявна реєстрація.

2. Користувачі мають мати можливість авторизуватися.
3. Користувачі можуть сформувати модель загроз.
4. Користувачі можуть керувати власними чек-листами.
5. Модератори можуть керувати пристроями, загрозами та пом’якшеннями.
6. Адміністратори можуть керувати модераторами.

Вебзастосунок реалізується шляхом використанням мови програмування Java, фреймворку Spring Framework та бібліотеки Lombok для серверної частини, мови програмування TypeScript та фреймворку Angular для клієнтської частини та системи керування базами даних PostgreSQL.

Додаткові вимоги:

1. Сумісність: вебзастосунок має бути сумісним з популярними браузерами
2. Адаптивність: вебзастосунок має адаптуватися під розмір екрану
3. Безпека: передбачається використання вбудованих засобів для запобігання SQL-ін’єкціям, використання односторонньої функції для зберігання паролів, перехоплення та обробка помилок, використання змінних середовища та використання CORS-політики
4. Права доступу: має бути реалізовано перенаправлення авторизованих користувачів, які намагаються зайти на сторінки авторизації та реєстрації, та неавторизованих користувачів, які намагаються зайти на сторінки для авторизованих користувачів
5. Доступність: вебзастосунок має підтримувати українську мову

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту розробляється така документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

– статична логічна структура вебзастосунку;

– динамічна логічна структура вебзастосунку;

– фізична структура компонентів вебзастосунку;

– фізична структура вузлів вебзастосунку;

5) плакати:

– вимоги до вебзастосунку;

– варіанти використання вебзастосунку.

6. ЕТАПИ ПРОЄКТУВАННЯ

Специфікування вимог до вебзастосунку формування моделі загроз безпеці мобільних пристроїв 30.12.2022

Розроблення концептуальної моделі вебзастосунку формування моделі загроз безпеці мобільних пристроїв 18.02.2023

Розроблення об’єктно-орієнтованої моделі вебзастосунку формування моделі загроз безпеці мобільних пристрої 20.04.2023

Розроблення робочого проєкту вебзастосунку формування моделі загроз безпеці мобільних пристроїв.....25.05.2023

Оформлення додатків дипломного проєкту 04.06.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого вебзастосунку виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖЕНО

Завідувач кафедри

_____Євгенія СУЛЕМА

«___» _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

Пояснювальна записка

ДП.045440-03-81

ПОГОДЖЕНО

Керівник проєкту:

_____ Василь ЦУРКАН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро МАКСИМЕНКО

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Вебзастосунок – програмний застосунок, доступний через мережу Інтернет за допомогою браузера;

Операційна система – програмне забезпечення найнижчого рівня, базовий комплес програм, що дозволяє комп'ютеру чи іншому пристрою працювати;

Вебтрафік – дані, що надсилаються та отримуються через мережу Інтернет;

Android – мобільна операційна система, розроблена компанією Google, яка використовується на смартфонах, планшетах, розумних годинниках тощо більшості виробників;

iOS – мобільна операційна система, розроблена компанією Apple для її власних смартфонів;

HarmonyOS – мобільна операційна на базі Android, розроблена компанією Huawei для власних пристроїв;

Бенефіціар – той, хто має з чогось користь;

Патч – виправлення програмного забезпечення, яке випускається автором цього програмного забезпечення;

Ентерпрайз – у контексті даної роботи корпоративне застосування, тобто для ентерпрайзу це для організацій;

ICS – автоматизована система керування;

Блог – особиста сторінка користувача, або ж групи людей, який/яка може розміщати на ній дописи;

Гайд – роз'яснення/інструкція стосовно виконання певних дій, часто написаний зрозумілою та неформальною мовою;

Моніторинг – відстежування стану;

PDF – формат файлів електронних документів;

Чек-лист – перелік, в якому можна проставляти відмітки;

Конфігурація – у контексті програмного забезпечення – його налаштування;

Авторизація – вхід у систему; процес визначення прав особи системою;

Браузер – програма для перегляду сторінок та вебсайтів у мережі;

SQL-ін’єкція – вид нападу, коли в поля вводу чи поля запиту додається код мовою SQL, який може бути небезпечним у разі, якщо в системі символи вхідної інформації не екрануються, тобто якщо не додаються спеціальні символи, які позначають, що символ, який увівся, відноситься до вмісту а не системного виклику;

Одностороння функція – така функція, результат якої легко обчислити, проте дуже важко отримати вхідні дані за результатом;

Клієнт/фронтенд – у контексті даної роботи це клієнтська частина програми, тобто частина, з якою взаємодіє користувач;

Сервер/бекенд – серверна частина програми, частина, в якій знаходиться бізнес-логіка та яка зберігає дані;

Змінні середовища – ідентифікатори системи, на якій запускається програмне забезпечення, за якими в системі зберігаються певні чутливі чи повторювані дані, або ж такі які мають бути легкозмінними, для того щоб не зберігати ці дані в самому програмному забезпеченні;

CORS-політика – Cross-Origin Resource Sharing-політика, механізм захисту, який полягає в тому, що частина програмного забезпечення блокує запити, які надходять з доменів, не визначених системою;

Домен – ідентифікатор вебсайту/вебсторінки;

Кібербезпека – галузь, яка розглядає безпеку комп’ютерних пристроїв, зокрема й мобільних;

Хостинг/хост – надання дискового простору та обчислювальних потужностей іншій особі, наприклад, під запуск застосунку;

Декомпозиція – розбиття на складові частини;

JWT-токен – певний автоматично сформований рядок, який використовується для перевірки дійсності особи, що здійснює запити;

Кукі – записи в сховищі браузера, де зберігаються певні дані, які потрібні для уникнення повторення дій, наприклад, авторизації при кожній дії в системі;

Віртуальна машина – змодельована обчислювальна машина з виділеними ресурсами, на одній реальній машині може запускатися декілька віртуальних;

СУБД – система управління базою даних;

Бібліотека – у контексті роботи – бібліотека для мови програмування, певний набір готових засобів, який спрощує написання;

Фреймворк – надбудова до мови програмування, що містить багато корисних функцій та спрощень;

БД – база даних;

Оператор – у контексті даної роботи – певна функція, яка застосовується до об’єкта, який має цей оператор.

ВСТУП

Нині спостерігається тенденція до зростання темпів технологічного розвитку. Хоча комп'ютер як такий з'явився досить давно, проте поточний стан технологій дав можливість використовувати блага прогресу в максимально різноманітних речах. Що далі, то більша кількість пристроїв різних типів входять до звичного життя та рутини і стають нормою. Так, лише 16 років тому було представлено перший смартфон, а зараз він уже кілька років як є основним мобільним пристроєм для більшості людей.

Утім, з розвитком мобільних пристроїв суттєво підвищується їхня складність, і це стосується не лише апаратного забезпечення та процесу виготовлення, а й програмного забезпечення та його розроблення. Оскільки відомо, що простішою є річ, то важче завдати їй шкоди. Таким чином, програмне забезпечення мобільних пристроїв розвинулося до рівнів, які важко було колись уявити для повноцінних комп'ютерів, а відтак, оскільки кількість можливостей стрімко зросла, так сам стрімко зріс і обсяг потенційного інтересу зловмисників до таких пристроїв та інформації, що ними обробляється. Тож це призвело до зростання кількості вразливостей.

Оскільки спостерігається перехід суспільства до «мобільного», що обумовлено насамперед стрімким ростом кількості мобільних пристроїв, що використовуються. Так, станом на січень 2023 року, 92.3% користувачів Інтернету хоча б частину часу в мережі проводять використовуючи телефон, 56.9% часу онлайн складає час, проведений зі смартфонів, а також на них припадає 60% світового вебтрафіку [1]. Таким чином, фактично, на даному етапі мобільні пристрої є основним різновидом пристроїв для використання глобальної мережі Інтернет, а позаяк переважно використання складних розумних пристроїв напряму пов'язане з мережею, то виходить, що мобільні пристрої є одним з основних різновидів видом таких пристроїв для людства, задоволення його потреб.

Оскільки практично складно забезпечити стовідсоткову безпеку від втручання зловмисників, завжди з’являтимуться нові й нові засоби, які загрожуватимуть безпеці мобільних пристроїв. Проте такі ризики можна мінімізувати, якщо знати на що потрібно звертати увагу та до яких заходів забезпечення безпеки вдаватися.

Отже, моделювання вебзастосунку формування моделі загроз безпеці мобільних пристроїв для його програмного реалізування є актуальним завданням.

Об’єкт дослідження: процес формування моделі загроз безпеці мобільних пристроїв.

Предмет дослідження: програмні застосунки формування моделі загроз безпеці мобільних пристроїв.

1. СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБЗАСТОСУНКУ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

1.1. Аналізування процесу формування моделі загроз безпеці мобільних пристроїв

Визначимо, що таке мобільні пристрої. В контексті даної роботи, мобільний пристрій це малий обчислювальний пристрій, достатньо малий щоб тримати та використовувати в одній чи двох руках, який має операційну систему, здатну запускати мобільні застосунки [2]. До таких пристроїв можна віднести, в першу чергу, смартфони, а також планшети, розумні годинники.

Кожен різновид пристроїв має свої особливості, на яких засновується та до яких прив’язується реалізація програмного забезпечення, яке пристроєм використовується. З огляду на це, кожен різновид пристроїв слід розглядати окремо, коли йдеться про забезпечення безпеки. Проте, попри наявність різних операційних систем для одного виду пристроїв (наприклад, для мобільних пристроїв це iOS, Android та HarmonyOS), є певні об’єднувальні властивості, через що досить багато вразливостей та небезпек часто збігається для всіх пристроїв класу (в даному випадку мобільних). Попри зовнішню різницю, смартфони, планшети, розумні годинники мають схожу структуру, особливості, та поділяються між собою щодо операційних систем. Тому для них можуть бути характерні однакові вразливості та заходи безпеки. Відтак у даній роботі розглядаються мобільні пристрої загалом.

При здійсненні нападу зловмисники можуть використовувати різні види та методи власне виконання дій та досягнення поставлених цілей, таких як виконання злоякісного коду, збереження даних, підвищення прав, обхід захисту, доступ до облікових даних. На кожну мету існує щонайменше декілька видів технік, за допомогою яких зловмисник може досягти її. Тож так само як і з цілями, при формуванні моделі необхідно враховувати якомога більше варіантів шляхів досягнення цілей зловмисниками, щоб за допомогою моделі можна було якомога повніше визначити всі частини

програмного забезпечення, які потребують доопрацювання для запобігання або мінімізування ризику реалізування загроз з боку зловмисників.

Також, для дійсно повноцінних розуміння та можливості використання такої моделі, вона має включати в себе інформацію про можливості запобігання та/або мінімізації загрози. Не завжди можливості для обходу безпеки дійсно можна усунути або мінімізувати в межах самого програмного забезпечення, проте можна сформувати інформацію, яка допоможе зрозуміти, як самостійно з додатковими заходами безпеки з боку людини загрозу можна мінімізувати.

До того ж, слід зауважити, що різні люди володіють абсолютно різним рівнем навичок та розуміння сучасних технологій, а отже, для того, аби якомога більше людей могли скористатися моделлю загроз потрібному пристрою для забезпечення його безпеки, необхідно щоб модель могла містити корисну інформацію для якомога ширшої авдиторії. Тобто, наприклад, треба щоб в моделі містилася інформація, яка була б корисною звичайному пересічному користувачеві, і з її допомогою пересічний користувач міг би вжити заходів з безпеки які мали б вплив. Проте, також не слід забувати, що деякі вразливості пересічний користувач у межах своїх знань та розуміння усунути не зможе. Скажімо, розуміння загроз використання певних функцій у коді буде корисним лише для інженера програмного забезпечення, який розробляє певний застосунок. Утім, така інформація також є корисною для певної категорії осіб (наприклад, зазначені вище програмні інженери), хоч і обмеженої. Таким чином, для максимального охоплення можливих бенефіціарів потрібно враховувати всі категорії тих, хто потенційно зможе мати користь з формування такої моделі загроз.

Модель має максимально спрощувати процес забезпечення безпеки, тому вона має бути дійсно зрозумілою, простою та конкретною у використанні для кожного, хто може нею скористатися. Якщо формувати

загальну модель, то лишається необхідність пошуку додаткової інформації, що не є ідеальним результатом використання такої моделі. Отже, кожна конкретна модель має співвідноситися з конкретним пристроєм, який цікавить певну особу, яка потребує забезпечення безпеки такого пристрою.

Також, необхідно розуміти, що жодна подібна модель не допоможе забезпечити стовідсоткову безпеку жодному пристроєві, позаяк час від часу знаходяться невідомі раніше вразливості, або ж навіть з'являються нові після певних оновлень програмного забезпечення або виходу його нових версій чи отримання патчів. Неможливо мати повний перелік всіх конкретних цілей та шляхів їх досягнення. Тому неможливо сформувати одну модель назавжди, необхідно стежити за актуальними новинами галузі, та адаптувати модель під усі нові зміни, які відбуваються з часом, аби модель була якомога повнішою. І, хоч така модель все ще не покриє всі можливі загрози мобільним пристроям, що більше факторів при її формуванні враховано, то більш мінімізованими є ризики.

Отже, якщо заходів безпеки, які стосуються того чи іншого мобільного пристрою, який розглядається при формуванні моделі загроз безпеці, досить багато, корисним буде відстежувати, які з наявних у сформованій моделі заходів забезпечення безпеки вже виконано, а які ще належить виконати, а також розуміти рівень досягнутого в ході вжиття заходів прогресу.

1.2. Аналізування програмних застосунків формування моделі загроз безпеці мобільних пристроїв

З кожним днем більшість людей стають свідомими щодо того, що сучасні мобільні пристрої мають різні вразливості програмного забезпечення, які ставлять їх під загрозу нападів з боку зловмисників, і зі зростанням складності таких вразливостей, загроз тільки більшає. Також достатньо людей розуміє, що існує достатньо причин для зловмисників цікавитися зламом мобільних пристроїв для тієї чи іншої мети у зв'язку з тим на що здатні нині такі пристрої. Через це вже існують застосунки, які

працюють з моделлю загроз мобільним пристроям у різні способи. Утім, наразі відсутні такі застосунки, які б автоматично формували модель загроз для обраного пристрою. У зв'язку з цим у даному підрозділі буде розглянуто три застосунки, які так чи інакше пов'язані з формуванням моделі загроз, та розглянуто переваги й недоліки кожного з них. Зокрема буде розглянуто: Microsoft Threat Modelling Tool, OWASP Mobile Application Security та ThreatModeler.

Для початку розглянемо Microsoft Threat Modelling Tool. Його випущено у вересні 2018 як безкоштовний інструмент, який призначений для того, щоб надавати користувачам можливість розроблювати моделі загроз, які далі використовуватимуться у процесі забезпечення безпеки. Даний програмний застосунок не спеціалізується конкретно на мобільних пристроях, проте може використовуватися і для них.

Загалом застосунок включає в себе такі етапи, як створення діаграми, визначення загроз, визначення заходів безпеки, що застосовуються до визначених загроз, а також перевіряння визначених заходів безпеки.

Спочатку створюється діаграма, яка відповідає концептуальній моделі процесів, пов'язаних з об'єктом моделі загроз, що формується. Наприклад, це може бути DFD-діаграма. Застосунок дозволяє при побудові діаграми визначати обмеження для користувачів різних ролей та різного рівня доступу.

Далі, на основі створеної діаграми можна згенерувати загрози, які можуть стосуватися об'єкту моделі загроз. Надається інформація стосовно потенційних векторів атаки, а в додатковому описі наводиться інформація конкретнішого характеру, як зазначає, що саме не так, та як це можна покращити або виправити. Можна визначити та змінювати пріоритетність та лишити нотатки. Також згенеровані загрози можна відмітити як «Незастосовні», «Не початі», «Потребують дослідження» та «Вирішені».

Після цього можна згенерувати звіт, у якому буде вказано загрози, проставлені відмітки, наведено встановлені пріоритети та додано введені нотатки. Далі цей звіт можна поширювати для аналізу іншими людьми [3].

Загалом, даний застосунок надає можливості автоматичного генерування моделі загроз та відстеження прогресу у їх вирішенні та вжитті заходів безпеки. Втім, він не спеціалізується на мобільних пристроях, і не заявлено жодного пристосування під них, тому ймовірно багато аспектів не є можливим врахувати у випадку використання цього застосунку. Окрім того, для генерування моделі слід спершу побудувати діаграму, що потребує додаткових знань та розуміння.

Тепер розглянемо OWASP Mobile Application Security. Цей застосунок визначає свою мету як «Визначити стандарти індустрії щодо безпеки мобільних застосунків». Загалом OWASP надає стандарт безпеки для них, та, окрім того, гайд із тестування, який охоплює засоби, способи та процеси тестування мобільних застосунків, а також вичерпний набір тест-кейсів, який має допомогти тестувальникам проводити послідовне та всеохопне тестування безпеки таких застосунків [4].

Окрім головної сторінки та кількох сторінок з додатковою інформацією, OWASP містить чотири основні розділи, а саме «MASTG», «MASVS», «MAS Checklist» та «MAS Crackmes».

«MASTG» – це розділ, який містить гайд із тестування безпеки мобільних застосунків. Вміст розділу можна переглядати в самому застосунку, а також можна завантажити pdf-файл. Тут надаються загальний гайд, особливості для Android, особливості для iOS та інструменти для тестування.

Розділ «MASVS» містить у собі інформацію про авторський стандарт безпеки. Тут за різними категоріями розписані концептуальні вимоги до безпеки мобільного застосунку, яких слід дотримуватися при розробці.

Розділ «MAS Checklist» містить шаблон чек-листа, який може стати у нагоді тестувальникам при перевірці безпеки мобільного застосунку. Чек-лист містить посилання на конкретні тесткейси.

Розділ «MAS Crackmes» містить головоломки для розробників/ тестувальників, які тренують навички стосовно забезпечення безпеки мобільних застосунків і покращують знання та уміння для ліпшого розуміння. Містить окремо головоломки для Android та для iOS.

Загалом, OWASP, хоч і має певні додаткові корисні можливості, але все одно є по суті просто базою знань. Окрім того, OWASP зосереджений конкретно на безпеці мобільних застосунків, а не мобільних пристроїв як таких.

Наостанок, розглянемо ThreatModeler. Даний застосунок дещо суттєво відрізняється від попередніх двох тим, що є програмною системою. Натомість він працює на багатьох пристроях водночас, які пов’язані з однією системою. Застосовний для великих компаній, які хочуть забезпечувати безпеку мобільних пристроїв співробітників. Він дозволяє моніторити стан пристроїв співробітників та виявляти загрози. Він визначає, як зловмисник обходить систему, визначаючи потенційні цілі для атаки, і які засоби можуть бути для цього використані. Зазначається, що для користування не є необхідними широкі знання в галузі кібербезпеки та тривале навчання.

Даний застосунок збирає інформацію з архітектурних компонентів системи пристрою та автоматично визначає всі застосовні загрози, які стосуються кожного компоненту. Разом з тим, збираються відповідні вимоги безпеки, сценарії тестування, керівництво розгляду загроз та інша корисна інформація., яку можна застосувати для усунення чи мінімізації ризиків та загроз безпеці мобільних пристроїв.

Застосунок постійно відстежує потенційні загрози та сповіщає про них. Він автоматично оновлює дані в реальному часі. Для підвищення швидкодії використовуються зокрема шаблони, які відповідають компонентам та застосункам, що часто використовуються. Це дозволяє значно швидше генерувати нові моделі загроз [5].

Таким чином, ThreatModeler зосереджується на безпеці самих пристроїв, проте він має дещо іншу спрямованість, аніж вебзастосунок, що розроблятиметься. По-перше, це по суті сервіс для моніторингу стану пристроїв, а не допоміжний інструмент для розуміння необхідних заходів безпеки, по-друге, він спрямований конкретно на організації. Окрім того, оскільки він спрямований на організації, то він оплачуваний. Також відсутня можливість просто придбати застосунок, можна лише подати заявку на демонстрацію від фахівця [6], за результатами якої, вочевидь, укладатиметься угода.

Порівняємо розглянуті в цьому підрозділі програмні застосунки шляхом складання порівняльної таблиці (табл. 1.1).

Таблиця 1.1

Порівняння особливостей підходів розглянутих застосунків

Застосунок	Microsoft Threat Modelling Tool	OWASP Mobile Application Security	ThreatModeler
1	2	3	4
Пристосованість та спеціалізованість на мобільних пристроях	–	+	+

Продовження табл. 1.1

1	2	3	4
Не потребує створення діаграм	–	+	–
Автоматично генерує загрози	+	–	+
Автоматично пропонує пом’якшення (заходи безпеки)	+	–	+
Надає можливість проставляти відмітки для відслідковування прогресу забезпечення безпеки	+	+	–
Є відкритим застосунком для будь-яких користувачів	+	+	–
Надає можливість формування моделі загроз для конкретного пристрою	–	–	+

1.3. Синтезування застосунку формування моделі загроз безпеці мобільних пристроїв

З огляду на результати аналізування процесу формування моделі загроз безпеці мобільних пристроїв, а також відомих програмних застосунків, існує потенційна потреба в розробленні застосунку, який би був орієнтований на реалізування завдань у межах процесу формування моделі загроз. Позаяк він займе певну свою нішу серед застосунків, спрямованих на забезпечення безпеки мобільних пристроїв, лише чи, зокрема, і матиме свою користь для тих, хто потребує деякого спрощення процесу забезпечення безпеки і насамперед в особистому використанні, проте не повноцінного моніторингу стану.

Як було встановлено за результатами аналізу процесу формування моделі загроз безпеці мобільних пристрої, при її побудові доцільно враховувати якомога більше методів, якими для досягнення своїх цілей можуть скористатися зловмисники. Це необхідно для того, аби надати користувачеві як найповнішу картину потенційних загроз та вразливостей, аби він міг повною мірою в межах своїх можливостей забезпечити безпеку та мінімізацію ризиків своєму пристрою чи пристроям.

Також, для забезпечення такої можливості для користувача в повній мірі він має розуміти як саме можна мінімізувати ризики та запобігти загрозам. Для цього мають надаватися варіанти дій щодо протидії тим чи іншим загрозам та вразливостям.

До того ж, користувачеві має бути зручно відстежувати, які потенційні загрози та вразливості вже пройшли його оцінку, а що ще необхідно оцінити. Зручний спосіб здійснення такої діяльності спростив би процес встановлення небезпек для користувача. Саме тому було б корисним, аби користувач міг автоматично додавати загрози зі сформованої моделі у певний чек-лист, який надалі міг би використовуватися для відстеження процесу оцінювання небезпек для його мобільного пристрою.

Також слід враховувати, що певний прошарок суспільства менш обізнаний у технологіях, тому модель варто формувати з точки зору користувача в першу чергу не за особливостями пристрою, а за конкретним пристроєм. Тому, що деякі користувачі можуть просто не знати або мати помилкові уявлення про деякі особливості власних пристроїв. Наприклад, багато людей можуть помилятися стосовно версії операційної системи, на якій працює їхній мобільний пристрій.

Окрім того, доцільно зауважити, що, оскільки, як було зазначено в аналізі, актуальна інформація постійно змінюється і потребує постійного відстежування та актуалізації даних задля відповідності застосунку

реальному стану в галузі, то має бути певна людина чи група людей, яка би за потреби перевіряла та оновлювала інформацію. При цьому задля запобігання дезінформації чи некоректному підходу до внесення інформації це має бути чітко визначена людина чи група людей. І, відповідно, застосунок формування моделі загроз має передбачати можливість редагування даних у системі, аби не було потреби здійснювати таку діяльність через базу даних або інші засоби нижчого рівня.

З огляду на можливості редагування інформації певною групою людей та створення особистих чек-листів, має бути передбачена система облікових записів, щоб кожен користувач чи адміністратор мав змогу увійти під власним ідентифікатором. При цьому особиста реєстрація потрібна лише звичайному користувачеві, позаяк редагуванням інформації має займатися окрема визначена група людей, отже акаунти таких людей мають створюватися адміністратором ресурсу. Адміністраторів має бути один чи декілька, але не більше, тому обліковий запис адміністратора має створюватися іншими засобами, ніж через основний інтерфейс застосунку.

Як тип такого застосунку обирається вебзастосунок, аби бути доступним із якомога більшої кількості різновидів пристроїв і меншої фрагментації, яка може призводити до невідповідностей. Доступність із різних пристроїв потрібна для того, щоб людина, яка займатиметься забезпеченням безпеки мобільного пристрою, найімовірніше переглядатиме інформацію не з конкретного пристрою, а з найбільш зручного для певного користувача. І, водночас, може відрізнятися від людини до людини. За аналогією, інформація може вноситися так само з різних пристроїв. Вебзастосунок повинен забезпечувати максимальну доступність для різних браузерів. Це більш просто, ніж враховувати багато операційних систем різноманітних типів та спрямувань.

Отже, в підсумку, на основі аналізування процесу формування моделі загроз безпеці мобільних пристроїв, відомих програмних застосунків, а також наведених у даному підрозділі припущень, було сформовано такі функціональні вимоги до вебзастосунку формування моделі загроз безпеці мобільних пристроїв:

- авторизація в системі;
- вихід із системи;
- відновлення паролю;
- реєстрація в системі;
- створення акаунту модератора адміністратором;
- переглядання та пошук по переліку модераторів адміністратором;
- керування модераторами адміністратором;
- переглядання інформації профілю;
- змінення даних профілю;
- змінення паролю;
- додавання пристрою модератором;
- переглядання та пошук пристроїв модератором;
- редагування пристроїв модератором;
- видалення пристроїв модератором;
- додавання загроз модератором;
- переглядання та пошук всіх загроз модератором;
- редагування загроз модератором;
- видалення загроз модератором;
- додавання пом’якшень модератором;
- переглядання та пошук всіх пом’якшень модератором;
- редагування пом’якшень модератором;
- видалення пом’якшень модератором;

- отримання переліку потенційних загроз для конкретного пристрою (формування моделі загроз);
- отримання переліку пом’якшень для конкретної загрози;
- створення чек-листа на основі отриманих загроз користувачем;
- проставлення відміток у чек-листі користувачем;
- переглядання та пошук власних чек-листів користувачем;
- редагування назв чек-листів користувачем;
- видалення чек-листа користувачем;
- виведення рівня прогресу в ході вжиття заходів безпеки.

Також, для коректного функціонування вебзастосунку у зручний для користувачів спосіб, забезпечення безпеки застосунку, а також для забезпечення виконання вимог [7] було визначено такі нефункціональні вимоги:

- сумісність застосунку з найпопулярнішими браузерами;
- адаптованість як до великих так і до менших екранів;
- наявність української мови;
- використання вбудованих засобів розробки для запобігання SQL-ін’єкціям;
- використання односторонньої функції для зберігання паролів у базі даних для запобігання зберігання їх у відкритому вигляді;
- перехоплення та оброблення помилок на стороні клієнта та виведення користувачеві лише відформатованих помилок для загального розуміння;
- перехоплення та оброблення помилок на стороні сервера та надсилання клієнту лише обмеженої інформації про помилку;
- забезпечення різних прав доступу та перевіряння прав при спробі доступу до ресурсу в застосунку;

- використання змінних середовища для запобігання витоку чутливих даних;
- використання CORS-політики для обмеження доступу до сервера з інших ресурсів.

Для наочної демонстрації розроблених вимог, наведемо створену на їхній основі діаграму в нотації SysML (рис. 1.1). Для уникнення громіздкості діаграми згрупуємо деякі функціональні вимоги в узагальнені:

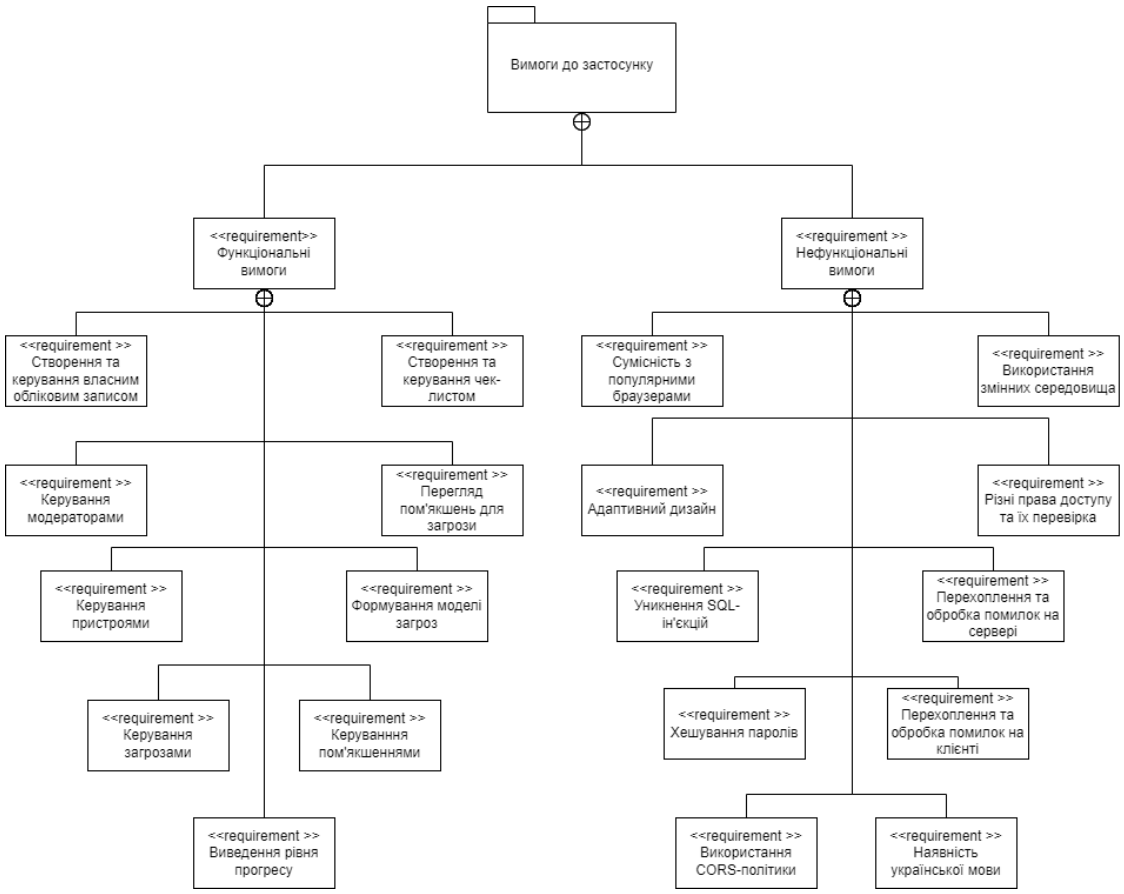


Рис. 1.1. Діаграма вимог до вебзастосунку формування моделі загроз безпеці мобільних пристроїв у нотації SysML

1.4. Висновки до розділу 1

Проаналізовано процес формування моделі загроз безпеці мобільних пристроїв. За його результатами визначено на що слід орієнтуватися та що враховувати при формуванні моделі. Так, встановлено, що суттєвий ріст

популярності та розповсюдженості мобільних пристроїв, а також зростання їхньої складності та можливостей, призвело до зацікавлення ними з боку зловмисників. Крім того при формуванні моделі загроз доцільно враховувати якомога більше цілей та шляхів їх досягнення задля забезпечення її повноти. Це доповнюється необхідністю розуміння інформації про пом’якшення загроз та мінімізування ризиків.

Проаналізовано відомі програмні застосунки, а саме Microsoft Threat Modelling Tool, OWASP Mobile Application Security та ThreatModeler. Визначено, що дані рішення мають своє застосування. Проте Microsoft Threat Modelling Tool не спеціалізується конкретно на мобільних пристроях. OWASP Mobile Application Security є більше базою знань, аніж засобом формування моделі загроз. І, ThreatModeler є моніторинговим рішенням для ентерпрайзу, яке хоч і надає модель загроз, проте актуальне лише для компаній.

Отже, за результатами аналізування процесу формування моделі загроз, відомих програмних застосунків сформовано функціональні та нефункціональні вимоги до вебзастосунку. Зокрема встановлено, що він буде сумісним із популярними браузерами та мати адаптивний дизайн, а також використовуватиме засоби безпеки. Вебзастосунок має надавати користувачеві можливість формувати модель загроз безпеці мобільних пристроїв, надавати інформацію про пом’якшення реалізувань загроз. Вести чек-листи, а також керувати наявною в системі інформацією для визначеної групи людей з метою підтримання її в актуальному стані.

2. КОНЦЕПТУАЛЬНА МОДЕЛЬ ВЕБЗАСТОСУНКУ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

2.1. Формалізація діяльності формування моделі загроз безпеці мобільних пристроїв

З огляду на отримані результати в попередньому розділу, по-перше, проаналізовано процес формування моделі загроз безпеці мобільних пристроїв, по-друге, проаналізовано програмні застосунку для розв'язання даного завдання, і, по-третє, синтезовано вебзастосунок формування моделі загроз безпеці мобільних пристроїв. Відтак тепер потрібно формалізувати зазначену діяльність. Це потрібно для того щоб коректно вибудувати концептуальну модель вебзастосунку на рівні функцій. Оскільки необхідно їх розуміти, вхідні та вихідні дані, обмеження та ресурси щодо їхнього реалізування.

Здійснимо процес формалізації шляхом побудови діаграми. Як нотацію оберемо IDEF0. IDEF0 – це методологія графічного моделювання процесів, яка широко застосовується з метою функціонального представлення даних, аналізування процесів в організації, застосунку. Це одна з 16 нотацій сімейства IDEF, і є однією з найбільш уживаних серед них [8]. Вона є досить гнучкою і тому може використовуватися в різних умовах та за різних обставин. При цьому надавати досить строгий стандарт формату, що дозволяє мати однозначну інтерпретацію діаграм.

Тож перейдемо до побудови діаграми. Перш за все доцільно створити контекстну діаграму, тобто діаграму найвищого рівня, яка узагальнює весь процес та деталізує лише зовніші чинники. Це потрібно для того щоб зрозуміти загальні деталі зазначеної діяльності, перш ніж розбивати її на частини та деталізувати представлення кожної з них.

Така діаграма містить лише один елемент позначення діяльності. В даному випадку це елемент під назвою «Формування моделі загроз безпеці мобільних пристроїв». Окрім того вона має містити вхідні та вихідні

елементи (стрілки), які будуть спільними для діаграм IDEF0 усіх рівнів, а саме: елементи входу, елементи виходу, елементи керування та елементи ресурсів.

Почнемо з елементів входу. Перш за все на вхід надходять дані користувача. Вони необхідні для того, аби, для початку, зареєструватися в системі, щоби могти користуватися всіма її функціональними можливостями. Пізніше, при повторному використанні системи, для того, щоб увійти в систему та мати доступ до функціональних можливостей та власних даних, а саме даних облікового запису/профілю та даних чек-листів.

Далі, як зазначалося, для початку роботи з формування моделі загроз безпеці мобільних пристроїв користувач має обрати модель пристрою, для якої він хоче сформувати модель загроз. Тож виникає потреба – отримати параметри пристрою, які можуть впливати на його безпеку та загрози їй.

Отже, елементами входу є:

- дані користувача;
- дані конкретного пристрою для формування моделі загроз.

Тепер розглянемо елементи виходу. В першу чергу на вихід, тобто користувачеві, подається власне сама сформована модель загроз безпеці мобільного пристрою, а саме формалізована та приведена до певного вигляду інформація про техніки, до яких можуть вдаватися зловмисники при нападах, а також заходи безпеки, до яких рекомендується вдатися користувачеві аби захистити свій пристрій та мінімізувати ризики. Також на основі сформованої моделі створюється чек-лист, за допомогою якого користувач може відслідковувати, які заходи безпеки вже вжив та яким є його прогрес.

Отже, маємо такі елементи виходу:

- формалізована інформація про загрози для пристрою;
- формалізована інформація про пом’якшення для пристрою;

- чек-лист.

Відтак розглянемо елементи керування. Досить часто на діаграмах цієї нотації до елементів керування відносять законодавчу та нормативну базу, керуючись якою відбувається процес, який розглядається на діаграмі. З точки зору вебзастосунку певні закони та норми, які слід враховувати, дійсно існують, проте в межах даного розділу розглядається виключно сам процес формування самої моделі, та ті аспекти, які важливі для цього процесу, а не вебзастосунку загалом.

До того, що напряму впливає на процес формування моделі загроз безпеці мобільних пристроїв можна віднести:

- модель пристрою;
- коректність даних користувача;
- становище у сфері кібербезпеки.

Модель пристрою є важливою позаяк від неї напряму випливають особливості, на які слід звертати увагу для розуміння того, які загрози можуть вплинути на безпеку конкретного пристрою. Коректність даних користувача необхідна, аби правильно його автентифікувати та авторизувати, щоби він мав змогу отримати інформацію про загрози та їх пом’якшення. Під становищем у сфері кібербезпеки мається на увазі інформація стосовно існуючих на сьогоднішній день загроз, злочинних угруповань, шкідливого програмного забезпечення, яка необхідна для формування актуальної моделі, що може бути корисною користувачеві станом саме на даний момент.

Наостанок розглянемо ресурси. Ними відображається все, що використовується для забезпечення функціонування зазначеного процесу. Це може бути технічне забезпечення, програмне забезпечення, люди від яких залежить процес. Отже, до ресурсів віднесемо:

- адміністраторів;
- модераторів;

- користувача;
- хостинг.

Адміністратори та модератори забезпечують актуальність інформації в системі, яка використовується процесом, хостинг необхідний для можливості роботи процесу, а користувач забезпечує функціонування самого процесу, тому це все є ресурсами процесу.

Тепер, на основі наведеної інформації щодо побудови контекстної діаграми побудуємо її. Отримаємо такий результат (рис. 2.1).

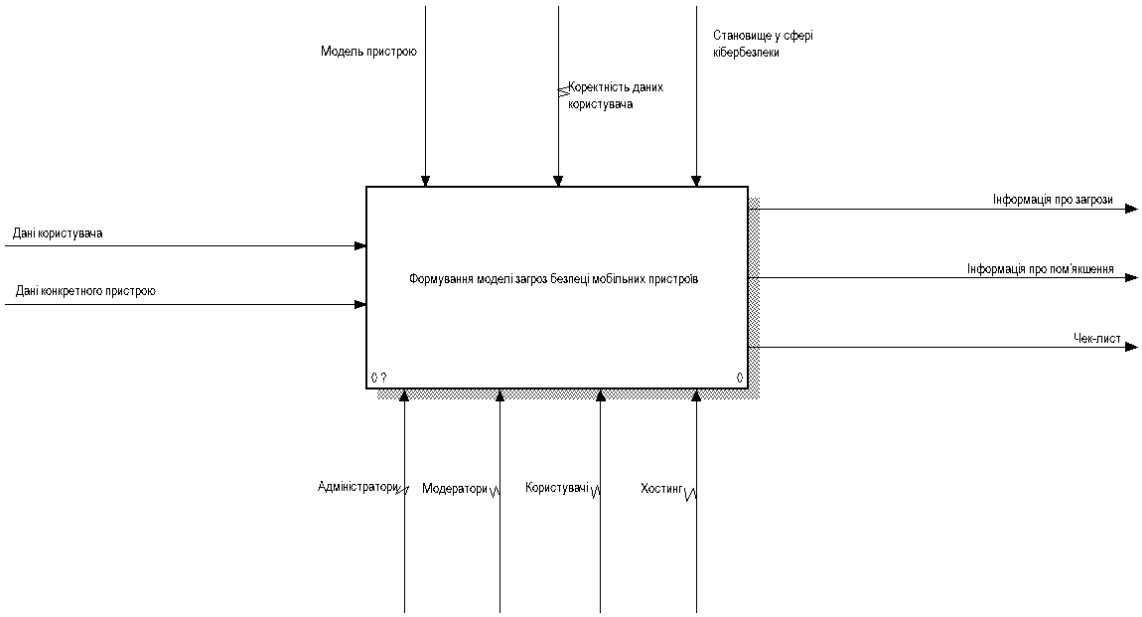


Рис. 2.1. Контекстна діаграма діяльності формування моделі загроз безпеці мобільного пристрою в нотації IDEF0

Для детальнішого розуміння процесу формування такої моделі у межах коректної його реалізації доцільно здійснити декомпозицію даної діаграми та розбити діяльність на функції нижніх рівнів.

Першою функцією є реєстрація користувача (рис. 2.2). Вона потрібна для того, аби в системі з’явилися дані користувача як такі, і в межах діяльності можна було б отримати до них доступ. І, як наслідок,

скористатися функцією формування моделі загроз безпеці мобільних пристроїв.

Єдиним елементом входу такої функції будуть власне дані користувача. Як ресурси зазначаємо хостинг, оскільки система вже має працювати щоб внести в неї дані. Крім того самого користувача, оскільки саме він виконує дії щодо внесення його даних у систему. Елементом керування буде коректність даних користувача, оскільки користувач зможе зареєструватися лише за умови, наприклад, відповідності паролю встановленим до нього вимогам. На виході отримуємо обліковий запис, створений у системі.

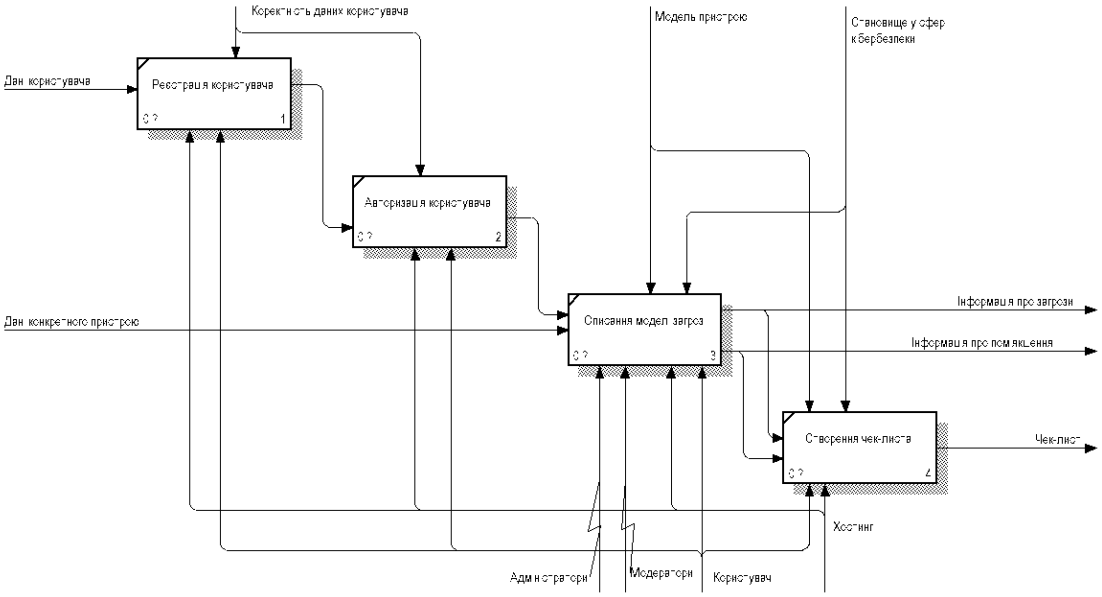


Рис. 2.2. Декомпозиція контекстної діаграми діяльності формування моделі загроз безпеці мобільному пристрою в нотації IDEF0

Наступною функцією є авторизація користувача (рис. 2.2). Ця фунція потрібна для того, щоб користувач власне отримав доступ до формування моделі загроз пристрою та чек-листів. Для верифікації того, що користувач є дійсним і має мати доступ до вищезазначених функціональних можливостей передбачається підтвердження його особи.

Елементи керування та ресурси тут є ідентичними до тих, які використовуються для реєстрації, з огляду на те, що з точки зору технічної реалізації процеси досить схожі. Однак, як вхід розглядаються дані облікового запису, які з’явилися в системі після реєстрації. На виході маємо підтвердження особи користувача.

Третьою функцією є власне описання моделі загроз на основі вхідних даних та інформації в системі (рис. 2.2). Цей підпроцес є по суті основним оскільки відповідає за саме формування моделі.

Першим елементом входу є підтвердження дійсності користувача. Це необхідно аби впевнитись, що користувач дійсно має мати доступ до цих функціональних можливостей. Іншими елементами входу є дані конкретного пристрою для формування моделі, які необхідні для аналізу того, які саме загрози можуть бути застосовані в конкретному випадку. Як ресурси виступають: хостинг (потрібний, щоб модель могла власне автоматично сформуватися і користувач мав до неї доступ), модератори (потрібні для підтримки інформації в системі, що використовується для формування моделі, в актуальному стані), адміністратори (потрібні для керування модераторами), користувач (потрібен для власне реалізування функції). Елементами керування є модель пристрою та становище в сфері кібербезпеки. Ця інформація, як зазначалося, необхідна для формування моделі загроз безпеці конкретного пристрою, яка водночас буде актуальною для часу її формування. Відповідно на виході маємо інформацію, отриману за результатами побудови моделі.

Останньою функцією є створення чек-листа на основі сформованої моделі загроз безпеці мобільного пристрою. Цією функцією реалізується збереження даних про загрози для конкретного пристрою користувача, а також надання можливості ставити відмітки для відслідковування користувачем власних дій щодо забезпечення його безпеки свого.

На вході маємо все, що було виходом попередньої функції. Ресурсами є хостинг та користувач, оскільки для роботи системи і зв'язку з користувачем потрібен хостинг, а функція реалізується користувачем. Елементи керування ті самі, що й для функції описання моделі загроз, оскільки чек-лист формується чітко на основі сформованої моделі. На виході отримаємо сформований чек-лист.

2.2. Визначення зв'язків між етапами формування моделі загроз безпеці мобільних пристроїв

Після виконання формалізації діяльності формування моделі загроз безпеці мобільних пристроїв доцільно визначити, які саме зв'язки постають між етапами даного процесу, аби при розробленні вебзастосунку коректним чином пов'язати його функціональні частини між собою. Для цього також побудуємо діаграму в графічній нотації IDEF3.

IDEF3, як зрозуміло з назви, також належить до сімейства нотацій діаграм IDEF, проте на відміну від IDEF0 зосереджується на зв'язках між підпроцесами, які являють собою етапи процесу. Вона також відображає принципи прийняття рішень. Це означає, що діаграма містить розгалуження, які передбачають варіанти дій або одночасні дії [9]. Така діаграма складається з блоків процесів, об'єктів звертання, розгалужень та стрілок зв'язків.

Оскільки дана нотація передбачає прийняття рішень, і, відповідно, розгалуження, то на даній діаграмі можемо доповнити перелік підпроцесів.

На початку процесу отримуємо на вхід дані користувача, що позначено відповідною дугою з такою назвою зліва на діаграмі (рис. 2.3). Відразу маємо розгалуження типу XOR – тобто два взаємовиключні варіанти подальшого ходу дій. На даному етапі можна або обрати реєстрацію, або відразу перейти до авторизації. Таким чином, одна однойменна дуга відходить до підпроцесу «Реєстрація користувача», а інша

однойменна дуга іде прямо до з’єднання. Окрім того, від процесу «Реєстрація користувача» маємо дугу «Дані облікового запису», яка йде до з’єднання.

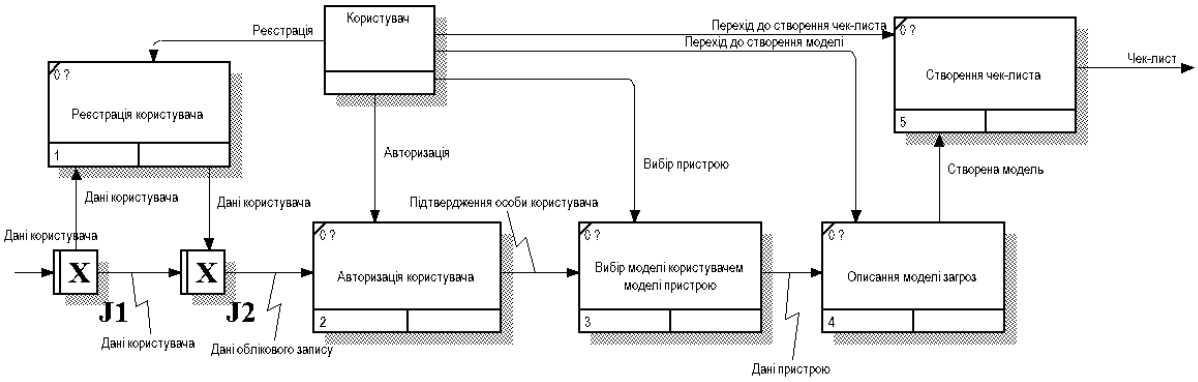


Рис. 2.3. Декомпозиція контекстної діаграми діяльності формування моделі загроз безпеці мобільних пристроїв у нотації IDEF3

Від з’єднання до наступного підпроцесу йде дуга «Дані облікового запису». Цей підпроцес – «Авторизація користувача». Від нього маємо дугу до наступного процесу – «Вибір користувачем моделі пристрою зі списку», оскільки саме вибір моделі пристрою для формування моделі загроз є першим кроком.

Від даного підпроцесу до з’єднання веде дуга «Дані пристрою», із самого з’єднання потім виходить однойменна дуга, що веде до наступного підпроцесу, а саме «Описання моделі загроз на основі вхідних даних та інформації в системі». Це є основний підпроцес який власне формує саму модель. Одразу після обрання моделі пристрою обирається формування моделі, тому цей підпроцес іде відразу за попереднім. З нього маємо дугу «Створена модель», яка веде до процесу «Створення чек-листа». Тут також немає розгалуження, оскільки за створенням моделі відразу слідує створення чек-листа, позаяк саме чек-лист по суті є способом збереження

сформованої моделі, який додатково надає можливості проставляння відміток. З цього підпроцесу виходить дуга «Чек-лист», яка вже йде на вихід. Оскільки на цьому процес формування закінчується, і користувач після сформованої моделі отримує сформований чек-лист, в якому він може проставляти відмітки і яким може керувати.

Всі вищенаведені підпроцеси з'єднані з єдиним об'єктом звертання – користувачем. Це пов'язано з тим, що коли відбувається виконання процесів, описаних на діаграмі, єдиним їх учасником є фактично сам користувач, який вводить дані та ініціює процеси. Таким чином, додаємо об'єкт «Користувач» та стрілки від нього: «Реєстрація» до підпроцесу реєстрації, «Авторизація» до підпроцесу авторизації, «Вибір пристрою» до підпроцесу вибору пристрою, «Перехід до створення моделі» до підпроцесу описання моделі загроз, та «Перехід до створення чек-листа» до підпроцесу створення чек-листа.

2.3. Визначення потоків даних між етапами формування моделі загроз безпеці мобільних пристроїв

У попередніх підрозділах формалізовано діяльність а також встановлено зв'язки між етапами процесу формування моделі загроз безпеці мобільних пристроїв. Наступним кроком є визначення потоків даних між цими етапами. Для цього побудуємо діаграму в графічній нотації DFD.

Діаграми в графічній нотації DFD відображають потоки даних між частинами процесів та допомагають концептуалізувати, де які дані зберігаються та звідки й куди передаються. Такі діаграми мають стрілки, які позначають власне потоки даних та три види блоків – сховища даних, процеси та зовнішні сутності, тобто, інакше кажучи, джерела даних [10].

Спершу побудуємо контекстну діаграму, яка відображатиме потоки даних для узагальненого процесу – формування моделі загроз безпеці

мобільних пристроїв. Саме цей процес і виступатиме як основний функціональний блоку на діаграмі (рис. 2.4).

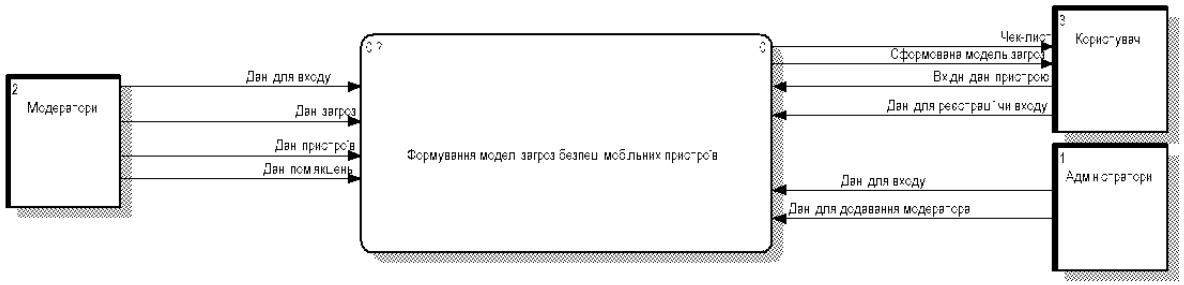


Рис. 2.4. Діаграма потоків даних формування моделі загроз безпеці мобільних пристроїв у нотації DFD

Як сутності визначено адміністраторів, модераторів та користувачів. Адміністратори є джерелом облікових даних модераторів (оскільки саме адміністратори реєструють модераторів), відповідно від них маємо потік даних «Дані для додавання модераторів», який призначений для реєстрації модераторів у системі. Також від них отримаємо потік даних «Дані для входу», позаяк адміністратори також можуть авторизуватися. Модератори є джерелом даних пристроїв, загроз та способів пом’якшень, які вносяться в систему задля підтримання інформації, на основі якої формується модель загроз, в актуальному стані. Відповідні потоки даних ведуть до основного функціонального блоку. Також модератори як і адміністратори є джерелом даних для входу, оскільки теж повинні авторизуватися в системі, тому маємо відповідний потік. Користувач є джерелом вхідних даних конкретного пристрою для формування моделі, позаяк для цього він обирає конкретну модель пристрою, для якої формуватиметься модель. Тож маємо потік даних «Вхідні дані пристрою». Також від користувача отримуємо потік «Дані для реєстрації чи входу». Оскільки користувач крім авторизації може також, власне, зареєструватися, на відміну від адміністраторів та

модераторів. З іншого боку, до користувача також маємо два потоки даних – «Сформована модель загроз» та «Чек-лист», оскільки системою після формування моделі загроз відображається отриманий результат користувачеві та створюється на її основі чек-лист, у якому користувач зможе ставити відмітки.

Для детальнішого розуміння потоків даних отриману контекстну діаграму (рис. 2.4) доцільно декомпозиціювати (рис. 2.5). Як функціональні блоки виступатимуть «Реєстрація користувача», «Авторизація користувача», «Описання моделі загроз» та «Створення чек-листів».

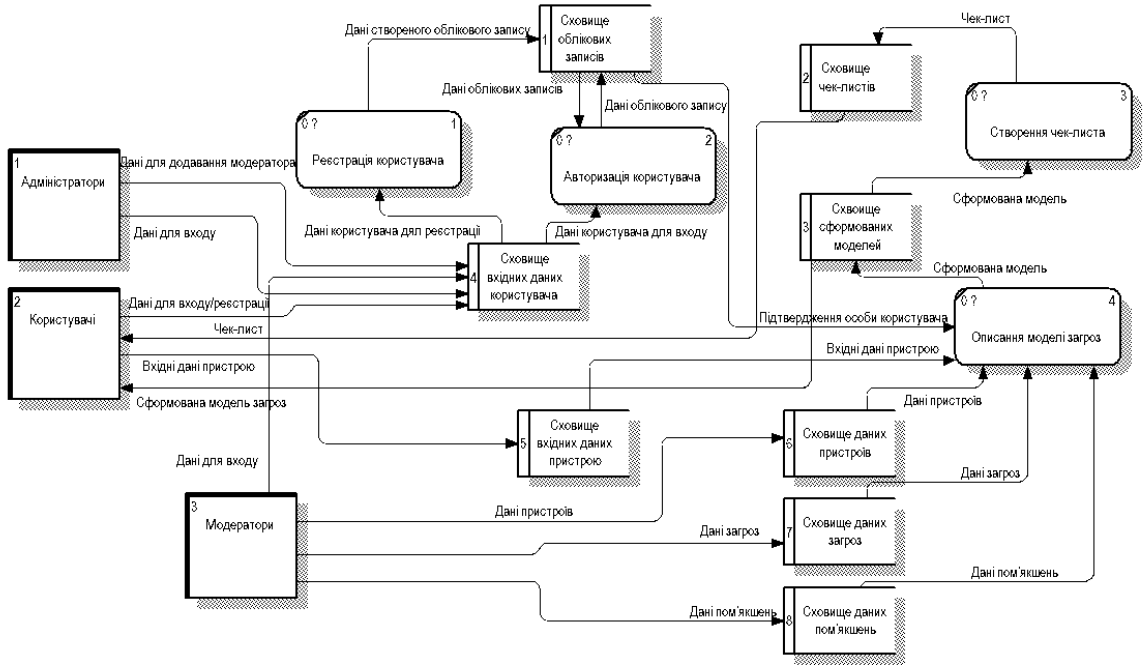


Рис. 2.5. Декомпозиція контекстної діаграми потоків даних формування моделі загроз безпеці мобільних пристроїв у нотації DFD

Розглянемо сховища, які слід додати. Спершу маємо сховище вхідних даних користувачів – сюди належать потоки даних для входу, від усіх користувачів, даних для реєстрації від звичайних користувачів та даних для додавання модератора від адміністраторів. Оскільки це все вхідні дані

користувачів. Звідси до реєстрації надходять реєстраційні дані, а до авторизації – дані для входу.

Від процесу реєстрації отримуємо потік даних до сховища облікових записів, а саме потік даних створеного облікового запису. Також процес авторизації отримує та надсилає дані облікового запису для перевірки справжності даних користувача. Окрім того, з цього сховища до процесу описання моделі загроз надходить потік «Підтвердження особи користувача», оскільки користувач має бути авторизованим для формування моделі.

Також від користувачів маємо потік вхідних даних пристрою, який прямує до сховища таких даних, із якого потім аналогічний потік – до описання моделі загроз, аби на основі моделі пристрою сформувати її. Окрім того, до цього процесу з відповідних сховищ ідуть потоки даних пристроїв, загроз та пом’якшень, куди вони потрапляють через аналогічні потоки, що надходять від модераторів. З цього процесу маємо вихідний потік даних – сформовану модель, яка передається у сховище сформованих моделей, звідки передається у процес створення чек-листа.

Отримавши сформовану модель, процес створення чек-листа формує його і на вихід надходить отриманий результат, передаючи його у сховище чек-листів. Сформована модель зі сховища сформованих моделей відтак прямує до користувача як і сформований чек-лист зі сховища чек-листів.

2.4. Висновки за розділом 2

Побудовано діаграму діяльності формування моделі загроз безпеці мобільних пристроїв у графічній нотації IDEF0. Виходячи з даної діаграми можна краще зрозуміти, на які функції поділяється зазначена діяльність. Крім того, що на неї впливає, що вона використовує для своїх дій, що приймає на вхід, та що віддає на вихід, чим обмежується. Так, було

встановлено, що загальний процес формування моделі загроз безпеці мобільних пристроїв поділяється на такі функції:

- реєстрація користувача;
- авторизація користувача;
- описання моделі на основі вхідних даних та інформації в системі;
- створення чек-листа.

Крім того побудовано діаграму процесів і зв’язків між ними в графічній нотації IDEF3. Це дозволило визначати всі розгалуження та прийняття рішень, які передбачаються процесом формування моделі загроз, а також хто саме впливає на виокремлені процеси. Так, задано наступну послідовність дій: спершу користувач або реєструється, а потім переходить до авторизації, або відразу переходить до авторизації, потім обирає модель власного пристрою, після чого переходить до етапу описання моделі загроз, а відтак – до процесу створення чек-листа. Єдиним, хто впливає на процес, є сам користувач.

Відтак було побудовано діаграму потоків даних формування моделі загроз безпеці мобільних пристроїв у графічній нотації DFD. Така діаграма дозволяє виокремити джерела та сховища даних. Таким чином, для чотирьох визначених функціональних блоків джерелами даних є: користувач, модератори та адміністратори. Серед функціональних блоків зосереджено увагу на реєстрації користувача, авторизації користувача, описанні моделі загроз та створенні чек-листа. Тоді як до сховищ віднесено сховище даних пристроїв, сховище даних пом’якшень, сховище даних загроз, сховище вхідних даних користувача, сховище вхідних даних пристрою, сховище облікових записів, сховище сформованих моделей загроз та сховище чек-листів.

Отже, побудовані діаграми загалом утворюють концептуальну модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв та допомагають розуміти зсередини концепт даної діяльності.

3. ОБ'ЄКТНО-ОРІЄНТОВАНА МОДЕЛЬ ВЕБЗАСТОСУНКУ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

3.1. Варіанти використання вебзастосунку формування моделі загроз безпеці мобільних пристроїв

У попередньому розділі формалізовано діяльність формування моделі загроз безпеці мобільних пристроїв у графічній нотації IDEF0, IDEF3, DFD. Визначено зв'язки між її етапами та потоки даних між ними. Це все необхідне для узагальненого розуміння діяльності формування моделі загроз безпеці мобільних пристрої.

Наступним кроком розроблення вебзастосунку є перехід на нижчий рівень абстракції – тобто описання реалізації без урахування конкретних засобів розробки, натомість з точки зору загальних для різних засобів компонентів та аспектів. Відтак для початку розглянемо найпростішу і найважливішу частину – варіанти використання. Отже, побудуємо відповідну діаграму у графічній нотації UML.

Діаграма варіантів використання – це одна з основних форм представлення функціональних вимог до програмного забезпечення, що розробляється. Такий формат відображає дії, які мають бути реалізовані в системі відносно акторів, між ними без деталізування того, як саме це має відбуватися. Діаграма варіантів використання розробляється з точки зору кінцевого користувача, таким чином, відображається вся зовнішньо видима функціональність програмного забезпечення.

Крім того, зазначеною діаграмою відображаються лише варіанти використання і не враховується послідовність, логіка вибору, умови та інше. Вона включає в себе акторів (користувачів системи) та блоки варіантів використання. Стрілки від акторів суцільні, тоді як пунктирні позначають включення варіантів використання в більш узагальнені [11].

Таким чином, побудуємо діаграму (рис. 3.1). Спершу виділимо акторів, яких відобразимо на діаграмі:

- користувач;
- модератор;
- адміністратор.

Основним актором є користувач, який реєструється та, власне, займається першочергово формуванням моделі загроз безпеці мобільних пристроїв та чек-листів. Модератор займається додаванням та відстеженням коректності інформації, в той час як адміністратор керує модераторами.

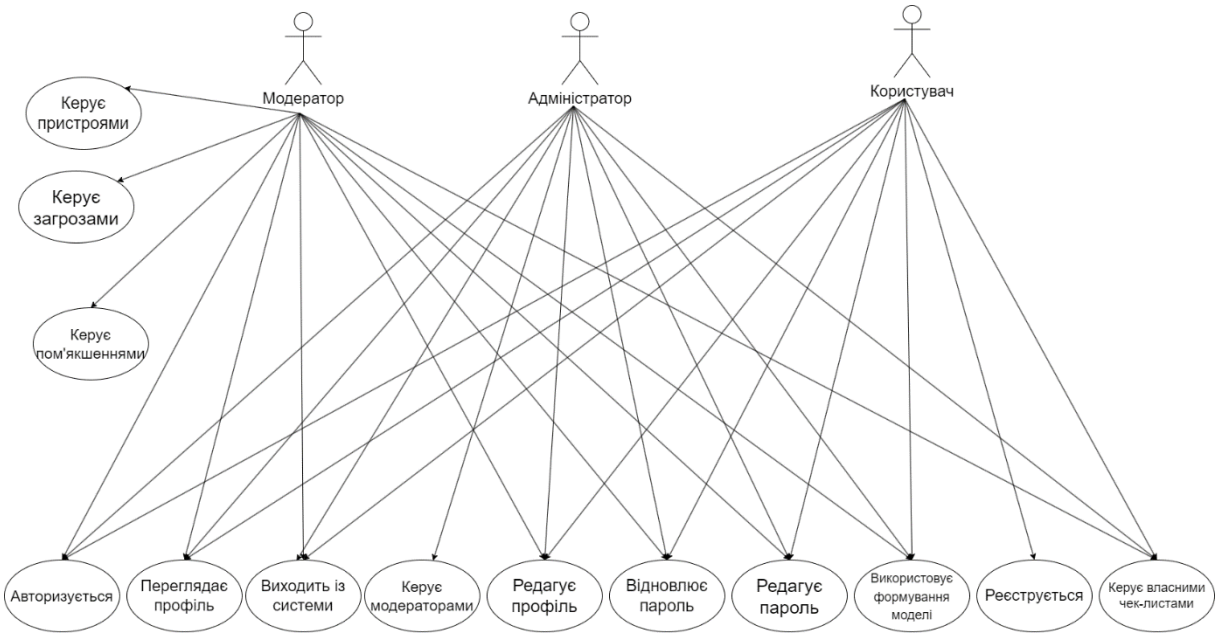


Рис. 3.1. Діаграма варіантів використання вебзастосунку у графічній нотації UML

Тепер визначимо, які є варіанти використання вебзастосунку для кожного з вищезазначених акторів.

Звичайний користувач у першу чергу може реєструватися, що є необхідним для роботи з вебзастосунком, аби прив’язувати чек-листи до користувача.

Після реєстрації відкривається ще низка можливостей, у той чи інший спосіб пов’язаних з обліковим записом користувача. Перш за все, користувач має мати змогу авторизуватися в системі, аби взагалі могти її використовувати. Відповідно, також передбачається можливість вийти із системи, наприклад якщо необхідно змінити обліковий запис, або якщо користувач авторизувався у публічному місці і не хоче там зберігати сесію у вебзастосунку аби інші люди не мали доступу до його акаунту.

Після авторизації можна переглянути власний профіль. Також є можливість його відредагувати, якщо, наприклад, користувач неправильно ввів дані чи вирішив змінити ім’я. Окрім того, передбачається можливість редагування пароля, наприклад, якщо користувач вирішив, що його пароль недостатньо надійний, а також можливість відновлення пароля, якщо він його забув.

Вочевидь після реєстрації стає доступною основна функціональність, а саме формування моделі загроз безпеці мобільних пристроїв. Це і є ще одним варіантом використання. До того ж керування власними чек-листами, оскільки вони є особистими для користувача.

Розглянемо варіанти використання для модератора. В першу чергу йому доступні дії з акаунтом, як і звичайному користувачеві, а саме він може авторизуватися, переглянути свій профіль, відредагувати його, вийти з системи, а також змінити чи відновити пароль. Також модератори можуть використовувати функцію формування моделі загроз та керування чек-листами, аби мати змогу переконатися в коректності інформації в системі, за потреби.

Втім, модератор не має змоги самотужки зареєструватися, позаяк модераторів реєструють адміністратори задля перевіреності модеруючих осіб. Також, оскільки модератори відповідають за наповнення системи, то вони мають низку варіантів використання, не передбачених для користувача. Таким чином, модератори можуть керувати пристроями, щоб

користувачі могли обирати модель, а не вводити параметри пристрою при її формуванні, а також керувати загрозами та пом’якшеннями, які вносяться в до неї.

Тепер перейдемо до варіантів використання адміністраторів. Перш за все, як і обом попереднім типам акторів, йому доступні дії з акаунтом – авторизація, вихід із системи, перегляд та редагування профілю, зміна та відновлення паролю та формування моделі загроз і керування чек-листами. Проте, як і у випадку з модератором, адміністратор не реєструється самостійно. Його також не додають через користувацький інтерфейс застосунку, він додається відразу в систему.

Також, на додачу до вищезазначених варіантів використання, адміністратор, як уже згадувалося, має змогу керувати модераторами, оскільки модераторів може бути багато, відтак іноді може відбуватися зміна діяльності, тому для керування модераторами краще мати користувацький інтерфейс.

Отже, користувач має такі варіанти використання:

- авторизується;
- реєструється;
- переглядає профіль;
- виходить із системи;
- редагує профіль;
- відновлює пароль;
- редагує пароль;
- використовує формування моделі загроз;
- керує власними чек-листами.

Модератор має такі варіанти використання:

- авторизується;
- переглядає профіль;
- виходить із системи;

- редагує профіль;
- відновлює пароль;
- редагує пароль;
- використовує формування моделі загроз;
- керує власними чек-листами;
- керує пристроями;
- керує загрозами;
- керує пом’якшеннями.

Адміністратор має такі варіанти використання:

- авторизується;
- переглядає профіль;
- виходить із системи;
- редагує профіль;
- відновлює пароль;
- редагує пароль;
- використовує формування моделі загроз;
- керує власними чек-листами;
- керує модераторами.

Тепер розглянемо деякі варіанти використання детальніше. Перш за все, формування моделі загроз включає в себе декілька частин, а саме:

- вказання моделі пристрою;
- перегляд сформованої моделі;
- створення чек-листа.

Чек-листи створюються в межах цього варіанту використання з огляду на те, що складаються вони на основі сформованих моделей. Також варіанти використання, які передбачають керування, містять в собі ті частини, які відповідають CRUD-можливостям, доцільним та передбаченим для кожного виду об’єкта (пристрій, загроза).

Отже, в рамках варіанту використання керування пристроями модератор:

- переглядає пристрої;
- додає пристрої;
- редагує пристрої;
- видаляє пристрої.

В межах варіанту використання керування загрозами:

- переглядає загрози;
- додає загрози;
- редагує загрози;
- видаляє загрози.

У випадку пом’якшень:

- переглядає пом’якшення;
- додає пом’якшення;
- редагує пом’якшення;
- видаляє пом’якшення.

Під час керування модераторами адміністратор:

- переглядає модераторів;
- реєструє модераторів;
- активує/деактивує модераторів.

Активація/деактивація передбачена щоб не видаляти облікові записи з системи, оскільки вони можуть знадобитися знову, або можуть деактивуватися на якийсь період, скажімо, при виникненні певної ситуації.

Під час керування чек-листами користувач, модератор чи адміністратор:

- переглядає чек-листи;
- ставить відмітки у чек-листі;
- видаляє чек-листи;
- змінює назву чек-листа.

Як зазначалося раніше, створення чек-листів належить до варіанту використання формування моделі загроз.

3.2. Логічна структура вебзастосунку формування моделі загроз безпеці мобільних пристроїв

3.2.1. *Статична логічна структура вебзастосунку формування моделі загроз безпеці мобільних пристроїв*

Після визначення варіантів використання вебзастосунку формування моделі загроз безпеці мобільних пристроїв доцільно перейти до подальшої конкретизації його структури. Одним із завдань, що при цьому вирішуються, є розроблення логічної структури в статичному та динамічному аспектах.

Статична логічна структура представляється діаграмою класів у графічній нотації UML. Нею відображаються класи, їхні атрибути та методи, а також зв’язки між ними. Кожен атрибут має тип, а кожен метод має тип значення яке повертається і також може мати або не мати параметри якогось типу [12].

Перейдемо до побудови діаграми (рис. 3.2). Виділено такі класи в межах розроблення логічної структури вебзастосунку формування моделі загроз безпеці мобільних пристроїв:

- Device – клас пристрою;
- Checklist – клас чек-листа;
- ChecklistEntry – клас запису в чек-листі;
- Technique – загроза, техніка, шо використовується зловмисником для атаки;
- Mitigation – пом’якшення;
- Applicability – застосовність, описує в яких випадках певна техніка може бути загрозою;
- Credentials – облікові дані, дані для входу;

- Account – інші дані облікового запису.

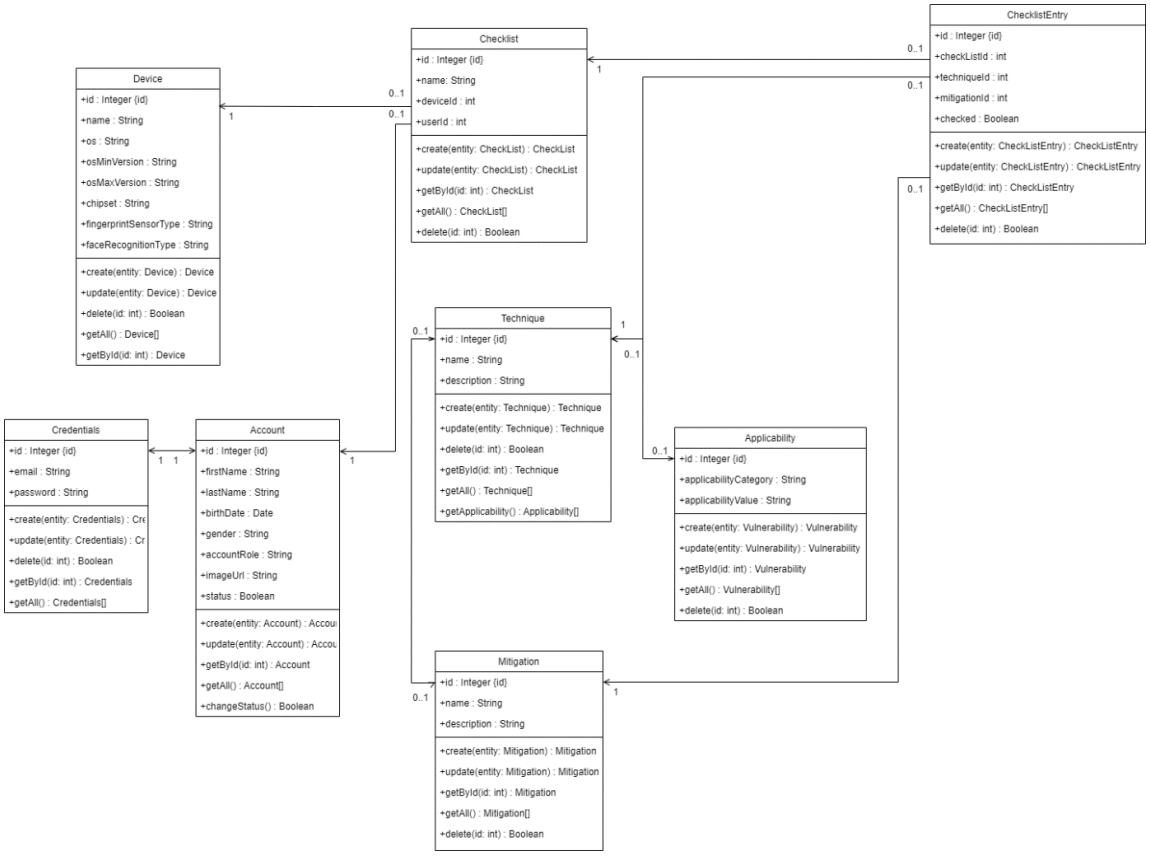


Рис. 3.2. Діаграма класів вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML

Загалом усі ці класи (рис. 3.2), за винятком облікового запису, мають одні й ті ж стандартні методи:

- create(entity: Entity): Entity – створення екземпляру класу;
- update(entity: Entity): Entity – оновлення екземпляру класу;
- delete(id: int): boolean – видалення екземпляру класу;
- getAll(): Entity[] – отримання всіх екземплярів класу;
- getById(id: int): Entity – отримання екземпляру класу за ідентифікатором.

де Entity – це клас, якому належить метод. Клас Account замість методу delete має метод changeStatus(): Boolean, позаяк облікові записи не видаляються, проте можуть бути деактивовані якщо належать модераторам.

Також клас Technique містить додатковий метод getApplicability, який потрібен аби визначити сфери застосовності певної загрози.

Тепер розглянемо зв’язки між класами (рис. 3.2).

- 1 до багатьох між Device та Checklist, оскільки для одного й того ж пристрою можуть формувати модель і, відповідно, створювати чек-лист різні користувачі;
- 1 до багатьох між Checklist та ChecklistEntry, бо один чек-лист має багато рядків;
- 1 до багатьох між Technique та ChecklistEntry, бо одна й та сама загроза може зустрічатися в різних чек-листах;
- 1 до багатьох між Mitigation та ChecklistEntry, бо одне й те саме пом’якшення може зустрічатися в різних чек-листах;
- багато до багатьох між Technique та Applicabilityпере, оскільки різні техніки можуть стосуватися одних і тих самих особливостей пристрою, а різні особливості можуть стосуватися однієї техніки;
- багато до багатьох між Technique та Mitigation, оскільки різні пом’якшення можуть стосуватися однієї й тієї ж загрози, а різні загрози можуть стосуватися одного пом’якшення;
- 1 до багатьох між Account та Checklist, позаяк один користувач може мати багато чек-листів;
- 1 до 1 між Credentials та Account, оскільки одні облікові дані співвідносяться лише з одним обліковим записом, а потрібно це з міркувань безпеки та уникнення передачі надмірної кількості даних.

Далі визначимо атрибути кожного класу (рис. 3.2).

Device:

- id – унікальний ідентифікатор пристрою в системі, унікальний і є числом (Integer);
- name – найменування пристрою, String;

- os – вид операційної системи, представляється перечисленням типу String;
- osMinVersion – початкова версія операційної системи, String;
- osMaxVersion – до якої версії операційної системи оновлювався/ оновлюватиметься пристрій, String;
- chipset – процесор, String;
- fingeprintSensorType – вид сканера відбитків пальців (ємнісний/ оптичний/ультразвуковий/відсутній), представляється перечисленням типу String;
- faceRecognitionType – вид розпізнавання обличчя (камерою/ інфрачервоним датчиком/3D-скануванням/відсутнє), представляється перечисленням типу String.

Technique:

- id – унікальний ідентифікатор загрози в системі, унікальний і є числом (Integer);
- name – найменування загрози/техніки, String;
- description – опис, String.

Mitigation:

- id – унікальний ідентифікатор загрози в системі, унікальний і є числом (Integer);
- name – найменування пом’якшення, String;
- description – опис, String.

Applicability:

- id – унікальний ідентифікатор вразливості в системі, унікальний і є числом (Integer);
- applicableCategory – категорія, до якої застосовується загроза, одне з полів класу Device, від яких можуть залежати вразливості, Enumeration типу String;

- applicableValue – значення вразливості відповідної категорії, String.

Checklist:

- id – унікальний ідентифікатор чек-листа в системі, унікальний і є числом (Integer);
- deviceId – ідентифікатор відповідного пристрою, Integer;
- userId – ідентифікатор відповідного користувача, якому належить чек-лист, Integer.

ChecklistEntry:

- id – унікальний ідентифікатор запису в чек-листі в системі, унікальний і є числом (Integer);
- checklistId – ідентифікатор відповідного чек-листа, Integer;
- techniqueId – ідентифікатор відповідної загрози, Integer;
- mitigationId – ідентифікатор відповідного пом’якшення, Integer;
- checked – позначає, чи відмічений запис у чек-листі, Boolean.

Account:

- id – унікальний ідентифікатор облікового запису в системі, унікальний і є числом (BigInteger);
- firstName – ім’я, String;
- lastName – прізвище, String;
- birthDate – дата народження, Date;
- gender – стать, Enumeration типу String;
- accountRole – Enumeration типу String, роль користувача (користувач, модератор чи адміністратор);
- imageUrl – посилання на аватарку, String;
- status – активований чи неактивований користувач, Boolean. Може бути порожнім, або заповненим якщо модератора деактивували принаймні раз.

Credentials:

- id – унікальний ідентифікатор облікових даних в системі, унікальний і є числом (BigInteger), збігається з id відповідного класу Account;
- email – електронна пошта користувача, String;
- password – захешований пароль, String.

3.2.2. *Динамічна логічна структура вебзастосунку формування моделі загроз безпеці мобільних пристроїв*

Наступним кроком після визначення статичної логічної структури є створення динамічної. Для цього за основу візьмемо діаграму діяльності у графічній нотації UML.

Діаграма діяльності є діаграмою динамічної структури програмного забезпечення і являє собою по суті блок-схему яка поєднує в собі виконання дій (діяльностей). Така діаграма показує послідовність дій у застосунку, а також місця розгалужень та прийняття рішень. Містить блоки діяльностей, блоки дій, блоки розгалуження, блоки початку й кінця, блоки даних та блоки результату [13].

Побудуємо діаграму діяльностей у графічній нотації UML (рис. 3.3-3.7). Спочатку виконується дія «Вхід у систему». На початку відразу ж маємо розгалуження. Користувач має три опції:

- відразу перейти до авторизації;
- обрати реєстрацію;
- обрати відновлення паролю.

У випадку обрання реєстрації виконується діяльність «Введення реєстраційних даних». Вносяться реєстраційні дані, і, як наслідок, користувача зареєстровано. У випадку обрання відновлення паролю відбувається введення даних для відновлення паролю, після чого пароль відновлюється.

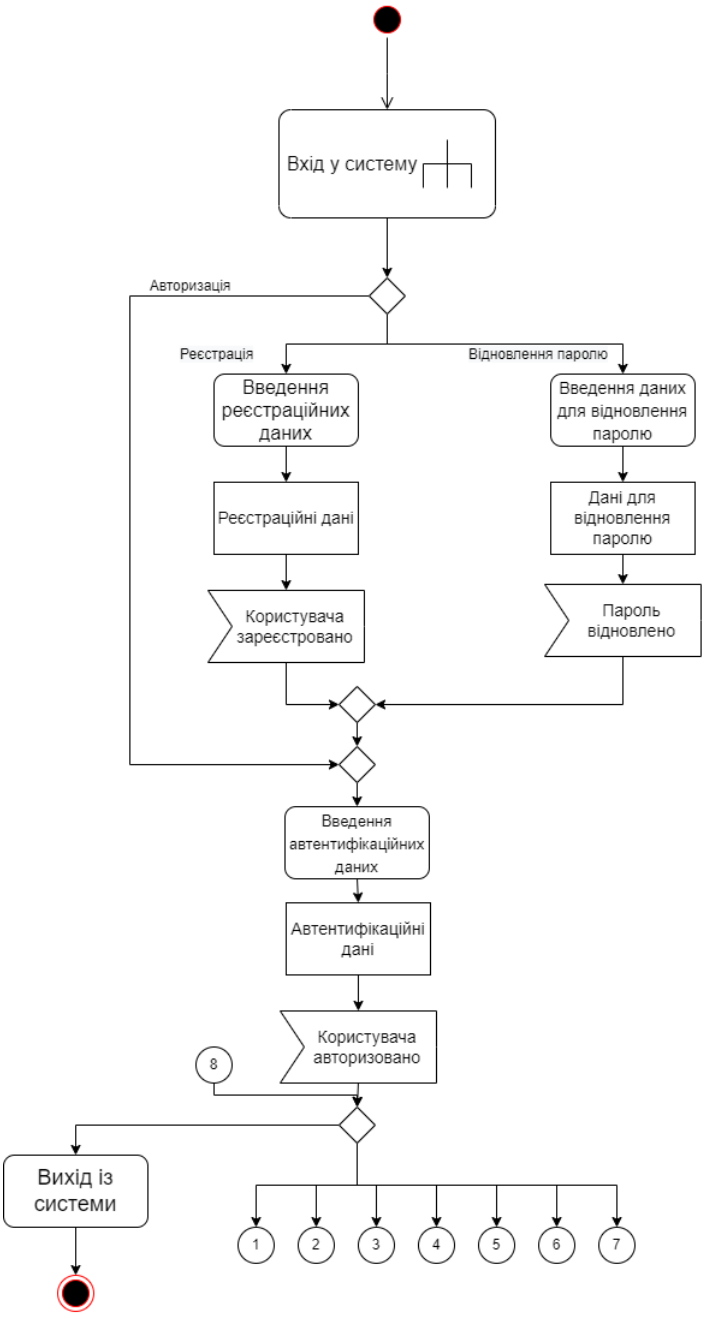


Рис. 3.3. Діаграма діяльності вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 1

Після цих кроків як і після реєстрації можна перейти назад до початку діяльності. Інакше відбувається перехід до власне авторизації. Спершу відбувається введення автентифікаційних даних, після чого користувача авторизовано.

Далі маємо розгалуження на різні варіанти використання вебзастосунку, оскільки вони всі доступні відразу після авторизації. Можливий варіант відразу вийти з системи, проте є й інші діяльності. Після кожної з них іде розгалуження, за яким можна або повернутися до її початку, або повернутися до розгалуження між діяльностями.

Перша діяльність це керування пристроями (рис. 3.4). Тут іде розгалуження на різні варіанти керування, можна додавати пристрої, після чого їх додано, можна редагувати, після чого відредаговано, можна їх видаляти, після чого їх видалено, а також можна шукати, тоді надаються їхні дані.

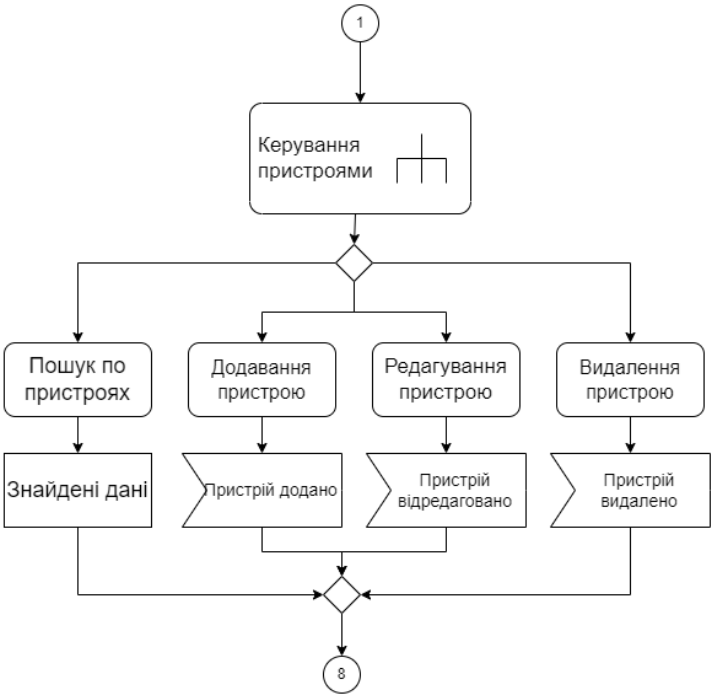


Рис. 3.4. Діаграма діяльності вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 2

Друга та третя діяльність це керування загрозами та пом’якшеннями, які працюють повністю аналогічно і поділяються на додавання, редагування, видалення та пошук (рис. 3.5, 3.6).

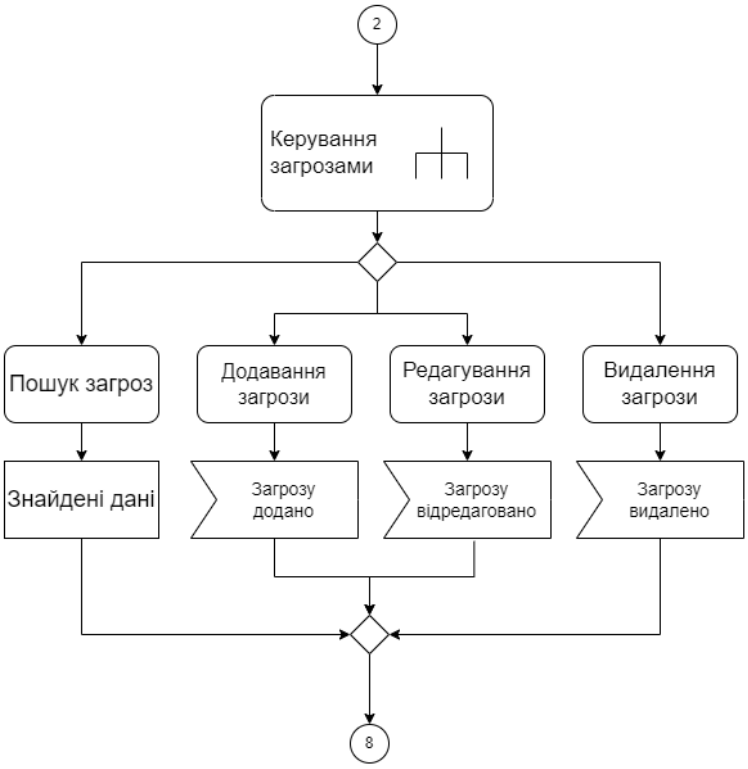


Рис. 3.5. Діаграма діяльності вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 3

Четверта діяльність це формування моделі загроз безпеці мобільних пристроїв (рис. 3.6). Першим кроком є обрання моделі пристрою. Далі відбувається отримання моделі, вона надається користувачеві. Потім створюється чек-лист, і, як результат, його створено.

Наступною діяльністю є керування профілем (рис. 3.6). Маємо розгалуження, можна або переглянути профіль, тоді отримаємо його дані, або його відредагувати, тоді його буде відредаговано.

Шостою діяльністю є керування модераторами (рис. 3.6). Вона схожа на керування пристроями, загрозами та пом’якшеннями, проте відсутнє редагування і замість видалення маємо деактивацію/активацію. Таким чином, маємо додавання модератора, внаслідок чого його додано, його деактивацію чи активацію, внаслідок чого його деактивовано чи активовано та пошук модераторів, в разі чого повертаються знайдені дані.

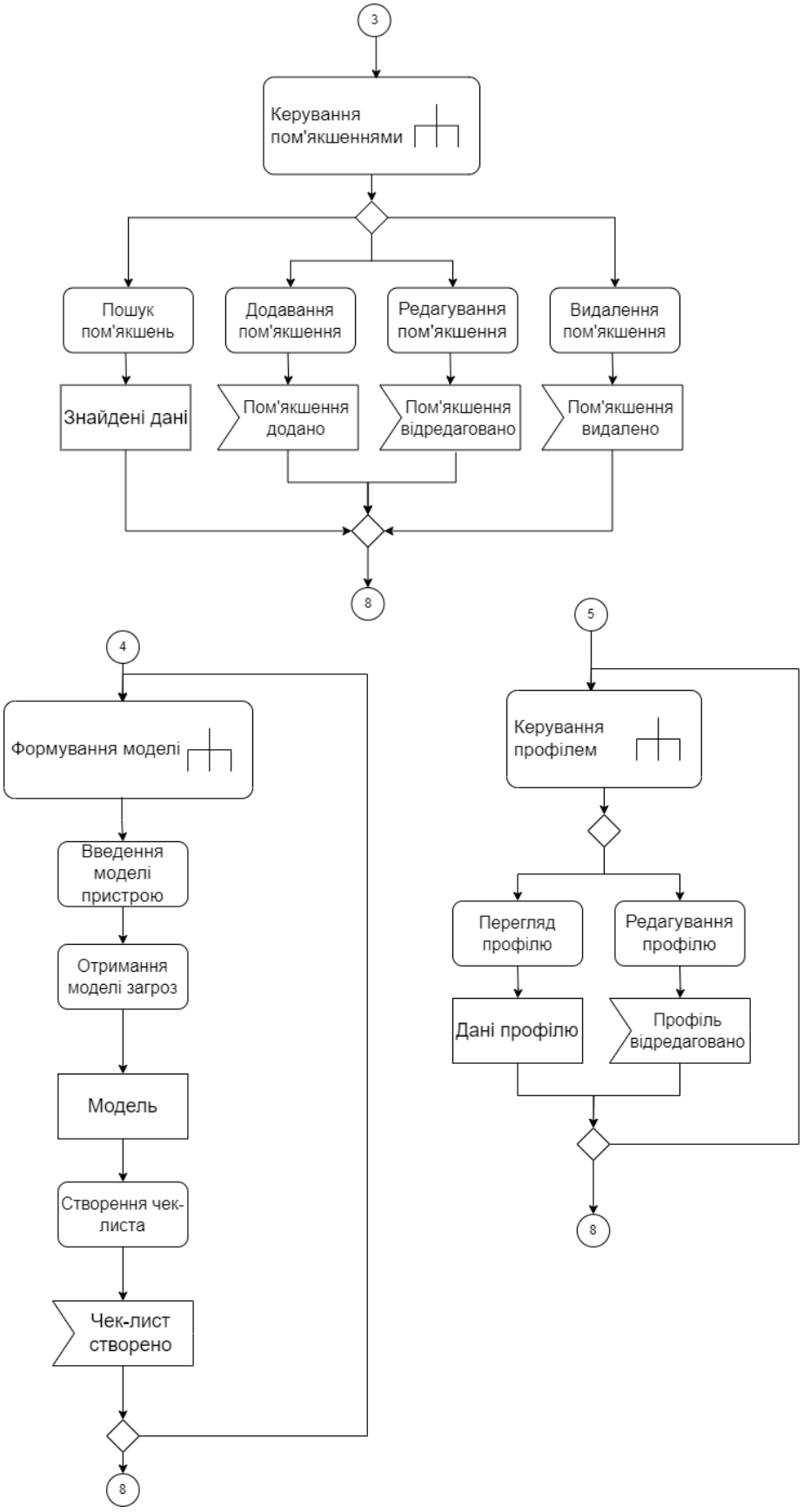


Рис. 3.6. Діаграма діяльності вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 4

Наостанок маємо діяльність керування чек-листами (рис. 3.7). Вона також аналогічна керуванню пристроями, пом’якшеннями та загрозами, проте не має створення, оскільки воно належить до діяльності формування моделі загроз. Таким чином, маємо пошук, редагування та видалення.

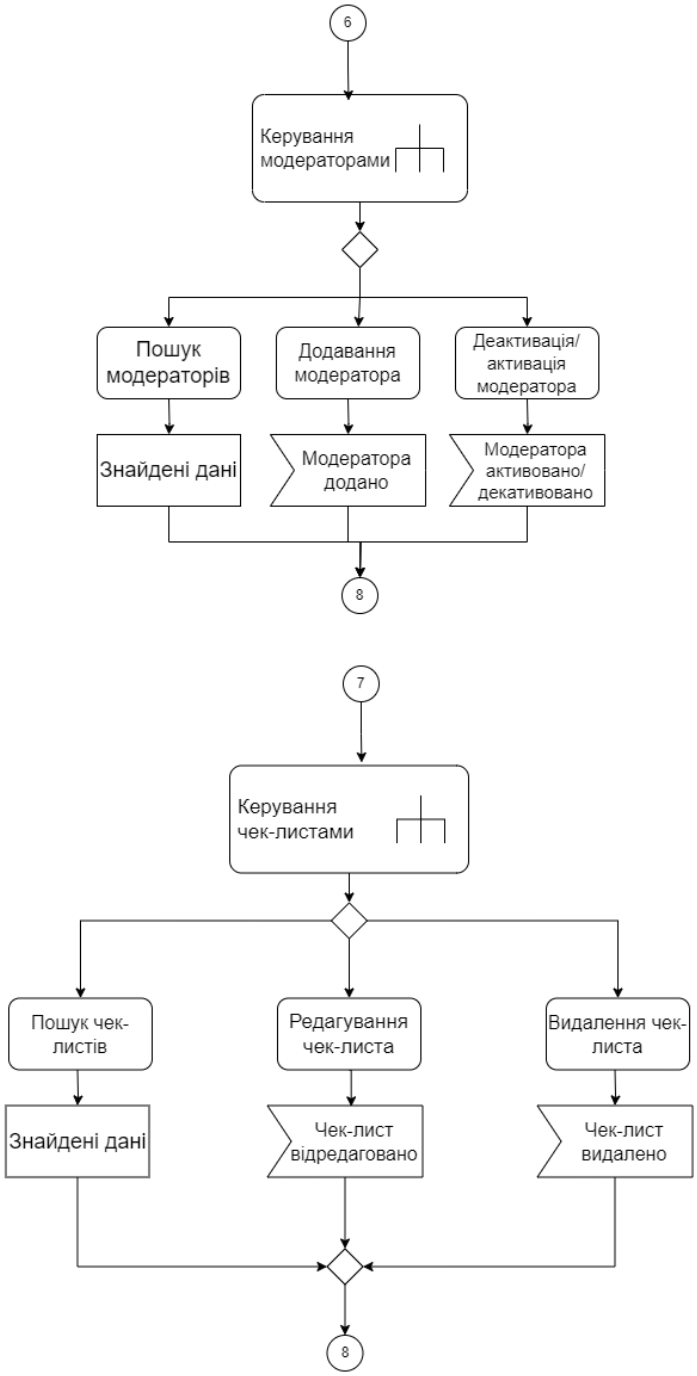


Рис. 3.7. Діаграма діяльності вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 5

3.3. Фізична структура вебзастосунку формування моделі загроз безпеці мобільних пристроїв

3.3.1. *Компоненти та зв'язки між ними*

На основі отриманої логічної структури створимо фізичну структуру вебзастосунку. Фізична структура все ще незалежна від конкретних засобів розробки, проте вже описує саму архітектуру програмного забезпечення. Тобто не дії та сутності, а його частини. Для того щоб визначити фізичну структуру спершу виокремимо компоненти вебзастосунку та зв'язки між ними.

Для цього побудуємо діаграму компонентів у графічній нотації UML (рис. 3.8). Такою діаграмою розбивається вебзастосунок на високорівневі частини, кожна з яких виконує свою роль і пов'язана з іншими [14]. Вона містить групи компонентів, а також з'єднання, які позначають зв'язки між компоненти.

Перша група компонентів це клієнт/фронтенд (рис. 3.8). Вона відображає клієнтську частину, тобто ту частину, з якою взаємодіє користувач і яку можна відкрити в браузері. Всередині ця група містить два компоненти.

Перший компонент – це інтерфейс користувача, тобто власне та частина, з якою взаємодіють користувачі та яка графічно відображається. Отримані дані цей компонент передає наступному, а щоб повернути їх користувачу він отримує дані від наступного компоненту.

Наступний компонент – це сервіси клієнта. Він відповідає за логіку з боку клієнта. Обробляє отриману від користувача та від сервера інформацію і передає далі на сервер або до компоненту користувацького інтерфейсу.

Друга група компонентів це допоміжні модулі клієнту (рис. 3.8). Вони обидва пов'язані з групою компонентів клієнта через зв'язок із компонентом сервісів. Перший компонент цієї групи це інтерсептор. Цей компонент відповідає за те, щоб з кожним запитом до сервера відправлялася певна

визначена інформація, наприклад JWT-токен, який відповідає за авторизованість. Другий – це кукі-сховище, яке керує збереженням у постійному сховищі браузера певних даних аби не вводити їх щоразу. Наприклад, кукі-сховище необхідне для збереження JWT-токена, інакше з кожним запитом довелося б відправляти облікові дані.

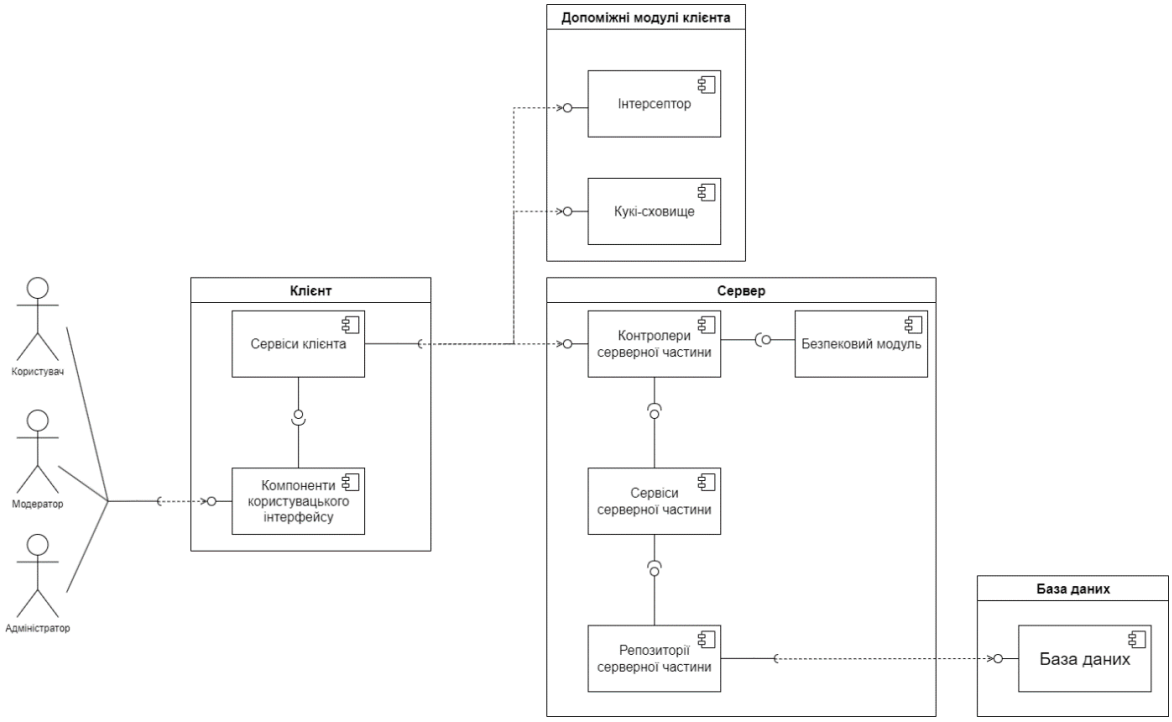


Рис. 3.8. Діаграма компонентів вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML

Третя група компонентів це сервер/бекенд (рис. 3.8). Вона позначає серверну частину, тобто ту, яка обробляє та зберігає дані, і з якою спілкуються екземпляри клієнту. Ця група містить чотири компоненти.

Перший компонент – це контролери серверної частини. Він відповідає за комунікацію з клієнтом і просто приймає від нього та надсилає йому інформацію відповідно до запиту.

Другий компонент – це безпековий модуль, який пов’язаний лише з компонентом контролерів та забезпечує аби доступ до різних ресурсів серверу був лише в тих, хто його дійсно має мати.

Третій компонент – це сервіси серверної частини. Тут міститься бізнес-логіка застосунку, він обробляє дані зі сховища і передає контролерам, також отримує від них дані.

Останнім компонентом цієї групи є репозиторії серверної частини. Вони являють собою сховище, яке напряму взаємодіє з базою даних, приймає дані з сервісів та записує в базу, і віддає на запит дані з бази, не проводячи при цьому взагалі або проводячи мінімальну обробку.

Остання група компонентів це база даних, яка містить в собі єдиний компонент – власне базу даних. Тут зберігаються усі дані, розміщені та організовані по таблицях.

3.3.2. *Вузли та зв'язки між ними*

Для завершення моделювання фізичної структури визначимо вузли вебзастосунку та зв'язки між ними. З цією метою побудуємо діаграму розгортання у графічній нотації UML (рис. 3.9, 3.10). Така діаграма відображає структуру вебзастосунку вузлами (середовищем виконання) та зв'язками між ними, а також їхній вміст. Показує конфігурацію, в якій застосунок має розгортатися. Містить власне вузли, зв'язки, артефакти та компоненти. Також у вигляді артефактів з'являються засоби розробки, оскільки це напряму пов'язано з самими вузлами [15].

Маємо три типи користувачів – користувача, модератора та адміністратора (рис. 3.9). Через комп'ютерну периферію вони пов'язані з пристроєм користувача зв'язком багато до багатьох. Пристрій користувача містить внутрішній вузол – браузер. Браузер має артефакти HTML, CSS та JavaScript. Кожен браузер повинен їх розпізнавати аби могти відображати сторінки веб-сайтів.

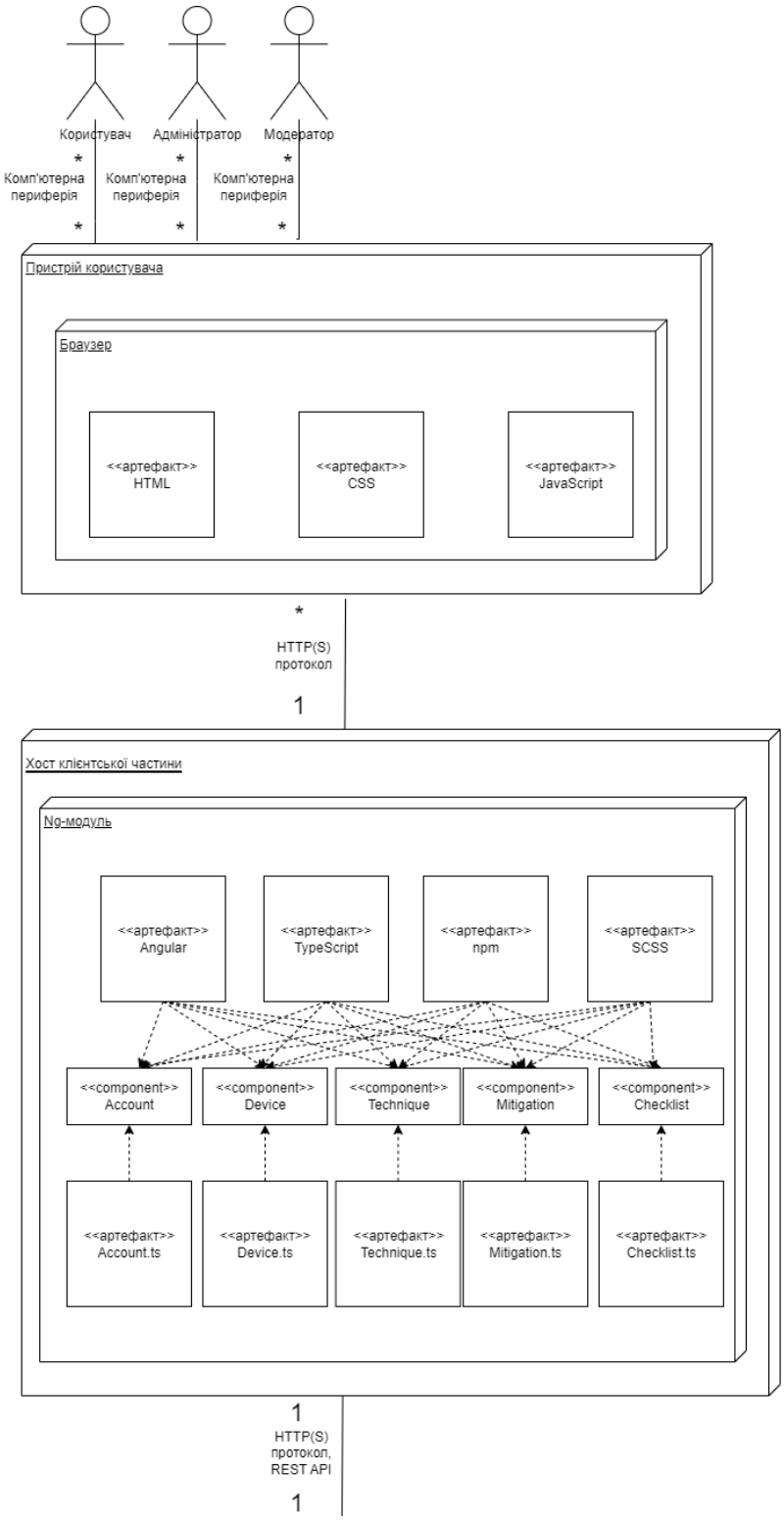


Рис. 3.9. Діаграма розгортання вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 1

Далі маємо другий вузол, хост клієнтської частини (рис. 3.9), з яким комп’ютер користувача пов’язаний через HTTPS-протокол зв’язком багато

до 1, оскільки багато користувачів підключаються до одного хосту. Хост містить в собі Ng-модуль, тобто спеціальний модуль який є середовищем виконання для Angular. Всередині маємо артефакт Angular, який є фреймворком для клієнта, артефакт TypeScript, який є мовою програмування, артефакт npm, який є менеджером пакетів, а SCSS, який є розширеним варіантом CSS та містить логічні конструкції. Далі маємо компоненти, на діаграмі виділено основні – Account, Device, Technique, Mitigation та Checklist, а також відповідні артефакти з розширенням .ts.

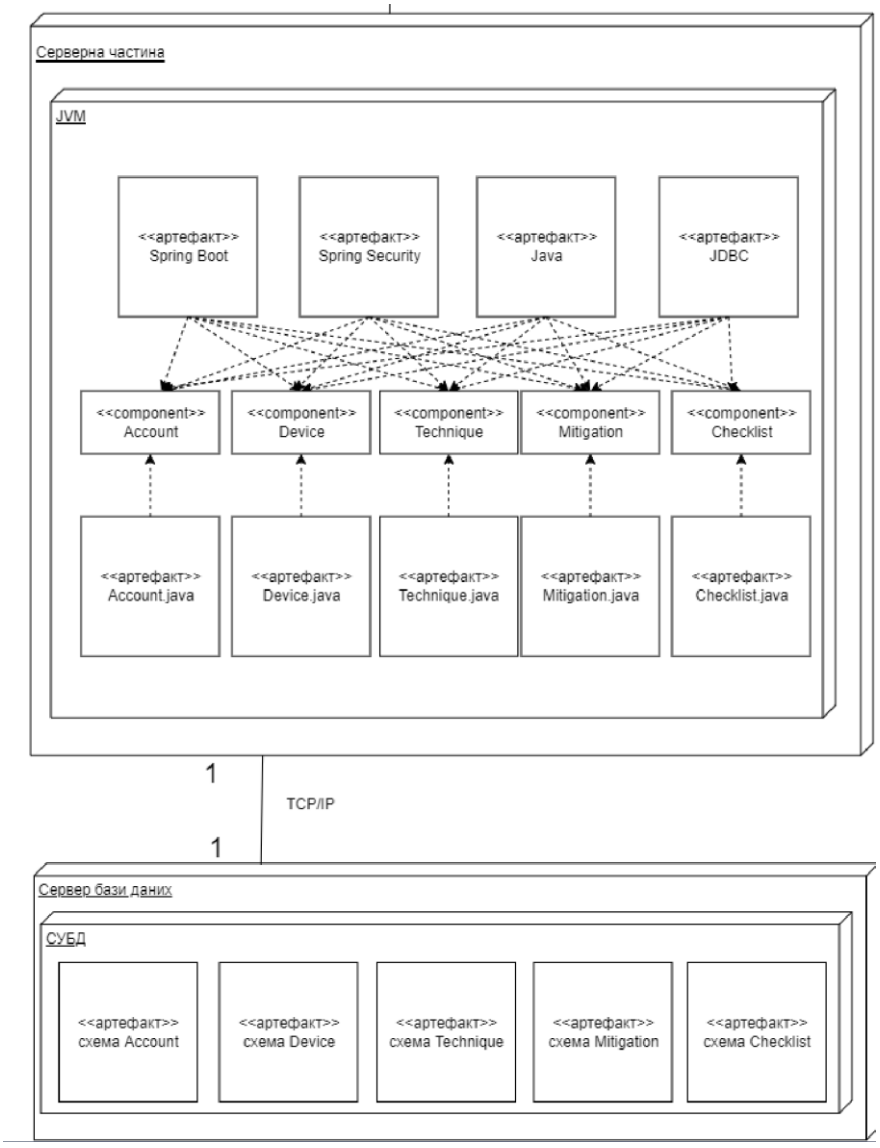


Рис. 3.10. Діаграма розгортання вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML, частина 2

Наступний вузол це серверна частина, зв’язок 1 до 1 через HTTPS-протокол та REST API. Внутрішнім вузлом є JVM, тобто віртуальна машина Java, яка є середовищем виконання для застосунків на Java. Тут артефактами є фреймворк для серверу Spring Boot, безпековий фреймворк Spring Security, власне Java та драйвер JDBC для зв’язку з базою даних зі застосунку на Java. Також маємо аналогічні компоненти та артефакти як у вузлі клієнту, але з розширенням .java.

Останнім вузлом, поєднаним через зв’язок 1 до 1 по TCP/IP є сервер бази даних. Він містить внутрішній вузол СУБД, а внутрішні артефакти це основні схеми – схема Account, схема Device, схема Technique, схема Mitigation та схема Checklist.

3.4. Висновки за розділом 3

Розроблено об’єктно-орієнтовану модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв. Для цього спочатку визначено його варіанти використання у графічній нотації UML. Побудова такої діаграми обумовлено необхідністю встановлення можливостей різних користувачів вебзастосунку. Виокремлено всі види користувачів, які мають змогу авторизуватися, переглянути профіль, відредагувати профіль, відновити пароль, відредагувати пароль, сформувати модель загроз, керувати чек-листами та вийти з вебзастосунку. Також користувач може зареєструватися, модератор – керувати пристроями, загрозами та пом’якшеннями, а адміністратор – модераторами.

Статичну логічну структуру вебзастосунку відображено діаграмою класів. Отримано 8 класів та встановлено зв’язки між ними, а саме: Device, Checklist, ChecklistEntry, Technique, Mitigation, Applicability, Account та Credentials. Відтак представлено динамічну логічну структуру як діаграму діяльності. Визначено першочерговість діяльності входу у вебзастосунок. Після цього доступні діяльності керування пристроями, керування пом’якшеннями, керування загрозами, керування чек-листами, керування

профілем, керування модераторами та формування моделі загроз, а також вихід з вебзастосунку.

Наостанок розроблено фізичну структуру вебзастосунку. Для цього спершу побудовано діаграму компонентів. Визначено, що застосунок поділяється на компоненти користувацького інтерфейсу, сервіси клієнту, інтерсептор, кукі-сховище, контролери серверної частини, сервіси серверної частини, репозиторії серверної частини, безпековий модуль та базу даних. Насамкінець побудовано діаграму розгортання з такими вузлами: пристрій користувача, хост клієнтської частини, сервера частина та сервер бази даних.

4. РОБОЧИЙ ПРОЄКТ ВЕБЗАСТОСУНКУ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

4.1. Реалізація вебзастосунку формування моделі загроз безпеці мобільних пристроїв

У попередньому розділі визначено варіанти використання вебзастосунку формування моделі загроз безпеці мобільних пристроїв, а також його логічну та фізичну структури. На основі отриманих результатів перейдемо до розроблення робочого проєкту вебзастосунку і почнем з обирання засобів програмування. Насамперед мов програмування бекенду та фротенду, фреймворків бекенду та фронтенду, за потреби, бібліотек, а також СУБД.

Почнемо з обирання мови програмування бекенду. Розглянемо деякі популярні мови для написання бекенду та порівняємо їх, а саме [16]: JavaScript, Python та Java.

Почнемо з JavaScript. Одна з ключових переваг використання JavaScript як мови бекенду полягає в тому, що вона дає змогу розробникам використовувати одну мову як для фронтенду, так і для бекенду. Це може оптимізувати розробку та полегшити підтримання коду.

Однак, варто зазначити, що JavaScript зазвичай не використовується для надскладних або високопродуктивних завдань, таких як наукові обчислення або машинне навчання. У цих випадках краще підходять такі мови як Python, Java або C++. Крім того, JavaScript має динамічну типізацію, що ускладнює виявлення помилок і забезпечення правильності коду.

До переваг JavaScript можна віднести [16]:

- легкість та швидкість;
- мова дуже популярна, тому під неї багато документації та іншої інформації в мережі;
- дозволяє використовувати одну мову для фронтенду та бекенду;

- масштабованість;
- інтегрованість з іншими системами.

До недоліків можна віднести [16]:

- не підходить для розроблення мережевих програм;
- слабкий мовний дизайн може призвести до появи проблем;
- складне обслуговування;
- динамічно типізована;
- повільна у випадку бекенду, ніж Java чи C++.

Загалом JavaScript може бути ефективним вибором для створення вебзастосунків на сервері, особливо для невеликих або середніх проєктів, які не потребують високої продуктивності чи складних обчислень.

Тепер розглянемо мову Python [16, 17]. Python відомий своєю простотою використання, масштабованістю та універсальністю. Так само є популярним тому в мережі є досить багато інформації.

Однією з ключових переваг використання Python є простота використання та читабельність [17]. Python має простий і послідовний синтаксис, що полегшує розробникам написання та підтримання коду. Це може допомогти скоротити час розробки та підвищити загальну продуктивність.

Python також відомий своєю масштабованістю та універсальністю. Він може обробляти великі обсяги даних і трафіку, що робить його хорошим вибором для створення масштабованих вебзастосукнів і API. Його також можна використовувати для широкого спектру завдань.

Ще однією перевагою використання Python як базової мови є його велика екосистема бібліотек і фреймворків [17]. Існує багато сторонніх бібліотек і модулів, доступних для Python, які можуть допомогти розробникам швидко та легко додавати функціональність до своїх програм.

Проте одним потенційним недоліком використання Python як мови серверної частини є його продуктивність. Python є інтерпретованою мовою,

що означає, що він може працювати повільніше, ніж скомпільовані мови, такі як, наприклад, C++. Це може бути недоліком під час створення високопродуктивних програм, які вимагають великої обчислювальної потужності. Окрім того від динамічно типізований [17].

Загалом, Python є потужною та універсальною мовою, яка може бути хорошим вибором для веброзробки бекенду. Особливо для програм, які вимагають простоти використання, масштабованості та універсальності.

Наостанок розглянемо Java [16, 18, 19]. Java є популярною та широко використовуваною мовою для розробки бекенду. Java відома своєю надійністю, масштабованістю та функціями безпеки [18, 19].

Однією з ключових переваг використання Java є її стійкість і надійність. Java має стійку систему типів і статично типізується, що означає, що помилки можна виявляти на ранніх стадіях процесу розроблення [18, 19]. Це може допомогти покращити якість і стабільність програми з часом.

Java також відома своєю масштабованістю [18]. Вона дозволяє одночасно обробляти велику кількість користувачів і запитів.

Однак, одним із потенційних недоліків використання Java є складність опановування мови. Java має велику та складну екосистему інструментів, фреймворків і бібліотек. Хоча велика кількість бібліотек є водночас і перевагою [18]. Загалом Java є потужною та надійною мовою.

Таким чином, JavaScript, Python та Java є дуже популярними мовами, тому для них усіх є велика кількість документації, інформації в мережі Інтернет та бібліотек. Також вони всі добре масштабуються. Проте JavaScript та Python динамічно типізовані. Окрім того, Java працює швидше і, як було зазначено, дозволяє обробляти велику кількість запитів водночас, що робить її більш підходящою для розроблення вебзастосунку з потенційно великою кількістю користувачів у перспективі. Тож мовою програмування бекенду обрано Java.

Задля спрощення та прискорення написання коду доцільно обрати відповідний фреймворк. Розглянемо фреймворки для Java. До популярних Java-фреймворків можна віднести Spring Framework, Hibernate, GWT та Jakarta Faces [19]. В поточному контексті останні два не є актуальними, позаяк це фреймворки для користувацького інтерфейсу. Тоді розглянемо Sptring Framework та Hibernate.

Spring Framework відомий своєю модульністю, масштабованістю та підтримкою різних архітектурних стилів, таких як MVC і REST [19].

Однією з ключових переваг використання Spring Framework є його модульність. Розробник може обирати потрібні компоненти та використовувати їх окремо або в поєднанні з іншими модулями. Це може допомогти зменшити складність коду та підвищити загальну продуктивність.

Spring також відомий своєю масштабованістю та підтримкою різних архітектурних стилів. Його можна використовувати для створення вебзастосунків за допомогою архітектури «Model-View-Controller (MVC)», а також RESTful API за допомогою фреймворку Spring Web MVC [19].

Ще однією перевагою використання Spring Framework є його широка екосистема інструментів і бібліотек. Spring містить багато вбудованих функцій, таких як ін’єкція залежностей і аспектно-орієнтоване програмування, які можуть допомогти покращити якість коду та скоротити час розробки. Крім того, для Spring доступно багато бібліотек і розширень, які можуть допомогти розробникам швидко й легко додавати функціональні можливості до своїх програм [19].

Однак, одним потенційним недоліком використання Spring Framework є його складність опанування.

Загалом, Spring Framework – це потужний і універсальний фреймворк, який може бути гарним вибором для створення вебзастосунків корпоративного рівня на основі Java. Він пропонує модульність,

масштабованість і підтримку різних архітектурних стилів, а також широку екосистему інструментів і бібліотек.

Тепер розглянемо Hibernate [18, 19]. Hibernate – популярний і широко використовуваний фреймоворк об’єктно-реляційного відображення (ORM) для програм на основі Java. Він відомий своєю простотою, ефективністю та легкістю використання.

Однією з ключових переваг використання Hibernate є його простота. Hibernate абстрагує низькорівневі деталі доступу та управління базами даних, полегшуючи розробникам взаємодію з ними за допомогою простих об’єктів Java. Це може допомогти зменшити складність коду та підвищити загальну продуктивність.

Hibernate також відомий своєю ефективністю. Він використовує різноманітні методи оптимізації, такі як відкладене завантаження та кешування, щоб зменшити кількість запитів до бази даних і підвищити загальну продуктивність. Крім того, Hibernate містить багато вбудованих функцій, таких як підтримка транзакцій і контроль паралельності, які можуть допомогти покращити цілісність і узгодженість бази даних. Ще однією перевагою використання Hibernate є простота використання. Hibernate містить багато вбудованих функцій і інструментів, таких як Hibernate Query Language (HQL) і Hibernate Criteria API, які можуть допомогти розробникам швидко та легко писати запити та взаємодіяти з базами даних.

Утім Hibernate може працювати повільно зі складними запитами та великою кількістю даних. Окрім того, він потребує конфігурації та додаткових ресурсів і складний у відлагодженні коду.

Загалом, Hibernate – це потужна та ефективна структура ORM, яка може бути гарним вибором для застосунків на основі Java, яким потрібен простий і легкий у використанні доступ до бази даних [18, 19].

Отже, Spring Framework та Hibernate не є взаємовиключними, а доповнюють одне одного. Втім Hibernate може бути корисніший при використанні складної структури бази даних, і спеціалізується конкретно на роботі зі сховищами. Тоді як Spring Framework має більш загальне спрямування та полегшує й пришвидшує написання коду в цілому. Він також має вбудовану систему для конфігурації безпеки, що значно спрощує її забезпечення. Окрім того було вирішено використовувати бібліотеку Project Lombok, позаяк вона суттєво зменшує кількість повторюваного коду в класах заміняючи його на прості анотації [18, 19].

Тепер розглянемо мови програмування для фронтенду. Безумовно найпопулярнішими мовами з великим відривом тут є JavaScript та TypeScript [16]. Безперечно є інші варіанти (наприклад, Java з використанням GWT), втім вони значно менше застосовуються, тому про них менше інформації в мережі, і, окрім того, вони складні для опанування. Тому в даному випадку вибір мови зводиться до вибору між JavaScript та TypeScript [16].

Загалом TypeScript є надбудовою над JavaScript і компілюється в нього. Однією з головних відмінностей між TypeScript і JavaScript є те, що TypeScript є статично типізованою мовою, тобто змінні мають бути оголошені з певним типом і не можуть змінювати тип під час виконання. Це може допомогти виявити помилки на ранніх етапах процесу розроблення та зробити код більш передбачуваним.

TypeScript також пропонує інші функції, яких немає в JavaScript, наприклад інтерфейси, класи та модулі. Ці функції полегшують організацію та структурування коду.

Однак, оскільки TypeScript є надбудовою над JavaScript, на TypeScript також можна писати динамічно типізований код.

Загалом, основна відмінність між TypeScript і JavaScript полягає в тому, що TypeScript додає до мови необов’язкову статичну типізацію та інші

функції, щоб зробити її більш масштабованою, зручною для обслуговування та надійною мовою. У зв’язку з цим було обрано само TypeScript [16].

Тепер оберемо фреймворк [20-23]. До популярних можна віднести Angular, React та Vue. React є бібліотекою, а не фреймворком, проте з огляду на його можливості доречніше його порівнювати саме з фреймворками.

Почнемо з Angular [20, 23]. Angular надає багатий набір інструментів і функцій, які можуть допомогти розробникам створювати складні та динамічні вебзастосунки швидко й ефективно. Деякі з цих функцій включають двостороннє зв’язування даних (two-way binding), ін’єкції залежностей і багаторазові компоненти, які можуть допомогти покращити організацію коду та скоротити час розробки.

Ще однією перевагою використання Angular є його масштабованість. Angular надає чітко визначену структуру та архітектуру, які можуть допомогти забезпечити підтримку та масштабованість вебзастосунків. Це може бути особливо важливо для великих або складних вебзастосунків, які можуть потребувати частих оновлень або внесенням змін.

Angular також відомий своєю гнучкістю. Його можна використовувати для створення широкого діапазону вебзастосунків, від простих односторінкових до великомасштабних корпоративних.

До недоліків можна віднести складність вивчення, а також суттєві відмінності між версіями AngularJS та Angular (однак, це не впливає на написання нових застосунків) [20].

Тепер розглянемо React [21]. Однією з ключових переваг використання React є його простота. React надає просту та зрозумілу модель програмування, яка дозволяє розробникам створювати складні інтерфейси користувача швидко та ефективно. Фреймворк базується на моделі декларативного програмування, яка дозволяє розробникам описувати, як має виглядати інтерфейс користувача на основі поточного стану програми.

Так само як і Angular, React досить масштабовний та гнучкий, отже з часом застосунки зручно розширювати, і писати на ньому можна застосунки різного рівня.

Однак, один потенційний недолік використання React полягає в тому, що він зосереджений на створенні інтерфейсів користувача і не надає повного рішення для створення вебзастосунків. Це означає, що розробникам може знадобитися використовувати додаткові бібліотеки або фреймворки для розроблення застосунків [21].

Перейдемо до Vue [22]. За аналогією з React, Vue досить простий і гнучкий. Основна бібліотека Vue відносно невелика, тому розробникам легко її вивчати та використовувати. Крім того, синтаксис Vue простий для розуміння, що робить його доступним як для досвідчених, так і для початківців-розробників. Він також добре інтегрується з іншими бібліотеками та фреймворками, такими як Vuex для керування станом і Vue Router для маршрутизації.

Vue також відомий своєю продуктивністю. Віртуальна система DOM Vue дозволяє ефективно оновлювати інтерфейс користувача без необхідності повторного відтворення всього DOM. Це може підвищити загальну продуктивність вебзастосунків і прискорити відповіді користувачам.

Недоліком використання Vue є те, що він не має такої великої спільноти чи екосистеми як деякі інші фреймворки JavaScript, такі як React або Angular. І, основна частина спільноті це китайці, тому знайти інформацію англійською чи українською досить складно. Це означає, що розробникам, які працюють із Vue, може бути важче знайти підтримку чи інструменти та бібліотеки сторонніх розробників, яких також менше ніж у згаданих аналогів [22].

Крім того, хоча простота Vue може бути перевагою, вона також може обмежити його функціональність для більших або складних вебзастосунків.

Іноді може доводитися покладатися на додаткові бібліотеки або фреймворки для створення повноцінної програми.

Таким чином, з огляду на кількість матеріалів, чітку структурованість, а також самодостатність в якості фреймворку було обрано Angular. Також в Angular досить часто використовується бібліотека RxJS задля можливості використовувати Observable при асинхронному програмування замість Promise. Тому й тут було прийнято рішення щодо її застосування. Observable має такі переваги над Promise:

- «Ліниве» виконання: Observable виконуються лише тоді, коли в них є підписники. Це може бути особливо корисним у сценаріях, коли потрібно виконати велику кількість асинхронних операцій, але не всі вони потрібні негайно. Тоді як у Promise вони виконуються негайно після їх створення, незалежно від того, потрібні вони чи ні;
- кілька значень: Observable можуть повертати кілька значень з часом, тоді як Promise – лише одне. Це робить Observable більш придатними для сценаріїв, де задіяні безперервні потоки даних, наприклад програми які виконуються в реальному часі або канали даних;
- скасування: Observable можна скасувати відписавшись, а Promises – ні. Це означає, що якщо запит більше не потрібен, його можна скасувати до завершення, що може заощадити ресурси та підвищити продуктивність;
- оператори: Observable має широкий набір вбудованих операторів, які можна використовувати для трансформації, фільтрації або маніпулювання даними, що надходять. Promise не має вбудованих операторів і потребує написання додаткового коду для досягнення тієї ж функціональності;
- оброблення помилок: Observable мають більш розширені можливості оброблення помилок, ніж Promise. Observable може

видавати помилки в будь-який момент протягом свого життєвого циклу, тоді як Promise може повертати лише одну помилку [23].

Наостанок розглянемо СУБД [24-26]. В якості різновиду БД обрано реляційні бази даних, оскільки сама база не потребує таких рівнів масштабованості, як, наприклад, соціальні мережі, які використовують графові БД. Окрім того, для даного застосунку підходить структурованість, тому можна відкинути документні БД і скористатися реляційною.

До популярних реляційних СУБД можна віднести PostgreSQL, Oracle та MySQL [24-26]. PostgreSQL може обробляти великі обсяги даних. Вона підтримує розширені методи індексування, такі як B-дерево, хеш і GiST, які дозволяють ефективно надсилати запити до великих наборів даних. Крім того, вона підтримує широкий спектр типів даних, включаючи масиви, JSON та XML, що робить її придатною для оброблення складних структур даних [24].

PostgreSQL також відома своїми функціями що забезпечують надійність та цілісність даних. Вона має вбудовану підтримку транзакцій і можливість накладати обмеження на дані для забезпечення узгодженості. Також включає такі функції як відновлення на певний момент часу, що дозволяє відновити дані у випадку збою [24]. Ще однією перевагою PostgreSQL є її гнучкість. Її можна розширити за допомогою спеціальних функцій, збережених процедур і тригерів, що дозволяє виконувати складну логіку на стороні бази даних. Окрім того PostgreSQL має досить велику спільноту та відкритий код, що забезпечує суттєву підтримку спільноти.

Втім, вона може бути дещо повільна та об’ємна в деяких випадках порівняно з іншими СУБД і не має такого рівня комерційної підтримки [24].

Розглянемо Oracle [25]. Загалом переваги та недоліки досить подібні до PostgreSQL, проте є свої особливості.

Oracle включає низку розширених функцій, включаючи розширений захист і шифрування, стиснення даних і паралельне виконання запитів. Ці

функції роблять її популярним вибором для організацій, яким потрібні розширені можливості управління даними.

Суттєвим недоліком Oracle є ліцензування та структура витрат. Ліцензування Oracle може бути дорогим, а вартість поточного обслуговування та підтримки з часом може збільшитися. Крім того, складність може зробити її більш складною для вивчення та використання, ніж деякі інші системи баз даних [25].

Наостанок розглянемо MySQL [26]. Однією з головних переваг MySQL є простота використання та встановлення. Вона має простий процес інсталяції, а її зручний інтерфейс спрощує налаштування та конфігурацію.

Ще однією перевагою MySQL є її продуктивність. Вона оптимізована для швидкої роботи та може ефективно обробляти високошвидкісні дані та транзакції. Містить функції кешування та оптимізації запитів, які ще більше поліпшують продуктивність. MySQL також має відкритий вихідний код та активну спільноту як і PostgreSQL.

Однак, у MySQL обмежена підтримка розширених функцій. Хоча вона включає підтримку багатьох розширених функцій, таких як збережені процедури та тригери. Вона не має такого ж рівня функціональності як деякі інші системи баз даних такі, як Oracle або PostgreSQL. Крім того, її функції безпеки можуть бути не такими надійними, як у деяких інших систем баз даних, що може зробити її більш вразливим до атак. Також, MySQL погано масштабується й не дуже добре працює з великою кількістю даних [26].

З огляду на відкритий вихідний код та підтримку спільноти, кращу масштабованість, а також розширені функції в якості СУБД обрано PostgreSQL.

Отже, для розроблення вебзастосунку формування моделі загроз безпеці мобільних пристроїв обрано мову програмування Java, бібліотеку Project Lombok та фреймворк Spring Framework. Тоді як бекенду, мову

програмування JavaScript, фреймворк Angular та бібліотеку RxJS для фронтенду, а також СУБД PostgreSQL.

4.2. Тестування вебзастосунку формування моделі загроз безпеці мобільних пристроїв

Використання обраних засобів програмування дозволило розробити робочий проєкт вебзастосунку. Наступник кроком протестуємо його. Для цього використаємо димне тестування. Воно передбачає застосування тестів, якими перевіряється базова функціональність вебзастосунку та коректність його роботи [27]. Розглянемо такі сценарії тестування:

1. Введення електронної пошти в некоректному форматі при авторизації.

   Послідовність дій:

   1) увійти на сторінку авторизації;
   2) ввести електронну пошту, яка не відповідає формату електронної пошти.

   Очікуваний результат: поле електронної пошти підсвічується червоним, виводиться відповідна помилка (рис. 4.1).

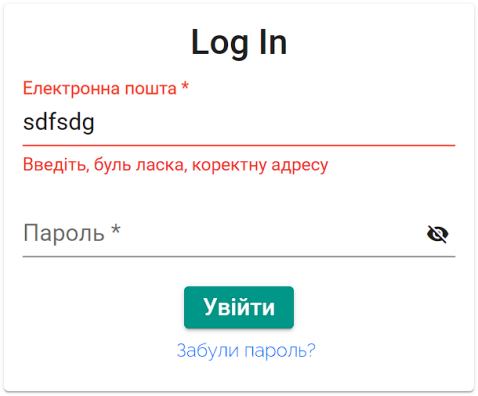


Рис. 4.1. Введення електронної пошти некоректного формату на сторінці авторизації

2. Спроба входу з некоректними даними облікового запису.

Послідовність дій:

1) увійти на сторінку авторизації;
2) ввести електронну пошту та пароль такі, що разом не відповідають ніякому обліковому запису;
3) натиснути кнопку входу.

Очікуваний результат: виведення повідомлення про некоректний логін чи пароль на червоному фоні (рис. 4.2).

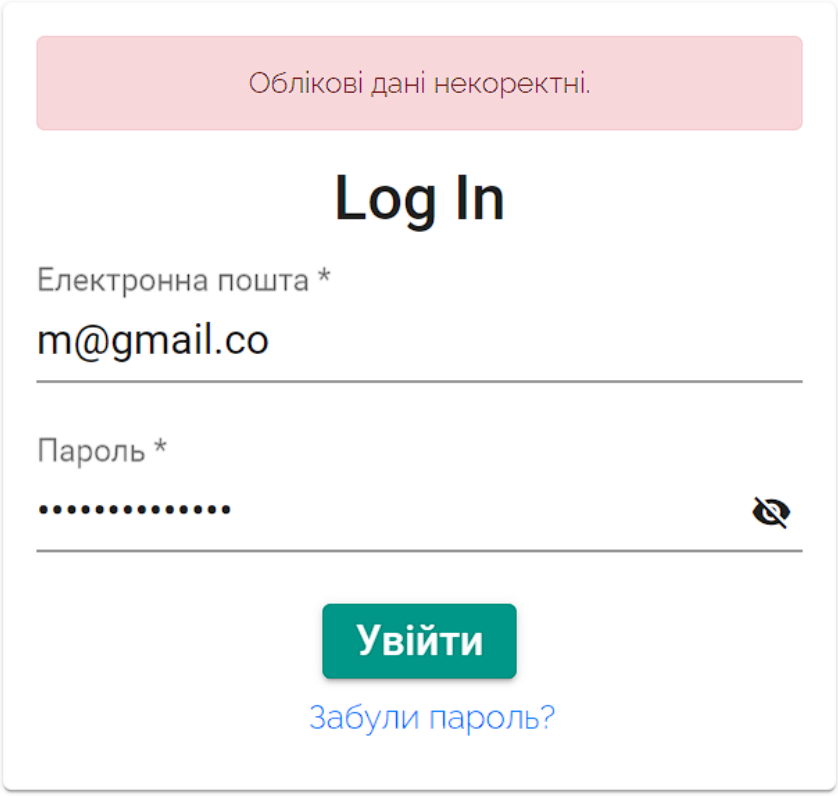


Рис. 4.2. Спроба входу з некоректними даними облікового запису

3. Спроба реєстрації без введення даних.

Послідовність дій:

1) увійти на сторінку реєстрації;
2) відразу натиснути кнопку реєстрації.

Очікуваний результат: поля підсвічуються червоним та під кожним виводиться помилка про відсутність інформації (рис. 4.3).

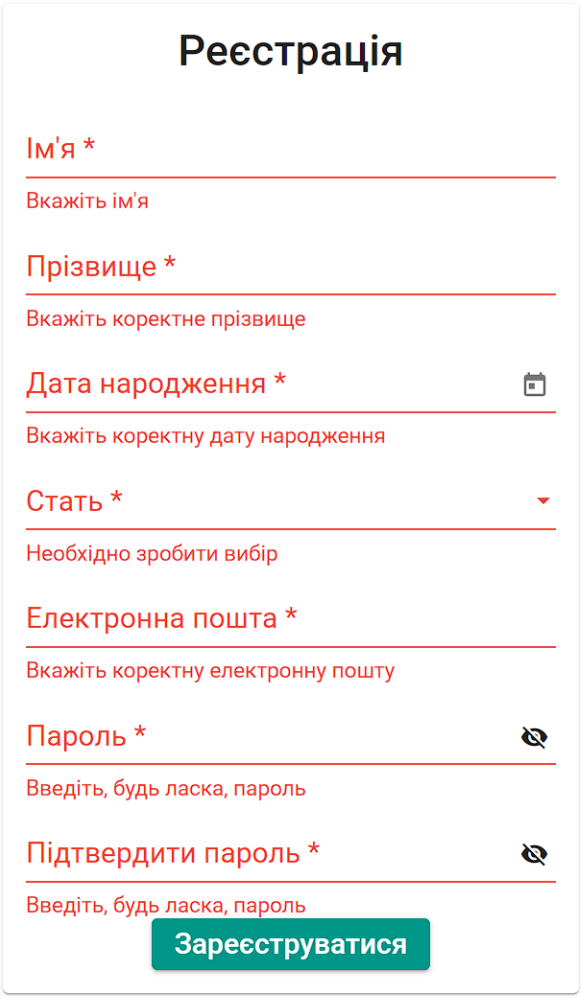


Рис. 4.3. Спроба реєстрації без введення даних

4. Введення некоректних даних для реєстрації.

Послідовність дій:

1) увійти на сторінку реєстрації;
2) заповнити некоректними даними будь-які поля з наступних: ім’я, прізвище, електронна пошта, пароль;
3) натиснути кнопку реєстрації.

Очікуваний результат: поля підсвічуються червоним та під кожним виводиться помилка про необхідність введення коректних даних (рис. 4.4).

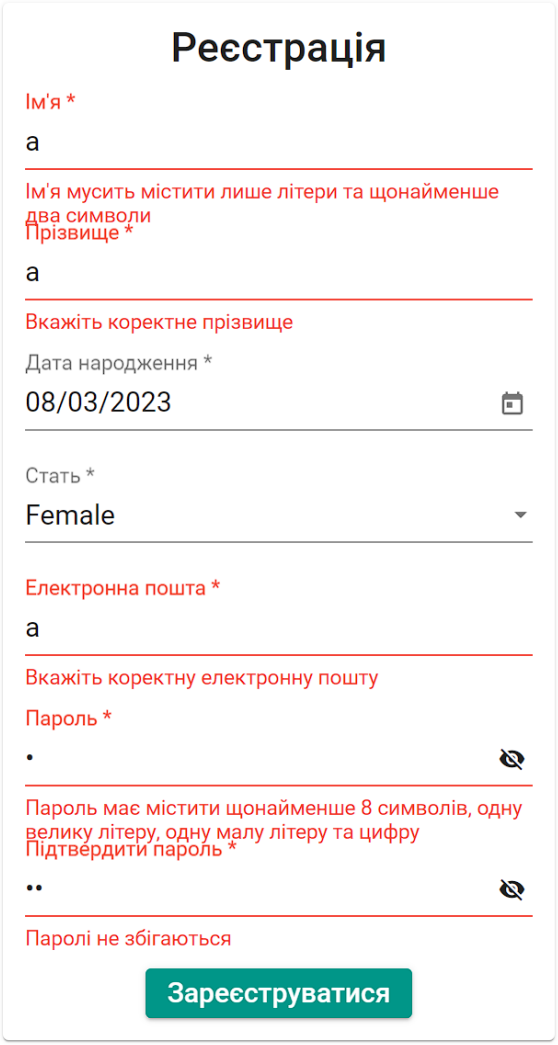


Рис. 4.4. Спроба реєстрації з некоректними даними

5. Спроба зміни імені та прізвища на такі які не відповідають формату.

   Послідовність дій:

   1) авторизуватися;

   2) Увійти на сторінку редагування профілю;

3) Ввести в поле імені та/або прізвища значення довжиною менш ніж 3 та/або з символами які не є літерами.

Очікуваний результат: поля підсвічуються червоним та виводяться помилки про довжину та символи (рис. 4.5).

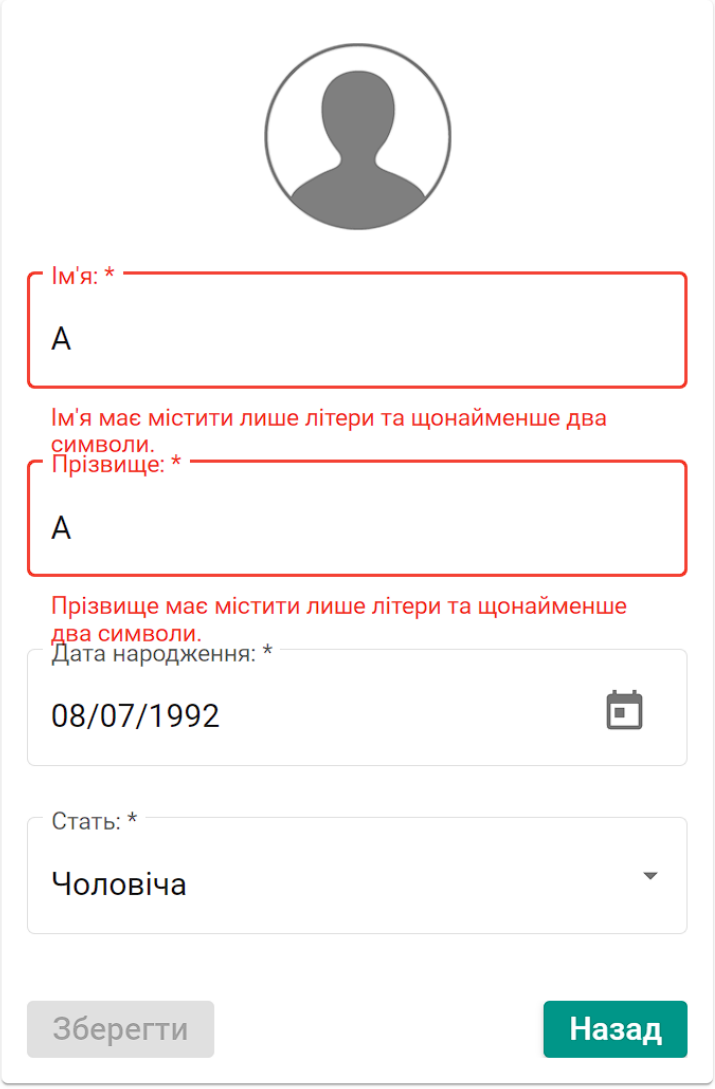


Рис. 4.5. Спроба зміни імені та прізвища на некоректний формат

6. Спроба зміни паролю з різними паролями в полях нового паролю.

Послідовність дій:

1) авторизуватися;

2) увійти на сторінку зміни паролю;

3) ввести такі паролі в поля нового паролю та повторення нового паролю, які не збігаються.

Очікуваний результат: поле повтору паролю підсвічується червоним та виводиться відповідна помилка (рис. 4.6).

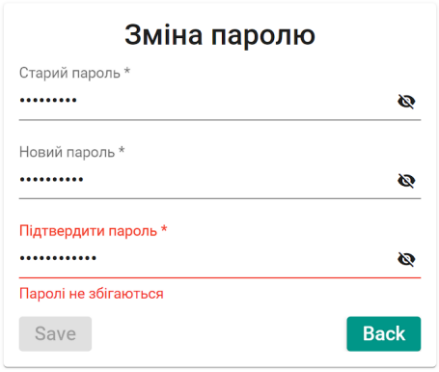


Рис. 4.6. Спроба зміни паролю з різними паролями в полях нового паролю

7. Спроба зміни паролю з некоректним старим паролем.

Послідовність дій:

1) авторизуватися;

2) увійти на сторінку зміни паролю;

3) ввести некоректний старий пароль;

4) натиснути кнопку збереження.

Очікуваний результат: виводиться помилка на червоному фоні яка повідомляє про некоректно введений старий пароль (рис. 4.7).

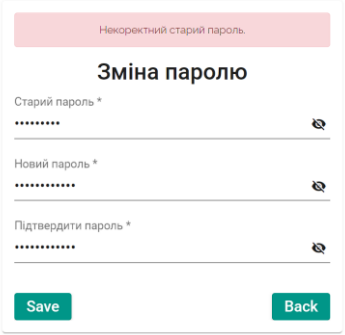


Рис. 4.7. Спроба зміни паролю з некоректним старим паролем

8. Спроба додавання чи редагування пристрою з некоректними даними.

Послідовність дій:

1) авторизуватися;

2) увійти на сторінку пристроїв;

3) натиснути на кнопку додавання чи редагування;

4) ввести некоректні дані.

Очікуваний результат: виводяться помилки під полями вводу (рис. 4.8, 4.9).

Рис. 4.8. Спроба додавання пристрою з некоректними даними

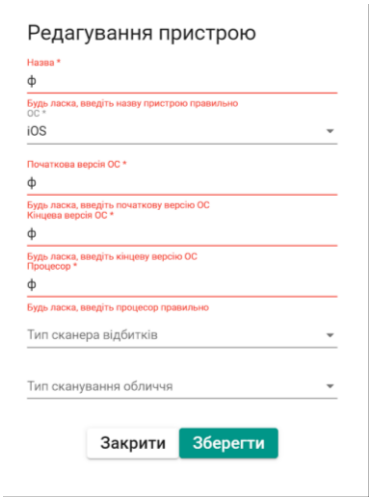


Рис. 4.9. Спроба редагування пристрою з некоректними даними

9. Спроба додавання чи редагування модератора з некоректними даними.

   Послідовність дій:

   1) авторизуватися;

   2) увійти на сторінку модераторів;

   3) натиснути на кнопку додавання чи редагування;

   4) ввести некоректні дані.

   Очікуваний результат: виводяться помилки під полями вводу (рис. 4.10, 4.11).

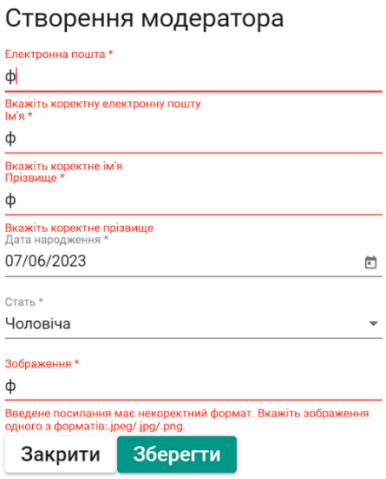


Рис. 4.10. Спроба додавання модератора з некоректними даними

Рис. 4.11. Спроба редагування модератора з некоректними даними

10. Спроба формування моделі без введення пристрою.

    Послідовність дій:

    1) авторизуватися;

    2) увійти на сторінку формування моделі;

    3) відразу натиснути кнопку формування.

    Очікуваний результат: поле вводу підсвічується червоним (рис. 4.12).

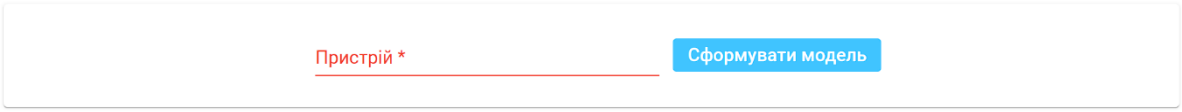


Рис. 4.12. Спроба формування моделі без введення пристрою

11. Спроба зміни назви чек-листа на порожню або надто довгу.

    Послідовність дій:

    1) авторизуватися;

    2) увійти на сторінку чек-листів;

    3) натиснути на кнопку редагування;

    4) прибрати назву або написати надто довгу.

    Очікуваний результат: поле вводу підсвічується червоним та виводиться помилка (рис. 4.13, 4.14).

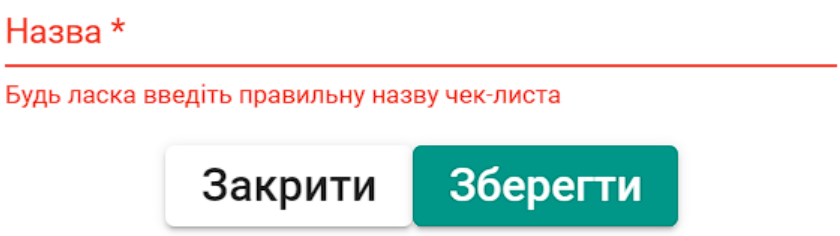


Рис. 4.13. Спроба зміни назви чек-листа на порожню

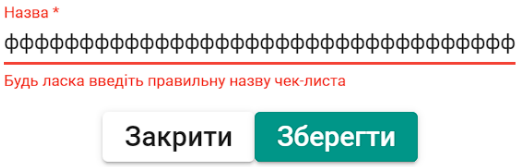


Рис. 4.14. Спроба зміни назви чек-листа на надто довгу

12. Спроба додавання чи редагування загрози з некоректними даними.

Послідовність дій:

1) авторизуватися;

2) увійти на сторінку загроз;

3) натиснути на кнопку редагування чи додавання;

4) ввести некоректні дані.

Очікуваний результат: поля вводу підсвічуються червоним та виводяться помилки (рис. 4.15, 4.16).

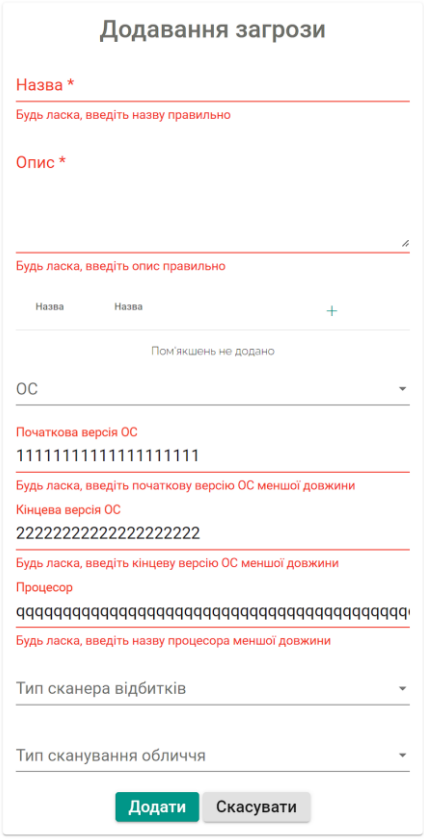


Рис. 4.15. Спроба додавання загрози з некоректними даними

13. Спроба додавання чи редагування пом’якшення з некоректними даними.

    Послідовність дій:

    1) авторизуватися;

    2) увійти на сторінку пом’якшень;

    3) натиснути на кнопку редагування чи додавання;

    4) ввести некоректні дані.

    Очікуваний результат: поля вводу підсвічуються червоним та виводяться помилки (рис. 4.17, 4.18).

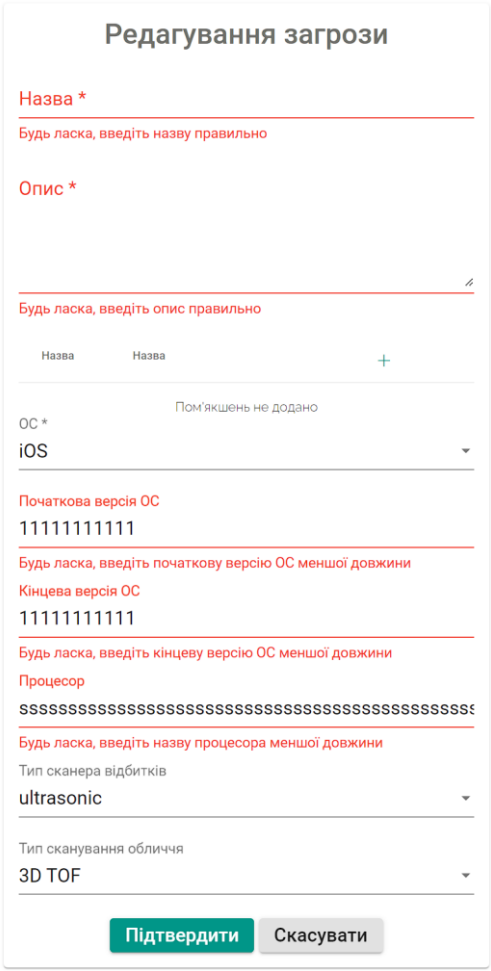


Рис. 4.16. Спроба редагування загрози з некоректними даними

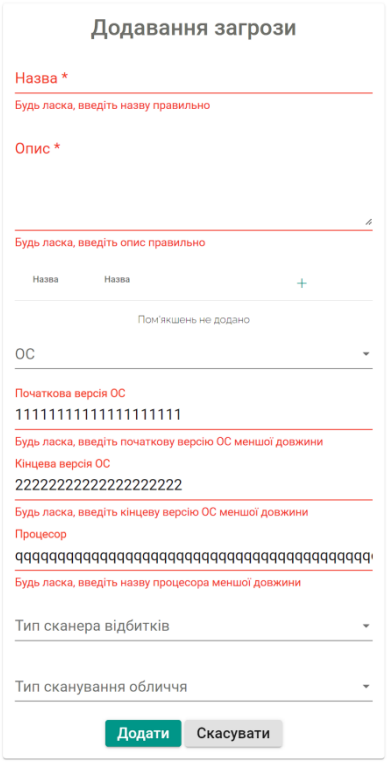


Рис. 4.17. Спроба додавання пом’якшення з некоректними даними

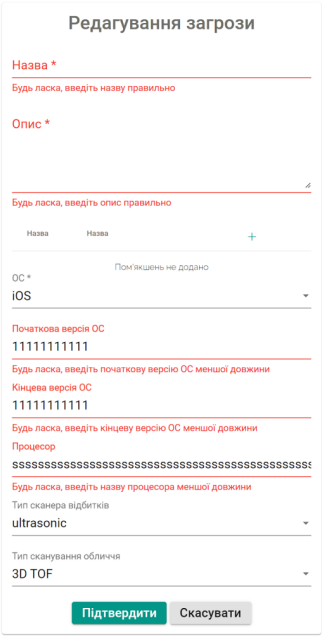


Рис. 4.18. Спроба редагування пом’якшення з некоректними даними

Отже, вебзастосунок протестовано, і, як наслідок, встановлено, що він достатньо стійкий та придатний до використання.

4.3. Демонстрація вебзастосунку формування моделі загроз безпеці мобільних пристроїв

Після відлагодження та тестування отримаємо робочий проєкт вебзастосунку. В подальшому він може розширюватися та змінюватися, втім у межах даної роботи це завершене програмне забезпечення, яке відповідає визначеним на початку функціональним та нефункціональним вимогам. Відтак продемонструємо використання вебзастосунку для формування моделі загроз безпеці мобільних пристроїв.

Коли користувач уперше відкриває вебзастосунок, перше, що він бачить – це сторінка авторизації (рис. 4.19). На цій сторінці потрібно ввести логін та пароль, окрім того можна перейти до відновлення паролю чи до реєстрації.

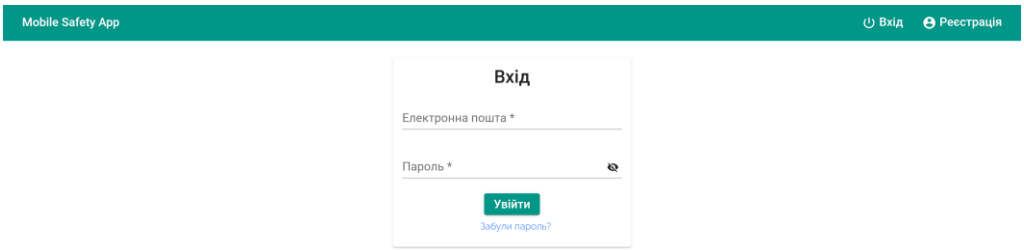


Рис. 4.19. Сторінка авторизації

Для поля вводу паролю маємо функцію відображення в двох варіантах – прихованому (рис. 4.20) та відкритому (рис. 4.21).

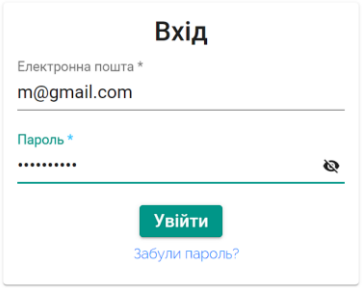


Рис. 4.20. Відображення паролю у прихованому стані

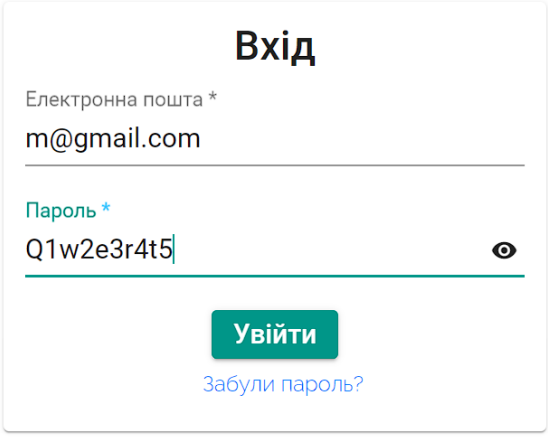


Рис. 4.21. Відображення паролю у відкритому стані

У випадку 5-кратного неправильного вводу даних при спробі входу на годину застосовується перевірка, яка змушує користувача пройти reCAPTCHA (рис. 4.22) перш ніж увійти.

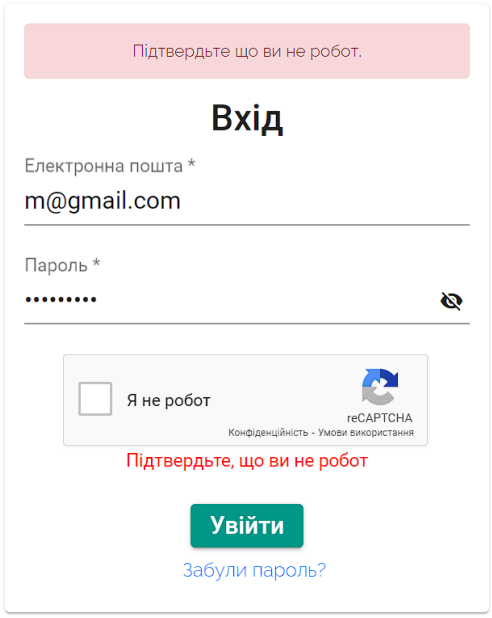


Рис. 4.22. Відображення перевірки reCAPTCHA

Якщо користувач забув пароль, то він може відновити його, перейшовши на відповідну сторінку (рис. 4.23) та увівши електронну пошту, до якої прив’язано обліковий запис (рис. 4.24).

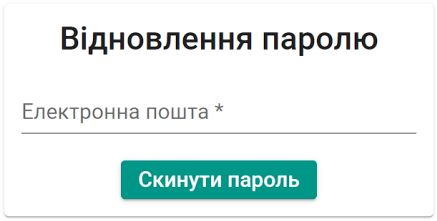


Рис. 4.23. Сторінка відновлення паролю

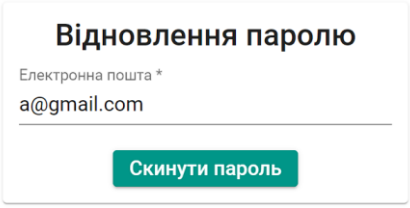


Рис. 4.24. Відновлення паролю

Шоб зареєструватися, користувачу слід перейти на сторінку реєстрації (рис. 4.25–4.27) та заповнити потрібні дані, а саме: ім’я, прізвище, дату народження, стать, електронну пошту та пароль. Обидва поля для введення пароля мають можливість відображення у відкритому форматі.

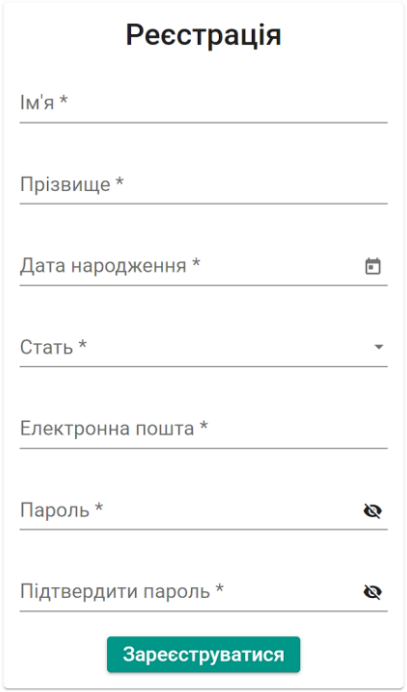


Рис. 4.25. Сторінка реєстрації

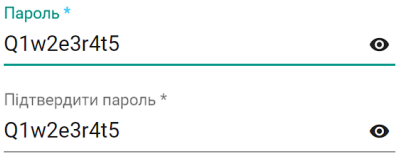


Рис. 4.26. Сторінка реєстрації з паролями у відкритому форматі

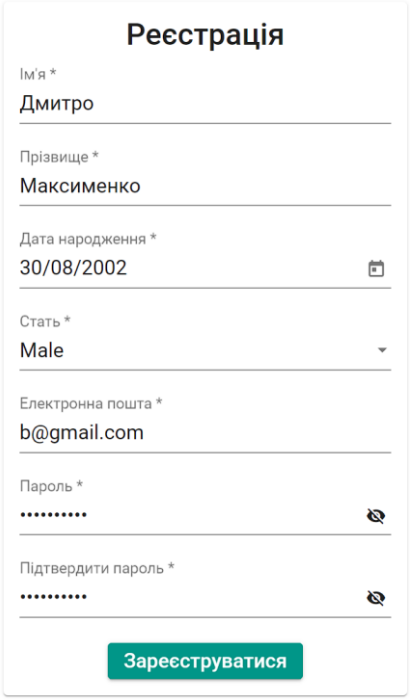


Рис. 4.27. Сторінка реєстрації з коректно заповненими даними

Після авторизації користувач переходить на сторінку власного профілю (рис. 4.28), де відображено його дані, а також є можливість перейти до його редагування, зміни паролю, або вийти з системи.

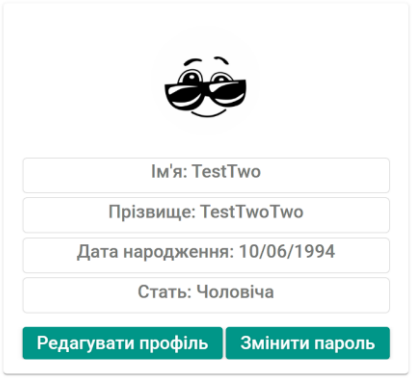


Рис. 4.28. Сторінка профілю

Залежно від ролі користувача відображається різний набір сторінок, на які можна перейти. В усіх доступні формування моделі та чек-листи, в користувача немає інших сторінок (рис. 4.29), в адміністратора додатково є керування модераторами (рис. 4.30), а в модератора додатково є керування пристроями, загрозами та пом’якшеннями (рис. 4.31).

Рис. 4.29. Панель меню для звичайного користувача

Рис. 4.30. Панель меню для модератора

Рис. 4.31. Панель меню для адміністратора

На сторінці редагування профілю (рис. 4.32, 4.33) можна змінити всі дані, корім паролю та пошти.

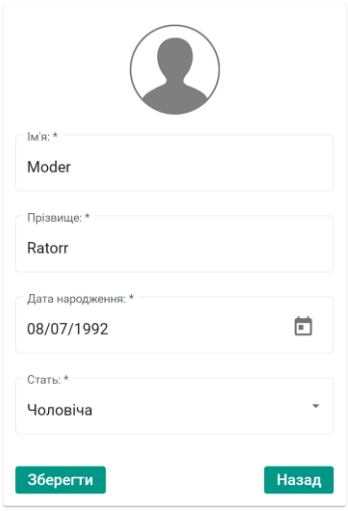


Рис. 4.32. Сторінка редагування профілю

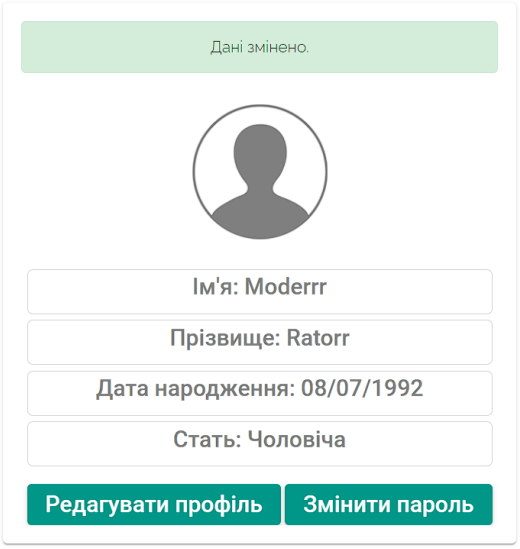


Рис. 4.33. Профіль відредаговано

На сторінці зміни паролю (рис. 4.34) слід ввести старий пароль, та двічі новий пароль.

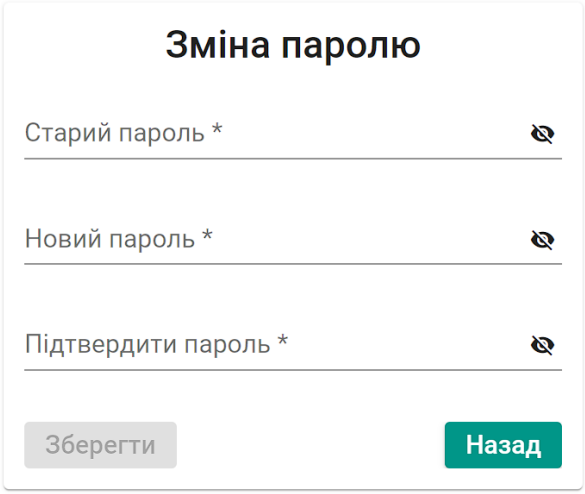


Рис. 4.34. Сторінка зміни паролю

Це необхідно (рис. 4.34), щоб користувач випадково не змінив пароль на небажаний. Всі три поля мають кнопки для переведення у відкритий формат і назад у прихований. Коли пароль змінено виводиться сповіщення (рис. 4.35).

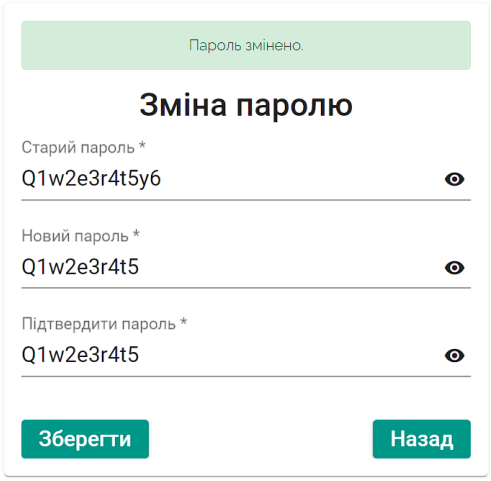


Рис. 4.35. Пароль змінено

Пристрої, загрози, пом’якшення та модераторів можна переглядати за наявності відповідної ролі у вигляді таблиці. Наприклад, так виглядає таблиця пристроїв (рис. 4.36).

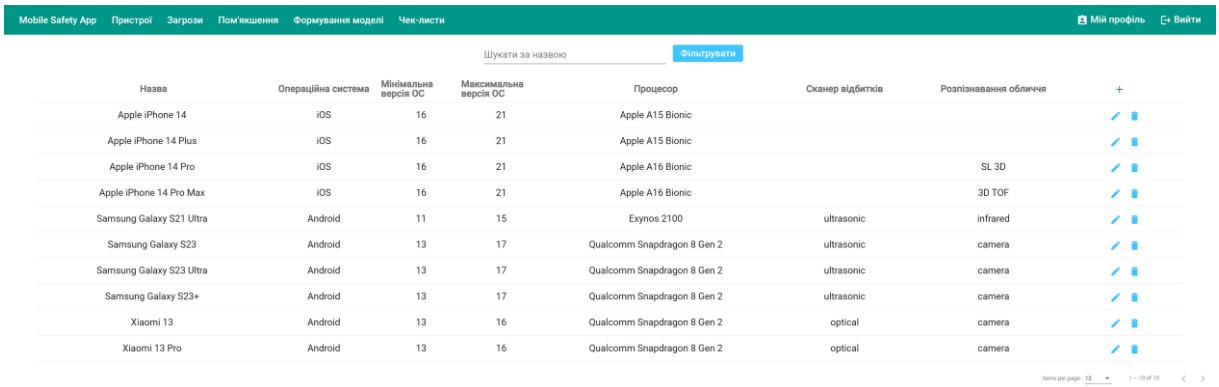


Рис. 4.36. Таблиця пристроїв

Пристрої, загрози та пом’якшення можна видаляти. При натисненні на іконку смітника з’явиться вікно підтвердження (рис. 4.37).

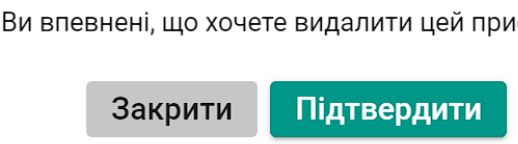


Рис. 4.37. Вікно підтвердження видалення пристроїв

Також, пристрої, загрози, пом’якшення та модераторів можна редагувати та додавати. Наприклад, так виглядає додавання загрози (рис. 4.38) та редагування пом’якшення (рис. 4.39).

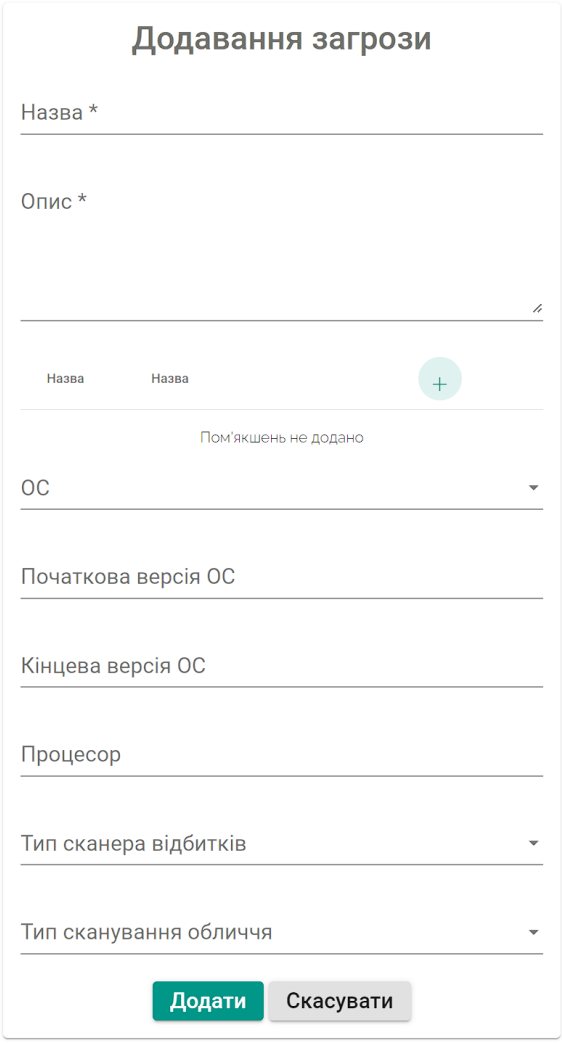


Рис. 4.38. Додавання загрози

Для формування моделі загроз слід ввести пошуковий запит, обрати пристрій з випадного списку та натиснути на кнопку формування моделі загроз (рис. 4.40). Буде виведено результат формування – таблиця з загрозами та відповідними пом’якшеннями для обраного мобільного пристрою (рис. 4.41).

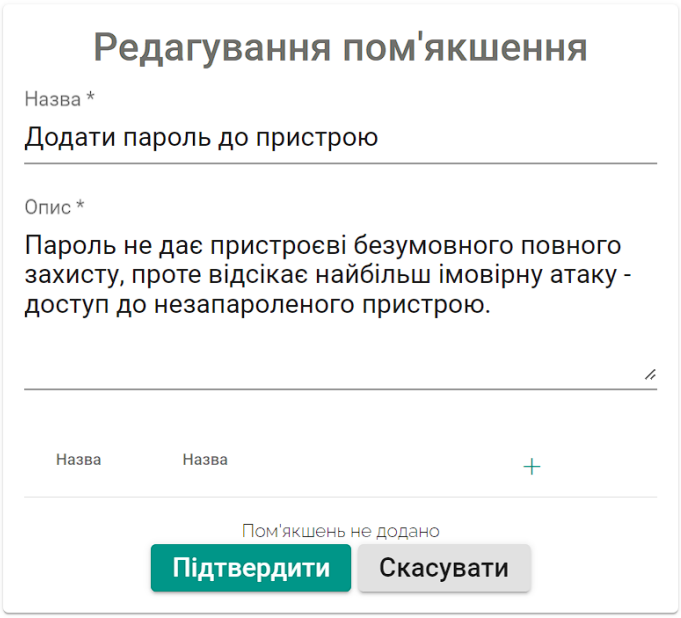


Рис. 4.39. Редагування пом’якшення

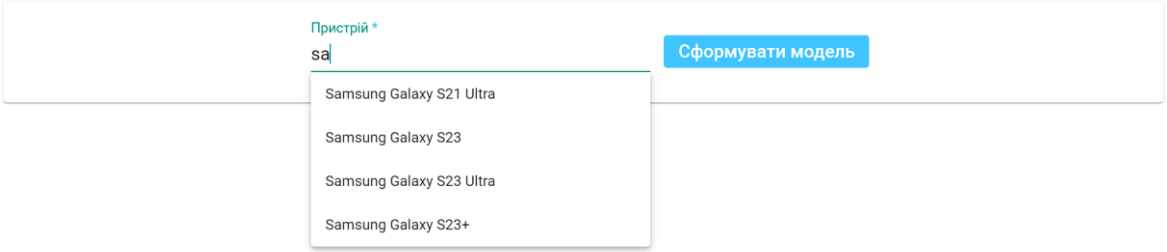


Рис. 4.40. Вибір пристрою для формування моделі

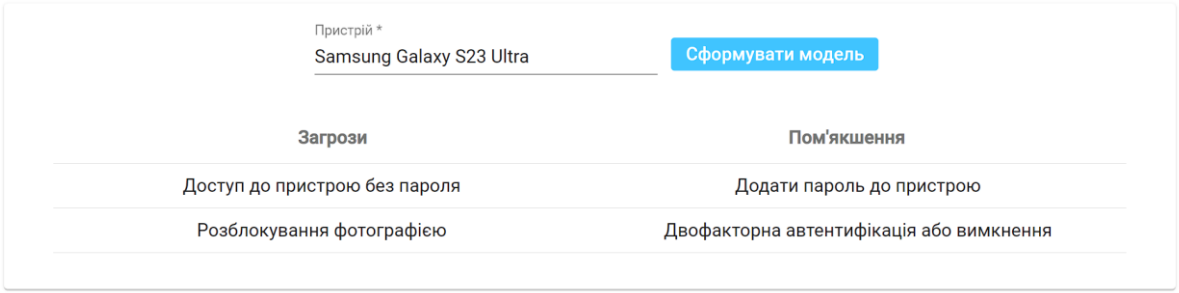


Рис. 4.41. Сформована модель (демонстраційний приклад)

При натисненні на загрозу чи пом’якшення зі сформованої моделі виведеться інформація (рис. 4.42).

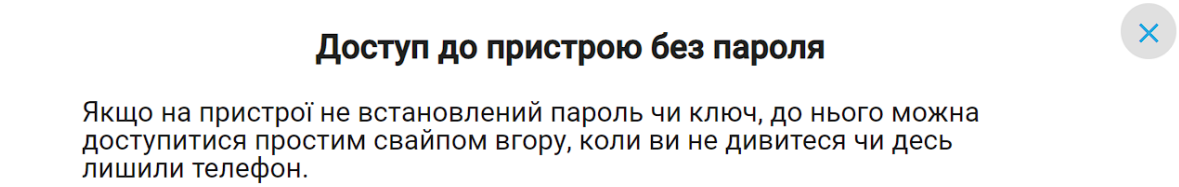


Рис. 4.42. Інформація про загрозу

На основі сформованої моделі створюється чек-лист (рис. 4.43), в якому можна ставити відмітки, унаслідок чого рухається індикатор прогресу. Цей чек-лист додається до переліку чек-листів користувача (рис. 4.44).

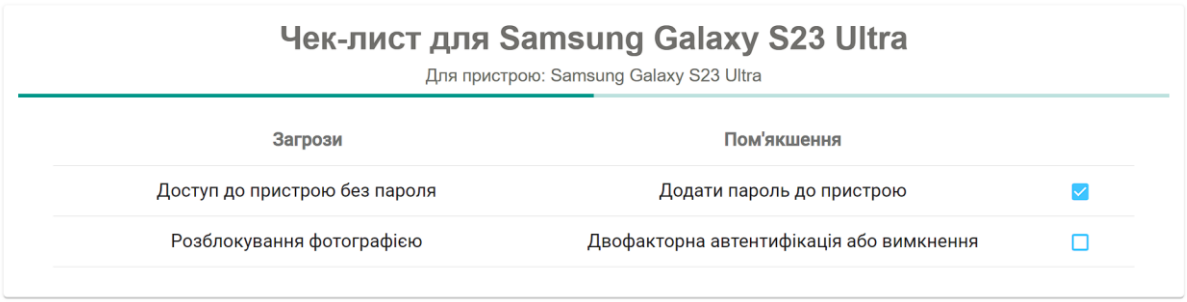


Рис. 4.43. Чек-лист (демонстраційний приклад)

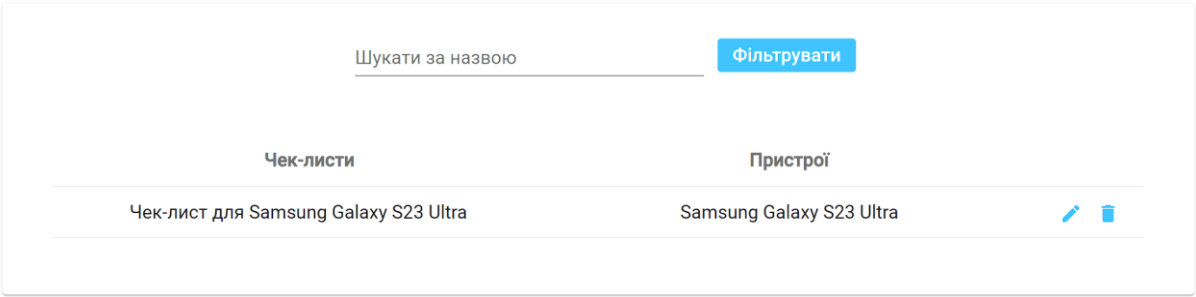


Рис. 4.44. Чек-листи користувача

Так само, як і пристрої, загрози та пом’якшення, чек-лист можна редагувати та видаляти. Редагується назва чек-листа (рис. 4.45).

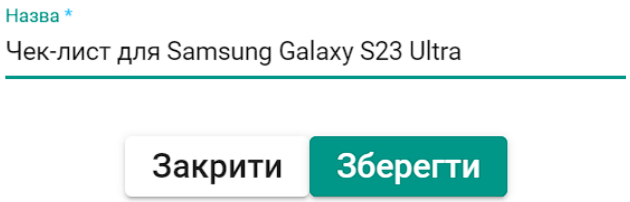


Рис. 4.45. Редагування назви чек-листа

Отже, розроблений робочий проєкт вебзастосунку продемонстровано для формування моделі загроз безпеці мобільних пристроїв.

4.4. Висновки за розділом 4

Для розроблення робочого проєкту вебзастосунку формування моделі загроз безпеці мобільних пристроїв проаналізовано, порівняно та обрано засоби програмування. В якості мови програмування для бекенду розглядалися Java, Python та JavaScript. Зрештою обрано Java, оскільки як мова бекенду вона швидша за інші, тому краще підходить для багатокористувацького вебзастосунку, а також має статичну типізацію. Це дозволяє зменшити кількість помилок про розроблення та спрощує відлагодження коду.

Серед фреймворків для Java розглядалися Spring Framework та Hibernate. Це не взаємовиключні фреймворки, проте прийнято рішення використовувати Spring Framework. Позаяк він всеохопний, а також має зручні засоби для налаштувань безпеки, тоді як Hibernate можна використати якщо в майбутньому структура сховища ускладниться. Окрім того, було вирішено використовувати бібліотеку Project Lombok, оскільки вона суттєво скорочує кількість повторюваного коду в класах.

Як мову програмування фронтенду було обрано TypeScript. Його було порівняно з JavaScript і зроблено висновок, що краще підходить TypeScript,

оскільки його особливості дозволяють структурувати програмне забезпечення, також він має статичну типізацію. Це дозволяє мінімізувати кількість помилок та спростити відлагодження.

Фреймворком для TypeScript обрано Angular, оскільки про нього є багато інформації в мережі, він дозволяє будувати чітку масштабовану структуру та є самодостатнім. Тоді як React та Vue часто потребують багато додаткових бібліотек і, окрім того, Vue має менше інформації в мережі бо спільнота не така розвинута. Разом з Angular використовується бібліотека RxJS, яку рекомендується використовувати разом з Angular.

СУБД обрано PostgeSQL, оскільки на відміну від MySQL вона ліпше масштабується та має більше функціональних можливостей, а Oracle потребує оплачуваного ліцензування.

Розроблений робочий проєкт вебзастосунку протестовано та продемонстровано, що переконало в його стабільності. До того ж відповідності специфікованим функціональним і нефункціональним вимогам.

ВИСНОВКИ

Отже, змодельовано вебзастосунок формування моделі загроз безпеці мобільних пристроїв та програмного реалізовано його. Актуальність його створення обґрунтовано тим, що нині мобільні пристрої, які являють собою міні-комп'ютери зі складною операційною системою та комунікаційними можливостями, стали невід'ємним складником життя людини. І, оскільки вони мають багато можливостей, то є предметом значного інтересу з боку зловмисників. Тому потрібно знати, які загрози можуть порушувати безпеку мобільних пристроїв і як мінімізувати пов'язані з ними ризики.

За результатами проведеного аналізу процесу формування моделі загроз безпеці мобільних пристроїв було визначено його особливості. Крім того зіставлено існуючі рішення. Як наслідок, встановлено, що вони містять недоліки та не дозволяють формувати зазначену модель. Відтак, було специфіковано функціональні та нефункціональні вимоги до вебзастосунку діаграмою вимог у графічній нотації SysML.

Для розроблення структури вебзастосунку спершу побудовано його концептуальну модель. Нею формалізовано діяльність формування моделі загроз безпеці мобільних пристроїв, процеси, потоки даних та зв'язки між ними. На основі цієї моделі конкретизовано архітектуру та послідовність і логіку дій вебзастосунку. Тож його представлено клієнтською і серверною частинами та базою даних.

З урахуванням побудованої концептуальної моделі та обраної архітектури розроблено об'єктно-орієнтовану модель вебзастосунку формування моделі загроз безпеці мобільних пристроїв у графічній нотації UML. Зокрема визначено його варіанти використання, логічну та фізичну структури. Для їх реалізування обрано відповідні засоби програмування. Розроблений робочий проєкт вебзастосунку протестовано та продемонстровано на прикладі формування моделі загроз безпеці мобільних пристроїв. І, встановлено його відповідність специфікованим вимогам.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Digital around the World. URL: https://datareportal.com/global-digital-overview (accessed on: 20.04.2023).
2. Lumsden J. Handbook of Research on User Interface Design and Evaluation for Mobile Technology. Birmingham: Aston University, 2008. 1171 p.
3. Getting started with the Threat Modeling Tool. Microsoft. 2022. URL: https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-getting-started (accessed on: 20.04.2023).
4. OWASP Mobile Application Securit. URL: https://mas.owasp.org/. (accessed on: 21.04.2023).
5. ThreatModeler. URL: https://threatmodeler.com/ threatmodeler/. (accessed on: 21.04.2023).
6. Demo. *ThreatModeler*. URL: https://go.threatmodeler.com/demo (accessed on: 22.04.2023).
7. Про забезпечення функціонування української мови як державної : Закон України від 25.04.2019 № 2704-VIII. URL: https://zakon.rada.gov.ua/ laws/show/2704-19#Text (дата звернення: 22.04.2023).
8. Lynch A. What is IDEF – Definition, Methods, and Benefits. URL: https://www.edrawsoft.com/what-is-idef.html. (accessed on: 26.04.2023).
9. Методологія IDEF3. URL: https://stud.com.ua/87186/ekonomika/ metodologiya_idef3 (дата звернення: 28.04.2023).
10. What is a Data Flow Diagram. URL: https://www.lucidchart.com/ pages/data-flow-diagram (accessed on: 29.04.2023).
11. What is Use Case Diagram? URL: https://www.visual-paradigm.com/ guide/uml-unified-modeling-language/what-is-use-case-diagram/ (accessed on: 03.05.2023).

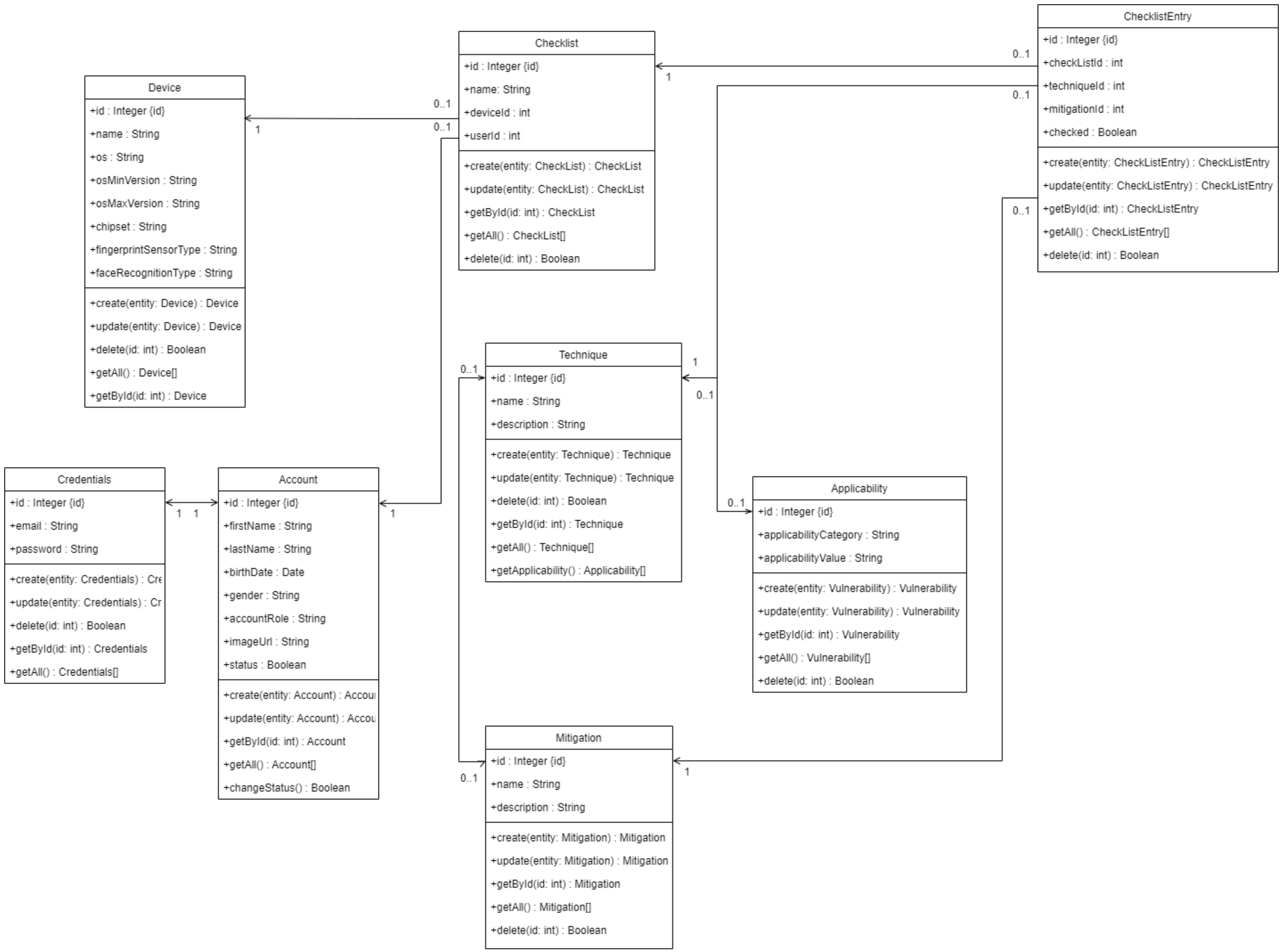
12. UML Class Diagram Tutorial. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (accessed on: 03.05.2023).
13. What is Activity Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/ (accessed on: 05.05.2023).
14. What Is Component Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/ (accessed on: 05.05.2023).
15. What is Deployment Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (accessed on: 06.05.2023).
16. Clark J. The Best 10 Backend Programming Languages. URL: https://blog.back4app.com/backend-programming-languages-list/ (accessed on: 09.05.2023).
17. Python Advantages and Disadvantages – Step in the right direction. URL: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/ (accessed on: 09.05.2023).
18. Molina G. The Pros and Cons of Java. 2021. URL: https://distantjob.com/blog/pros-cons-java-development/ (accessed on: 09.05.2023).
19. Java Spring Pros and Cons. URL: https://www.javatpoint.com/java-spring-pros-and-cons (accessed on: 10.05.2023).
20. Advantages and Disadvantages of Angular. *knowledgehut*. 2022. URL: https://www.knowledgehut.com/blog/web-development/advantages-and-disadvanta ges-of-angular (accessed on: 10.05.2023).
21. Pros and Cons of ReactJS. URL: https://www.javatpoint.com/pros-and-cons-of-react (accessed on: 10.05.2023).
22. Tobiasz F. Using Vue: Pros and Cons. URL: https://thecodest.co/blog/pros-and-cons-of-vue/ (accessed on: 10.05.2023).

23. Gowrimathi S. Angular Promises Versus. *Syncfusion*. 2023. URL: https://www.syncfusion.com/blogs/post/angular-promises-versus-observables.aspx (accessed on: 10.05.2023).
24. Learning PostgreSQL | What is PostgreSQL, Advantages, Disadvantages. *Code Topology*. 2022. URL: https://codetopology.com/database/ what-is-postgresql/ (accessed on: 12.05.2023).
25. Singh H. Oracle Database Advantages, Disadvantages and Features [Guide 2023]. *NineHertz*. 2023. URL: https://theninehertz.com/blog/advantages-of-using-oracle-database (accessed on: 12.05.2023).
26. MySQL Advantages And Disadvantages. *W3schools*. URL: https://www.w3schools.blog/mysql-advantages-disadvantages (accessed on: 12.05. 2023).
27. Pittet S. The different types of software testing. *Atlassian*. URL: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software- testing (accessed on: 12.05. 2023).

ДОДАТКИ

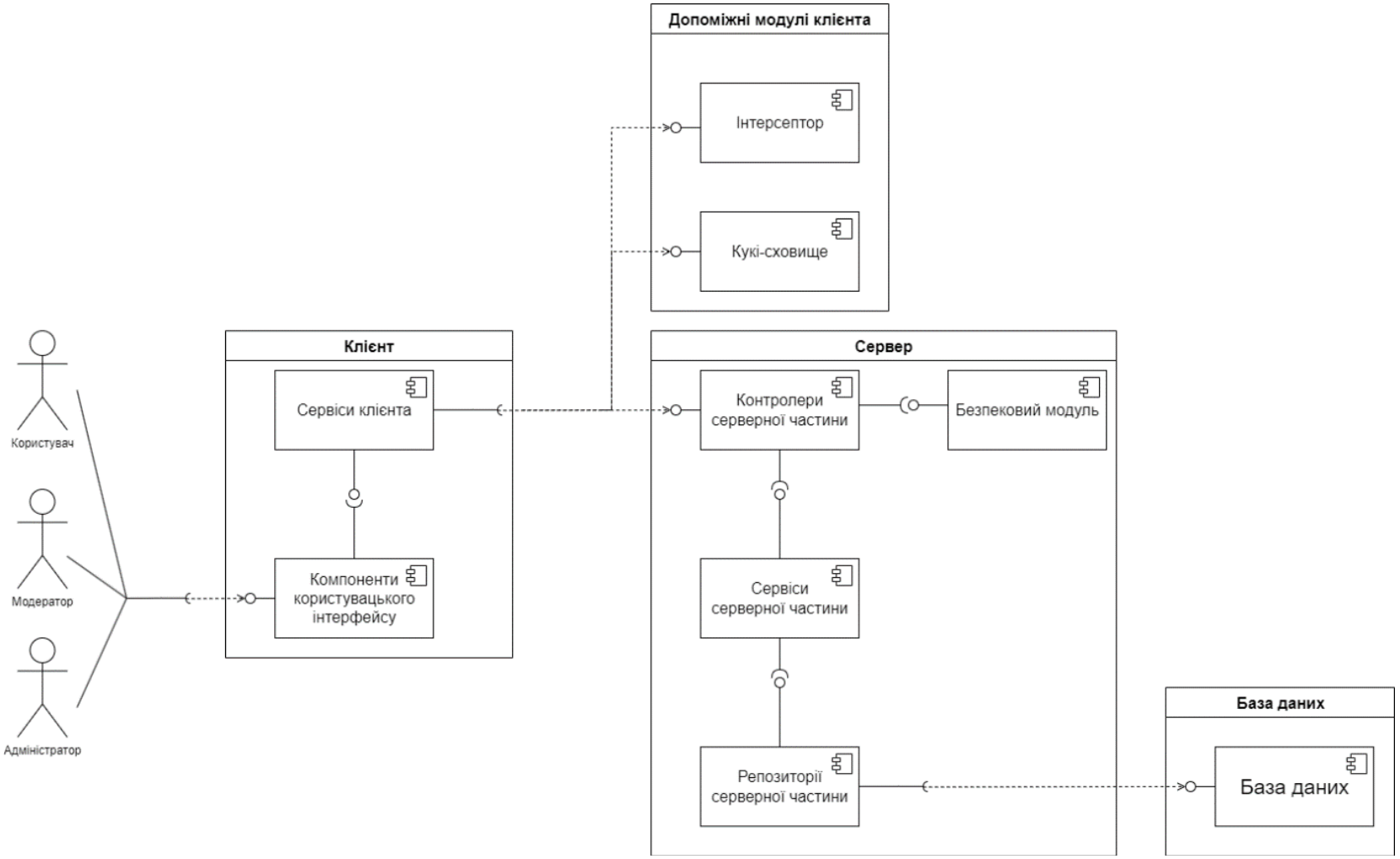
Додаток 1

Копії графічних матеріалів

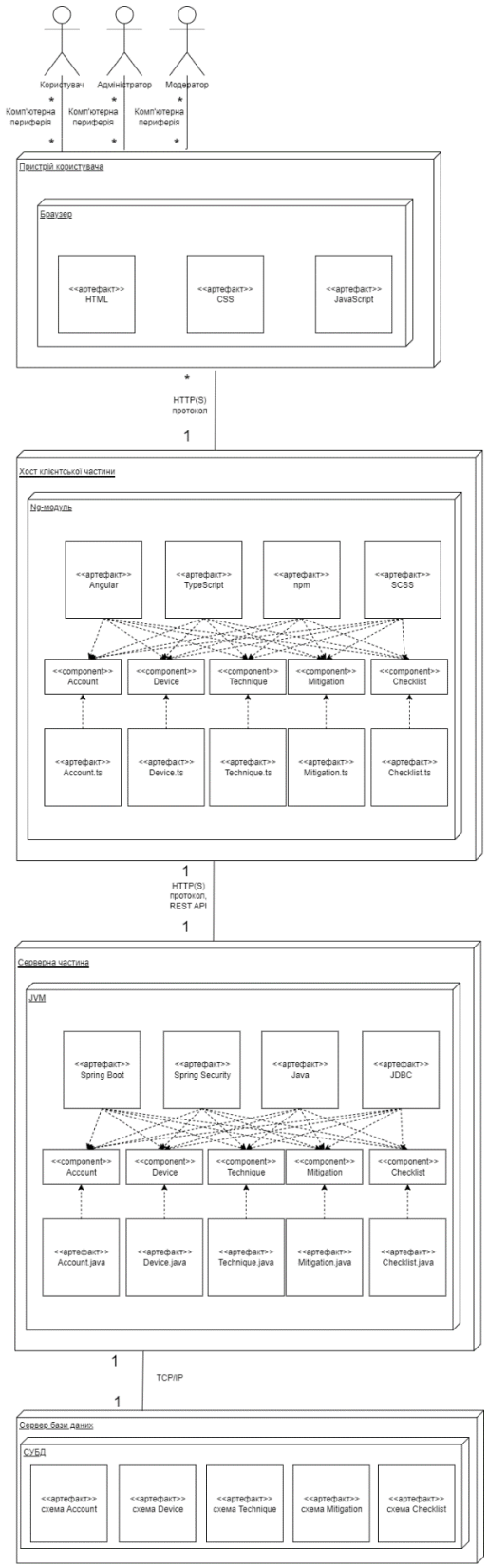


ДП.045440-06-99. Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв. Статична логічна структура вебзастосунку. Діаграма класів

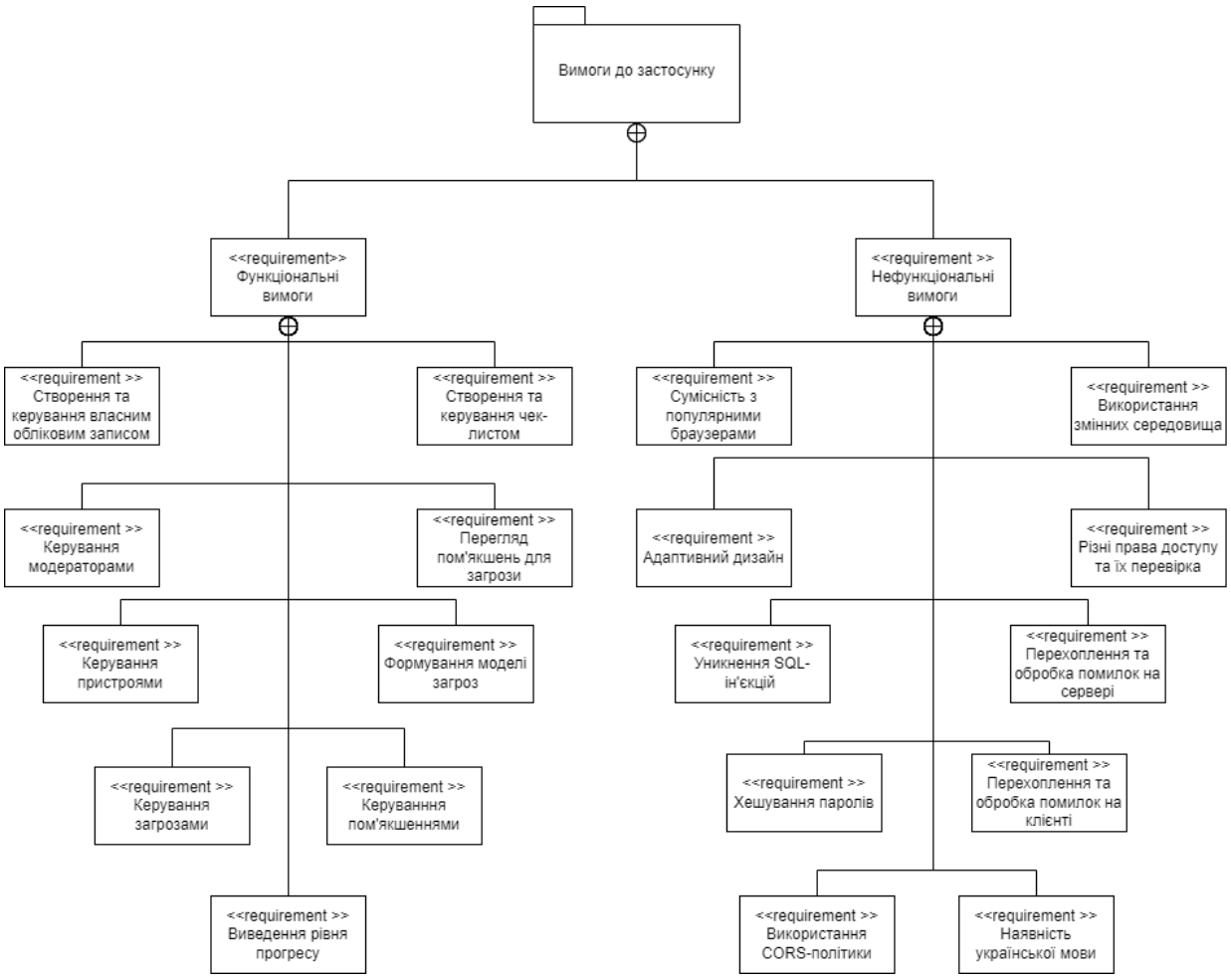
ДП.045440-07-99. Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв. Динамічна логічна структура вебзастосунку. Діаграма діяльності



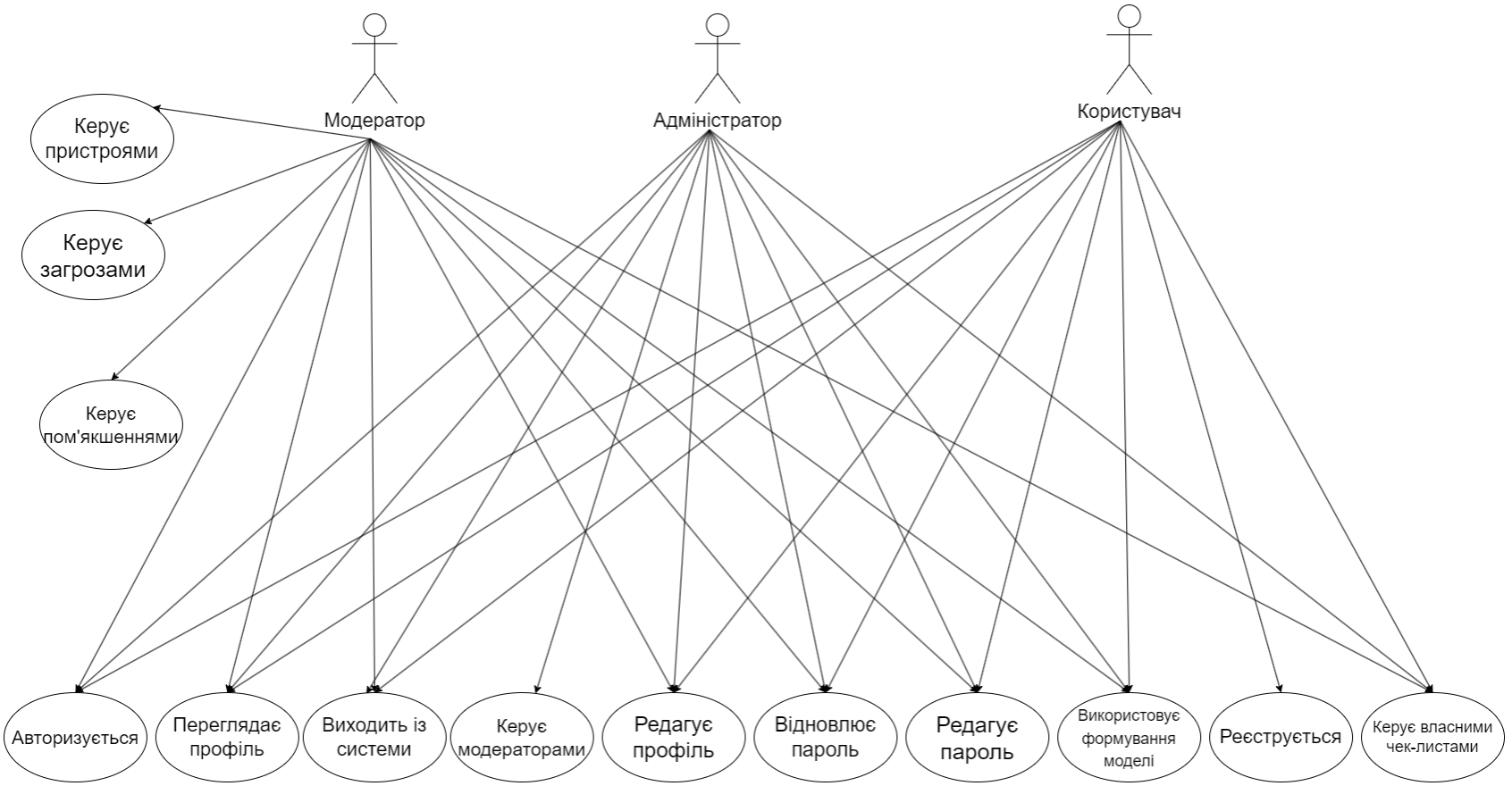
ДП.045440-08-99. Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв. Фізична структура компонентів вебзастосунку. Діаграма компонентів



ДП.045440-09-99. Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв. Фізична структура вузлів вебзастосунку. Діаграма розгортання



Вимоги до вебзастосунку
Максименко Д. Ю., група КП-91



Варіанти використання вебзастосунку
Максименко Д. Ю., група КП-91

Додаток 2

Фрагменти коду програми

DiplomaMobileDevicesSecurityApp.java

```
package com.diploma;

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

@SpringBootApplication
public class DiplomaMobileDevicesSecurityApp {
    public static void main (String[] args) {

SpringApplication.run(DiplomaMobileDevicesSecurityApp.class, args);
    }
}
```

ModelGenerationRepositoryImpl.java

```
package com.diploma.repository;

import com.diploma.model.GeneratedModelEntry;
import com.diploma.model.ChecklistEntry;
import com.diploma.model.Device;
import
com.diploma.repository.interfaces.ModelGenerationRepository;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

@Repository
public      class      ModelGenerationRepositoryImpl      extends
BaseJdbcRepository implements ModelGenerationRepository {

    @Value("${sql.modelGeneration.generate}")
    private String generateModelRequest;

    @Value("${sql.modelGeneration.createChecklist}")
    private String createChecklistRequest;


@Value("${sql.modelGeneration.createChecklistEntriesBeginning}")
    private String createChecklistEntriesBeginning;

    @Value("${sql.modelGeneration.createChecklistEntriesPart}")
    private String createChecklistEntriesPart;


    public              ModelGenerationRepositoryImpl(JdbcTemplate
jdbcTemplate) {
        super(jdbcTemplate);
    }
```

```
    @Override
    public                          Collection<GeneratedModelEntry>
getGeneratedModel(Device device) {
        return jdbcTemplate.query(
                generateModelRequest,
                new
BeanPropertyRowMapper<>(GeneratedModelEntry.class),
                device.getOs(),
                device.getOs() + " " + device.getOsMinVersion(),
                device.getOs(),
                device.getOs() + " " + device.getOsMaxVersion(),
                device.getOs(),
                device.getChipset(),
                device.getFingerprintScanner(),
                device.getFaceRecognition()
        );
    }

    @Override
    public long createChecklist(String name, long deviceId, long
accountId) {
        return jdbcTemplate.queryForObject(createChecklistRequest,
Long.class, name, deviceId, accountId);
    }

    @Override
    public void createChecklistEntries(Collection<ChecklistEntry>
entries) {
        StringBuilder         requestBuilder         =         new
StringBuilder(createChecklistEntriesBeginning);
        List<Long> params = new ArrayList<>();
        for (ChecklistEntry entry : entries) {
            requestBuilder.append(createChecklistEntriesPart);
            params.add(entry.getChecklistId());
            params.add(entry.getTechniqueId());
            params.add(entry.getMitigationId());
        }
        requestBuilder.deleteCharAt(requestBuilder.length() - 1);
        jdbcTemplate.update(requestBuilder.toString(),
params.toArray());
    }
}
```

TechniqueMitigationServiceImpl.java

```
package com.diploma.service;

import com.diploma.dto.ApplicabilityDTO;
import com.diploma.dto.PaginationDTO;
import com.diploma.dto.TechniqueMitigationDTO;
import com.diploma.dto.TechniqueMitigationWithLinksDTO;
import com.diploma.exception.CustomException;
import com.diploma.mapper.TechniqueMitigationMapper;
import com.diploma.model.TechniqueMitigation;
import com.diploma.model.Applicability;
import com.diploma.model.TechniqueMitigationEntity;
import
com.diploma.repository.interfaces.TechniqueMitigationRepository;
```

```
import com.diploma.service.interfaces.TechniqueMitigationService;
import lombok.AllArgsConstructor;
import org.springframework.dao.DataAccessException;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Collection;
import java.util.List;

@AllArgsConstructor
@Service
public class TechniqueMitigationServiceImpl implements
TechniqueMitigationService {
    private final TechniqueMitigationRepository
techniqueMitigationRepository;
    private final TechniqueMitigationMapper
techniqueMitigationMapper;

    @Override
    public TechniqueMitigationDTO getTechniqueMitigationById(Long
id, TechniqueMitigationEntity entity) {
        try {
            return
techniqueMitigationMapper.techniqueMitigationToTechniqueMitigationDT
O(techniqueMitigationRepository.findByIdAndEntity(id, entity));
        } catch (EmptyResultDataAccessException ex) {
            throw new CustomException(HttpStatus.NOT_FOUND,
String.format("%s with id %s not found.", entity.toString(), id));
        }
    }

    @Override
    public ApplicabilityDTO getApplicabilityByTechniqueId(Long id)
{
        try {
            List<Applicability> applicabilityList =
techniqueMitigationRepository.getApplicabilityByTechniqueId(id);
            if (applicabilityList.isEmpty()) {
                return new ApplicabilityDTO();
            }
            return
techniqueMitigationMapper.applicablityToApplicabilityDTO(applicabili
tyList.get(0));
        } catch (EmptyResultDataAccessException ex) {
            throw new CustomException(HttpStatus.NOT_FOUND,
String.format("Technique with id %s not found.", id));
        }
    }

    @Override
    public PaginationDTO<TechniqueMitigationDTO>
getFilteredTechniquesMitigations(String name, int limit, boolean
order, int currentPage, TechniqueMitigationEntity entity) {
        name = "%" + name + "%";
```

```
            int                          count                          =
techniqueMitigationRepository.countFilteredTechniquesMitigations(nam
e, entity);
            Collection<TechniqueMitigation>
techniqueMitigationCollection                                          =
techniqueMitigationRepository.filterTechniquesMitigations(
                    name, order, limit, currentPage * limit, entity
            );
            return new PaginationDTO<>(

techniqueMitigationMapper.techniqueMitigationCollectionToDtoCollecti
on(techniqueMitigationCollection), count
            );
        }

        @Override
        @Transactional
        public                           TechniqueMitigationWithLinksDTO
getTechniqueMitigationWithLinksById(Long                               id,
TechniqueMitigationEntity entity) {
            try {
                TechniqueMitigationWithLinksDTO mainDto =

techniqueMitigationMapper.techniqueMitigationToTechniqueMitigationWi
thLinksDTO(

techniqueMitigationRepository.findByIdAndEntity(id, entity));

mainDto.setLinks(techniqueMitigationMapper.techniqueMitigationCollec
tionToDtoCollection(

techniqueMitigationRepository.findLinksByIdAndEntity(id, entity)));
                return mainDto;
            } catch (EmptyResultDataAccessException ex) {
                throw    new    CustomException(HttpStatus.NOT_FOUND,
String.format("%s with id %s not found.", entity.toString(), id));
            }
        }

        @Override
        public      void      deleteTechniqueMitigation(Long      id,
TechniqueMitigationEntity entity) {
            try {
                techniqueMitigationRepository.deleteByIdAndEntity(id,
entity);
            } catch (DataAccessException e) {
                throw    new    CustomException(HttpStatus.NOT_FOUND,
String.format("%s with id %s not found.", entity, id));
            }
        }

        @Override
        @Transactional
        public  void  createTechnique(TechniqueMitigationWithLinksDTO
techniqueMitigationWithLinksDTO, ApplicabilityDTO applicabilityDTO) {
            TechniqueMitigation              technique              =
techniqueMitigationMapper.techniqueMitigationWithLinksDTOToTechnique
Mitigation(techniqueMitigationWithLinksDTO);
```

```
            List<TechniqueMitigation>              links              =
techniqueMitigationMapper.dtoCollectionTotechniqueMitigationCollecti
on(techniqueMitigationWithLinksDTO.getLinks());
            Applicability                applicability                =
techniqueMitigationMapper.applicablityDTOToApplicability(applicabili
tyDTO);
            long                        techniqueId                        =
techniqueMitigationRepository.createTechniqueMitigation(technique,
TechniqueMitigationEntity.TECHNIQUE);

techniqueMitigationRepository.createApplicability(techniqueId,
applicability);

techniqueMitigationRepository.createTechniqueLinks(techniqueId,
links);
        }

        @Override
        @Transactional
        public          void          updateTechnique(long          id,
TechniqueMitigationWithLinksDTO      techniqueMitigationWithLinksDTO,
ApplicabilityDTO applicabilityDTO) {
            try {

techniqueMitigationRepository.deleteApplicabilitiesByTechniqueId(id)
;

techniqueMitigationRepository.deleteLinksByTechniqueMitigationId(id,
TechniqueMitigationEntity.TECHNIQUE);
                TechniqueMitigation             technique             =
techniqueMitigationMapper.techniqueMitigationWithLinksDTOToTechnique
Mitigation(techniqueMitigationWithLinksDTO);
                technique.setId(id);

techniqueMitigationRepository.updateTechniqueMitigation(technique,
TechniqueMitigationEntity.TECHNIQUE);
                List<TechniqueMitigation>            links            =
techniqueMitigationMapper.dtoCollectionTotechniqueMitigationCollecti
on(techniqueMitigationWithLinksDTO.getLinks());
                Applicability              applicability              =
techniqueMitigationMapper.applicablityDTOToApplicability(applicabili
tyDTO);
                techniqueMitigationRepository.createApplicability(id,
applicability);
                techniqueMitigationRepository.createTechniqueLinks(id,
links);

            } catch (DataAccessException e) {
                throw     new     CustomException(HttpStatus.NOT_FOUND,
String.format("Technique with id %s not found.", id));
            }
        }

        @Override
        @Transactional
        public  void  createMitigation(TechniqueMitigationWithLinksDTO
techniqueMitigationWithLinksDTO) {
```

```
            TechniqueMitigation              mitigation              =
techniqueMitigationMapper.techniqueMitigationWithLinksDTOToTechnique
Mitigation(techniqueMitigationWithLinksDTO);
            List<TechniqueMitigation>             links             =
techniqueMitigationMapper.dtoCollectionTotechniqueMitigationCollecti
on(techniqueMitigationWithLinksDTO.getLinks());
            long                     mitigationId                   =
techniqueMitigationRepository.createTechniqueMitigation(mitigation,
TechniqueMitigationEntity.MITIGATION);

techniqueMitigationRepository.createMitigationLinks(mitigationId,
links);
        }

        @Override
        @Transactional
        public         void         updateMitigation(long         id,
TechniqueMitigationWithLinksDTO techniqueMitigationWithLinksDTO) {
            try {

techniqueMitigationRepository.deleteLinksByTechniqueMitigationId(id,
TechniqueMitigationEntity.MITIGATION);
                TechniqueMitigation            mitigation            =
techniqueMitigationMapper.techniqueMitigationWithLinksDTOToTechnique
Mitigation(techniqueMitigationWithLinksDTO);
                mitigation.setId(id);

techniqueMitigationRepository.updateTechniqueMitigation(mitigation,
TechniqueMitigationEntity.MITIGATION);
                List<TechniqueMitigation>           links           =
techniqueMitigationMapper.dtoCollectionTotechniqueMitigationCollecti
on(techniqueMitigationWithLinksDTO.getLinks());

techniqueMitigationRepository.createMitigationLinks(id, links);

            } catch (DataAccessException e) {
                throw    new    CustomException(HttpStatus.NOT_FOUND,
String.format("Technique with id %s not found.", id));
            }
        }
    }
```

ChecklistMapper.java

```
    package com.diploma.mapper;

    import com.diploma.dto.ChecklistDTO;
    import com.diploma.dto.ChecklistEntryDTO;
    import com.diploma.model.Checklist;
    import com.diploma.model.ChecklistEntry;
    import org.mapstruct.Mapper;
    import org.mapstruct.Mapping;
    import org.mapstruct.Mappings;
    import org.springframework.stereotype.Service;

    import java.util.Collection;

    @Service
```

```
   @Mapper(componentModel = "spring")
   public interface ChecklistMapper {
       @Mappings({
               @Mapping(target="id", source="checklistEntries.id"),
               @Mapping(target="checklistId",
source="checklistEntries.checklistId"),
               @Mapping(target="techniqueId",
source="checklistEntries.techniqueId"),
               @Mapping(target="techniqueName",
source="checklistEntries.techniqueName"),
               @Mapping(target="techniqueDescription",
source="checklistEntries.techniqueDescription"),
               @Mapping(target="mitigationId",
source="checklistEntries.mitigationId"),
               @Mapping(target="mitigationName",
source="checklistEntries.mitigationName"),
               @Mapping(target="mitigationDescription",
source="checklistEntries.mitigationDescription"),
               @Mapping(target="isChecked",
source="checklistEntries.isChecked")
       })
       Collection<ChecklistEntryDTO>
checklistEntryCollectionToDTOCollection(Collection<ChecklistEntry>
checklistEntries);

       @Mappings({
               @Mapping(target="id", source="checklist.id"),
               @Mapping(target="name", source="checklist.name"),
               @Mapping(target    =    "deviceName",    source    =
"checklist.deviceName")
       })
       ChecklistDTO checklistToChecklistDTO(Checklist checklist);

       @Mappings({
               @Mapping(target="id", source="checklists.id"),
               @Mapping(target="name", source="checklists.name"),
               @Mapping(target    =    "deviceName",    source    =
"checklists.deviceName")
       })
       Collection<ChecklistDTO>
checklistCollectionToDTOCollection(Collection<Checklist>
checklists);
   }
```

DeviceController.java

```
   package com.diploma.controller;

   import com.diploma.dto.DeviceDTO;
   import com.diploma.dto.PaginationDTO;
   import com.diploma.dto.DevicePredefinedValuesDTO;
   import com.diploma.service.interfaces.DeviceService;
   import io.swagger.annotations.ApiResponse;
   import io.swagger.annotations.ApiResponses;
   import org.springframework.security.access.prepost.PreAuthorize;
   import org.springframework.validation.annotation.Validated;
   import org.springframework.web.bind.annotation.*;
```

```
import javax.validation.Valid;
import javax.validation.constraints.Min;

@RestController
@Validated
@RequestMapping("/api/devices")
public class DeviceController {
    private final DeviceService deviceService;

    public DeviceController(DeviceService deviceService) {
        this.deviceService = deviceService;
    }

    @PreAuthorize("hasAnyRole('ROLE_MODERATOR',       'ROLE_USER',
'ROLE_ADMIN')")
    @GetMapping
    @ApiResponses(value = {
            @ApiResponse(code = 400, message = "Bad request")})
    public PaginationDTO<DeviceDTO> getFilteredDevices(
            @RequestParam(value = "pageSize", defaultValue = "12",
required = false) @Min(value = 1, message = "Page size must be higher
than 0") int pageSize,
            @RequestParam(value = "pageNum", defaultValue = "0",
required = false) @Min(value = 0, message = "Current page must be
higher than 0") int currentPage,
            @RequestParam(value  =  "name",  defaultValue  =  "",
required = false) String name,
            @RequestParam(value = "order", defaultValue = "true",
required = false) boolean order) {
        return  deviceService.getFilteredDevices(name,  pageSize,
order, currentPage);
    }

    @PreAuthorize("hasAnyRole('ROLE_MODERATOR')")
    @GetMapping(value = "/{id}")
    @ApiResponses(value = {
            @ApiResponse(code  =  400,  message  =  "Something  went
wrong"),
            @ApiResponse(code = 404, message = "Item not found")})
    public DeviceDTO getDeviceById(@PathVariable long id) {
        return deviceService.getDeviceById(id);
    }

    @PreAuthorize("hasRole('ROLE_MODERATOR')")
    @PostMapping
    @ApiResponses(value = {
            @ApiResponse(code  =  200,  message  =  "Device  has  been
added"),
            @ApiResponse(code  =  400,  message  =  "Something  went
wrong")})
    public  DeviceDTO  createDevice(@RequestBody  @Valid  DeviceDTO
deviceDTO) {
        return deviceService.createDevice(deviceDTO);
    }

    @PreAuthorize("hasRole('ROLE_MODERATOR')")
    @PutMapping(value = "/{id}")
    @ApiResponses(value = {
```

```
            @ApiResponse(code = 200, message = "Update
successful"),
            @ApiResponse(code = 400, message = "Something went
wrong"),
            @ApiResponse(code = 422, message = "Parameter(s) is/are
invalid"),
            @ApiResponse(code = 404, message = "Item not found")})
    public DeviceDTO updateDevice(@PathVariable long id,
@RequestBody @Valid DeviceDTO deviceDTO) {
        return deviceService.updateDevice(deviceDTO, id);
    }

    @PreAuthorize("hasAnyRole('ROLE_MODERATOR')")
    @DeleteMapping(value = "/{id}")
    public void deleteDevice(@PathVariable long id) {
        deviceService.deleteDevice(id);
    }

    @PreAuthorize("hasAnyRole('ROLE_MODERATOR')")
    @GetMapping(value = "/predefinedValues")
    @ApiResponses(value = {
            @ApiResponse(code = 400, message = "Bad request")})
    public DevicePredefinedValuesDTO getDevicePredefinedValues() {
        return deviceService.getDevicePredefinedValues();
     }
}
```

app.component.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Subject, Subscription, takeUntil } from 'rxjs';
import { Account } from './_models/account';
import { Role } from './_models/role';
import { AuthService } from './_services/auth.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent implements OnDestroy {
  title = 'Mobile Safety App';
  account : Account | null;
  Role = Role;
  subscription: Subscription;

  constructor(
    private authService: AuthService
  ){
    this.subscription = this.authService.account.subscribe((x =>
this.account = x));
  }
  ngOnDestroy(): void {
    this.subscription.unsubscribe();
  }

  logout(){
```

```
      this.authService.logout();
    }
  }
```

app.routing.module.ts

```
  import {NgModule} from '@angular/core';
  import {RouterModule, Routes} from '@angular/router';
  import {AuthFormsGuard, AuthGuard} from './_helpers';
  import {Role} from './_models';

  const accountModule = () => import('./account/account.module').then(x => x.AccountModule);
  const adminModule = () => import('./admin/admin.module').then(x => x.AdminModule);
  const profileModule = () => import('./profile/profile.module').then(x => x.ProfileModule);
  const modelGenerationModule = () => import('./model-generation/model-generation.module').then(x => x.ModelGenerationModule);
  const checklistModule = () => import('./checklist/checklist.module').then(x => x.ChecklistModule);
  const deviceModule = () => import('./device/device.module').then(x => x.DeviceModule);
  const techniqueModule = () => import('./technique/technique.module').then(x => x.TechniqueModule);
  const mitigationModule = () => import('./mitigation/mitigation.module').then(x => x.MitigationModule);

  const routes: Routes = [
    { path: '', redirectTo: '/account/signin', pathMatch: 'full' },
    { path: 'account', loadChildren: accountModule, canActivate: [AuthFormsGuard] },
    { path: 'admin', loadChildren: adminModule, canActivate: [AuthGuard], data: { roles: [Role.Admin, Role.User] } },
    { path: 'profile', loadChildren: profileModule, canActivate: [AuthGuard] },
    { path: 'model-generation', loadChildren: modelGenerationModule, canActivate: [AuthGuard], data: { roles: [Role.Admin, Role.User, Role.Moderator] } },
    { path: 'checklists', loadChildren: checklistModule, canActivate: [AuthGuard], data: { roles: [Role.Admin, Role.User, Role.Moderator] } },
    { path: 'devices', loadChildren: deviceModule, canActivate: [AuthGuard], data: { roles: [Role.Moderator] } },
    { path: 'techniques', loadChildren: techniqueModule, canActivate: [AuthGuard], data: { roles: [Role.Moderator] } },
    { path: 'mitigations', loadChildren: mitigationModule, canActivate: [AuthGuard], data: { roles: [Role.Moderator] } },
    { path: '**', redirectTo: '/account/signin', pathMatch: 'full' }
  ];

  @NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
  })
```

```
export class AppRoutingModule {
}
```

technique-mitigation.service.ts

```
import {Injectable} from '@angular/core';
import {environment} from "../../environments/environment";
import {HttpClient, HttpParams} from "@angular/common/http";
import {Observable} from "rxjs";
import {ActivatedRoute, Router} from "@angular/router";
import { TechniqueMitigation } from '../_models/technique-
mitigation';
import { StandardSearchParams } from '../_models/standard-search-
params';
import { Page } from '../_models';
import { TechniqueMitigationWithLinks } from '../_models/technique-
mitigation-with-links';
import { Applicability } from '../_models/applicability';
import { TechniqueApplicabilityWithLinks } from
'../_models/technique-applicability-with-links';

const baseUrl = `${environment.serverUrl}/techniquesMitigations`;

@Injectable({
  providedIn: 'root'
})
export class TechniqueMitigationService {
  searchParams: StandardSearchParams;

  constructor(
    private activatedRoute: ActivatedRoute,
    private router: Router,
    private http: HttpClient
  ) {
  }


  getTechniqueById(techniqueId: string):
Observable<TechniqueMitigation> {
    return
this.http.get<TechniqueMitigation>(`${baseUrl}/techniques/${techniqu
eId}`);
  }

  getMitigationById(mitigationIs: string):
Observable<TechniqueMitigation> {
    return
this.http.get<TechniqueMitigation>(`${baseUrl}/mitigations/${mitigat
ionIs}`);
  }

  getTechniqueWithLinksById(techniqueId: string):
Observable<TechniqueMitigationWithLinks> {
    return
this.http.get<TechniqueMitigationWithLinks>(`${baseUrl}/techniques/w
ithLinks/${techniqueId}`);
  }
```

```
    getMitigationWithLinksById(mitigationId:                string):
Observable<TechniqueMitigationWithLinks> {
      return
this.http.get<TechniqueMitigationWithLinks>(`${baseUrl}/mitigations/
withLinks/${mitigationId}`);
    }

    getApplicabilityByTechniqueId(techniqueId:              string):
Observable<Applicability> {
      return
this.http.get<Applicability>(`${baseUrl}/techniques/applicability/${
techniqueId}`);
    }

    getTechniqueList(currentPage:      number,      searchedParams:
StandardSearchParams,               pageSize:               number):
Observable<Page<TechniqueMitigation>> {
      return
this.http.get<Page<TechniqueMitigation>>(`${baseUrl}/techniques`, {
          params: new HttpParams()
            .set('pageSize', pageSize)
            .set('pageNum', currentPage)
            .set('name', searchedParams.name)
            .set('order', searchedParams.order == 'asc')});
    }

    getMitigationList(currentPage:      number,      searchedParams:
StandardSearchParams,               pageSize:               number):
Observable<Page<TechniqueMitigation>> {
      return
this.http.get<Page<TechniqueMitigation>>(`${baseUrl}/mitigations`, {
          params: new HttpParams()
            .set('pageSize', pageSize)
            .set('pageNum', currentPage)
            .set('name', searchedParams.name)
            .set('order', searchedParams.order == 'asc')});
    }

    getTechniquesBySearch(searchParams:        StandardSearchParams,
pageSize: number): Observable<Page<TechniqueMitigation>> {
      this.searchParams = searchParams;
      return this.getTechniqueList(0, this.searchParams, pageSize);
    }

    getTechniquesByPageNum(currentPage: number, pageSize: number):
Observable<Page<TechniqueMitigation>> {
      return  this.getTechniqueList(currentPage,  this.searchParams,
pageSize);
    }

    deleteTechnique(id: string) : Observable<Object> {
      return this.http.delete(`${baseUrl}/techniques/${id}`);
    }

    createTechnique(dto:           TechniqueApplicabilityWithLinks):
Observable<Object> {
      return this.http.post(`${baseUrl}/techniques`, dto);
```

```
    }

    editTechnique(dto:            TechniqueApplicabilityWithLinks):
Observable<Object>{
      return
this.http.put(`${baseUrl}/techniques/${dto.techniqueWithLinks.id}`,
dto);
    }

    getMitigationsBySearch(searchParams:      StandardSearchParams,
pageSize: number): Observable<Page<TechniqueMitigation>> {
      this.searchParams = searchParams;
      return this.getMitigationList(0, this.searchParams, pageSize);
    }

    getMitigationsByPageNum(currentPage: number, pageSize: number):
Observable<Page<TechniqueMitigation>> {
      return  this.getMitigationList(currentPage,  this.searchParams,
pageSize);
    }

    deleteMitigation(id: string) : Observable<Object> {
      return this.http.delete(`${baseUrl}/mitigations/${id}`);
    }

    createMitigation(dto:            TechniqueMitigationWithLinks):
Observable<Object> {
      return this.http.post(`${baseUrl}/mitigations`, dto);
    }

    editMitigation(dto:              TechniqueMitigationWithLinks):
Observable<Object>{
      return this.http.put(`${baseUrl}/mitigations/${dto.id}`, dto);
    }

  }
```

checklist-info.component.ts

```
  import {Component, OnDestroy, OnInit} from '@angular/core';
  import  {ReplaySubject,  concatAll,  delay,  from,  of,  takeUntil,
startWith} from 'rxjs';
  import {ActivatedRoute, Router} from "@angular/router";
  import { AlertService} from '../../_services';
  import { Checklist } from 'src/app/_models/checklist';
  import         {         ChecklistService         }         from
'src/app/_services/checklist.service';
  import       {       TechniqueMitigationService       }       from
'src/app/_services/technique-mitigation.service';
  import   {   ViewTechniqueMitigationComponent   }   from   '../view-
technique-mitigation/view-technique-mitigation.component';
  import { MatDialog } from '@angular/material/dialog';
  import { Device } from 'src/app/_models/device';

  @Component({
    selector: 'app-checklist-info',
    templateUrl: './checklist-info.component.html',
```

```
  styleUrls: ['./checklist-info.component.scss'],
})
export class ChecklistInfoComponent implements OnInit {
  length: number;
  checklist: Checklist;
  device: Device;
  columnsToDisplay = ['technique', 'mitigation', 'tick'];
  destroy: ReplaySubject<any> = new ReplaySubject<any>();
  isNotFound: boolean = false;
  alertMessage: string;
  currentProgress: number;
  progressIncrement: number;
  progressValue$: any;

  constructor(
    private checklistService: ChecklistService,
    private techniqueMitigationService:
TechniqueMitigationService,
    private route: ActivatedRoute,
    private alertService: AlertService,
    private dialog: MatDialog,
    private router: Router) {
  }

  ngOnInit(): void {
    const id: string | null =
this.route.snapshot.paramMap.get('id');
    if (id && Number(id)) {
      this.getChecklistInfo(id);
    }
    else {
      this.router.navigateByUrl("/checklists");
    }
  }

  private getChecklistInfo(id: string) : void {
    this.checklistService.getChecklistById(id)
      .pipe(takeUntil(this.destroy))
      .subscribe(
        {next: response => {
          this.checklist = response;
          this.progressIncrement = 1 / (response.entries.length) *
100;
          const checkedEntriesCount: number =
response.entries.filter((obj) => {
            return obj.checked === true;
          }).length;
          this.currentProgress = checkedEntriesCount /
response.entries.length * 100;
          this.progressValue$ = from([
            of(this.currentProgress).pipe(delay(800))
          ]).pipe(
            startWith(of(this.currentProgress)),
            concatAll()
          );
          },
          error: error => {
            this.displayError(error);
```

```
            }}
        );
    }

    openTechniqueDialog(id: string): void {
      console.log(id);
      this.techniqueMitigationService.getTechniqueById(id)
          .pipe(takeUntil(this.destroy))
          .subscribe({
            next: response => {
              const                    dialogRef                    =
this.dialog.open(ViewTechniqueMitigationComponent, {
                data: {techniqueMitigation: response}
              });
            },
            error: error => {
              switch(error.status){
                case 404:
                  this.alertService.error(error.error.message,
false, false, "error-dialog");
                  break;
                default:
                  this.alertService.error("несподівана      помилка,
спробуйте пізніше", false, false, "error-dialog");
                  break;
              }
              this.alertService.error(this.alertMessage);
            }});
    }

    openMitigationDialog(id: string): void {
      this.techniqueMitigationService.getMitigationById(id)
          .pipe(takeUntil(this.destroy))
          .subscribe({
            next: response => {
              const                    dialogRef                    =
this.dialog.open(ViewTechniqueMitigationComponent, {
                data: {techniqueMitigation: response}
              });
            },
            error: error => {
              switch(error.status){
                case 404:
                  this.alertService.error(error.error.message,
false, false, "error-dialog");
                  break;
                default:
                  this.alertService.error("несподівана      помилка,
спробуйте пізніше", false, false, "error-dialog");
                  break;
              }
              this.alertService.error(this.alertMessage);
            }});
    }

    changeIsChecked(index: number, id: string, checked: boolean):
void {
      this.alertService.clear();
```

```
      let multiplier: number = 1;
      if (checked) {
        multiplier = -1;
      }
      this.progressValue$ = from ([
        of(this.currentProgress    +    this.progressIncrement    *
multiplier).pipe(delay(200))
      ]).pipe(
        startWith(of(this.currentProgress)),
        concatAll()
      );
      this.currentProgress += this.progressIncrement * multiplier;
      this.checklistService.changeIsChecked(id, !checked)
        .pipe(takeUntil(this.destroy))
        .subscribe({
            next: () => {
              this.checklist.entries[index].checked = !checked;
            },
            error: error => {
              if (error.status == 404) {
                this.alertService.error(error.error.message,  false,
true);

              } else {
                this.alertService.error("Сталася помилка, будь ласка
повторіть пізніше.", false, true);

              }
            }
          }
        )

    }

    displayError(error: any) : void {
      switch (error.status) {
        case 400:
          this.alertService.error("Щось пішло не так",true,true);
          break;
        case 404:
          this.isNotFound = true;
          break;
        default:
          this.alertService.error("Сталася помилка на сервері, будь
ласка повторіть пізніше.",true,true);
          break;
      }

    }

    ngOnDestroy(): void {
      this.destroy.next(null);
      this.destroy.complete();
    }

  }
```

device-list-page.component.ts

```
import {Component, ElementRef, ViewChild} from '@angular/core';
import {FormBuilder, FormControl, FormGroup, Validators} from "@angular/forms";
import {PageEvent} from "@angular/material/paginator";
import {Observable, ReplaySubject} from "rxjs";
import {AlertService, AuthService} from "../../_services";
import {Page} from "../../_models/page";
import {map, startWith, takeUntil} from "rxjs/operators";
import {MatDialog, MatDialogConfig} from "@angular/material/dialog";
import {COMMA, ENTER} from '@angular/cdk/keycodes';
import { Sort, SortDirection } from '@angular/material/sort';
import { DeviceService } from 'src/app/_services/device.service';
import { DeletionConfirmationComponent } from '../deletion-confirmation/deletion-confirmation.component';
import { Device } from 'src/app/_models/device';
import { StandardSearchParams } from 'src/app/_models/standard-search-params';
import { DeviceCreationComponent } from '../device-creation/device-creation.component';
import { DeviceEditComponent } from '../device-edit/device-edit.component';


@Component({
  selector: 'device-list-page',
  templateUrl: './device-list-page.component.html',
  styleUrls: ['./device-list-page.component.scss']
})
export class DeviceListPageComponent {
  pageContent: Page<Device>;
  searchForm: FormGroup;
  destroy: ReplaySubject<any> = new ReplaySubject<any>();
  columnsToDisplay = ['name', 'os', 'os_min_version', 'os_max_version', 'chipset', 'fingerprint_scanner', 'face_recognition', 'actions'];
  pageSize: number = 12;
  currentPage: number;
  alertMessage: string;
  userRole?: string;
  sortOrder: SortDirection = 'asc';

  constructor(
    private deviceService: DeviceService,
    private alertService: AlertService,
    private dialog: MatDialog,
    private formBuilder: FormBuilder
  ) {
  }


  ngOnInit(): void {
    this.searchForm = this.createFormGroup();
    this.getDevicesBySearch();
  }
```

```
  createFormGroup(): FormGroup {
    return this.formBuilder.group({
      name: ['']
    });
  }


  getDevicesBySearch(): void {
    const filter: StandardSearchParams = this.searchForm.value;
    filter.order = this.sortOrder;
    this.deviceService.getDevicesBySearch(this.searchForm.value,
this.pageSize)
      .pipe(takeUntil(this.destroy))
      .subscribe({
        next: response => {
          this.pageContent = response;
          this.currentPage = 0;
        },
        error: error => {
          this.displayError(error);
        }
      });
  }

  getDevicePage(pageIndex: number, pageSize: number): void {
    this.deviceService.getDevicesByPageNum(pageIndex, pageSize)
      .pipe(takeUntil(this.destroy))
      .subscribe({
        next: response => {
          if (response.content.length === 0) {
            this.getDevicePage(pageIndex - 1, pageSize);
          }
          this.pageContent = response;
          this.currentPage = pageIndex;
          this.pageSize = pageSize;
        },
        error: error => {
          this.displayError(error);
        }
      });
  }

  confirmDeletion(id: string): void {
    const                        dialogRef                        =
this.dialog.open(DeletionConfirmationComponent);
    dialogRef.afterClosed().subscribe(dialogResult => {
      if (dialogResult) {
        this.deviceService.deleteDevice(id)
      .pipe(takeUntil(this.destroy))
      .subscribe({
        next: () => {
          this.alertService.success("Пристрій
видалено",true,true);
          this.getDevicePage(this.currentPage, this.pageSize);
        },
        error: error => {
          this.displayError(error);
```

```
          }
        });
        }
      });
    }

    sortData(sortOrder: string) : void {
      sortOrder === 'desc' ? this.sortOrder = 'desc' : this.sortOrder
= 'asc';
      this.getDevicesBySearch();
    }


    paginationHandler(pageEvent: PageEvent): void {
        this.getDevicePage(pageEvent.pageIndex, pageEvent.pageSize);
    }

    createDevice() {
      const dialogConfig = new MatDialogConfig();
      dialogConfig.disableClose = true;
      dialogConfig.autoFocus = true;

      const  dialogRef  =  this.dialog.open(DeviceCreationComponent,
dialogConfig);

dialogRef.afterClosed().pipe(takeUntil(this.destroy)).subscribe(()
=> {
        this.getDevicePage(this.currentPage, this.pageSize);
      })
    }

    editDevice(device: Device){
      const dialogConfig = new MatDialogConfig();
      dialogConfig.disableClose = true;
      dialogConfig.autoFocus = true;
      let dataDialog = Object.assign({}, device);
      dialogConfig.data = {
        device: dataDialog
      };
      const   dialogRef   =   this.dialog.open(DeviceEditComponent,
dialogConfig);

dialogRef.afterClosed().pipe(takeUntil(this.destroy)).subscribe((dat
a: Device) => {
        if(data){
          device.name = data.name;
          device.os = data.os;
          device.osMinVersion = data.osMinVersion;
          device.osMaxVersion = data.osMaxVersion;
          device.chipset = data.chipset;
          device.fingerprintScanner = data.fingerprintScanner;
          device.faceRecognition = data.faceRecognition;
        }
      })
    }


    displayError(error: any) : void {
```

```
      switch (error.status) {
        case 400:
          this.alertMessage = "Something went wrong";
          break;
        case 404:
          this.alertMessage = error.error.message;
          break;
        default:
          this.alertMessage = "There was an error on the server,
please try again later."
          break;
      }
      this.alertService.error(this.alertMessage,true,true);
    }

    ngOnDestroy(): void {
      this.destroy.next(null);
      this.destroy.complete();
    }
  }
```

deletion-confirmation.component.ts

```
  import { Component, Inject, OnInit } from '@angular/core';
  import { MatDialogRef, MAT_DIALOG_DATA } from
'@angular/material/dialog';

  @Component({
    selector: 'app-deletion-confirmation',
    templateUrl: './deletion-confirmation.component.html',
    styleUrls: ['./deletion-confirmation.component.scss']
  })
  export class DeletionConfirmationComponent implements OnInit {

      constructor(public dialogRef:
MatDialogRef<DeletionConfirmationComponent>) {
      }

      ngOnInit() {
      }

      onConfirm(): void {
        this.dialogRef.close(true);
      }

      onDismiss(): void {
        this.dialogRef.close(false);
      }
    }
```

mitigation-add-edit.component.ts

```
  import { Component, OnInit } from '@angular/core';
  import { FormBuilder, FormControl, FormGroup, Validators } from
'@angular/forms';
  import { MatDialog, MatDialogConfig } from
'@angular/material/dialog';
```

```
import { ActivatedRoute, Router } from '@angular/router';
import { ReplaySubject, takeUntil } from 'rxjs';
import { AlertService} from 'src/app/_services';
import { DevicePredefinedValues } from 'src/app/_models/device-predefined-values';
import { TechniqueMitigation } from 'src/app/_models/technique-mitigation';
import { TechniqueMitigationService } from 'src/app/_services/technique-mitigation.service';
import { Applicability } from 'src/app/_models/applicability';
import { TechniqueMitigationWithLinks } from 'src/app/_models/technique-mitigation-with-links';
import { LinksEditComponent } from '../links-edit/links-edit.component';
import { DeviceService } from 'src/app/_services/device.service';

@Component({
  selector: 'mitigation-add-edit',
  templateUrl: './mitigation-add-edit.component.html',
  styleUrls: ['./mitigation-add-edit.component.scss']
})
export class MitigationAddEditComponent implements OnInit {

  columnsToDisplay: string[] = ['name', 'description', 'actions'];
  devicePredefinedValues: DevicePredefinedValues;
  destroy: ReplaySubject<any> = new ReplaySubject<any>();
  mitigation: TechniqueMitigationWithLinks;
  applicability: Applicability;
  title: string;
  modeEdit: boolean = false;
  form: FormGroup;

  constructor(
    private dialog: MatDialog,
    private techniqueMitigationService: TechniqueMitigationService,
    private deviceService: DeviceService,
    private formBuilder: FormBuilder,
    private alertService: AlertService,
    private router: Router,
    private activatedRoute: ActivatedRoute
  ) {
    this.form = this.formBuilder.group({});
  }

  ngOnInit(): void {
    const id = this.activatedRoute.snapshot.params['id'];
    if (id) {
      this.modeEdit = true;

this.techniqueMitigationService.getMitigationWithLinksById(id).pipe(takeUntil(this.destroy)).subscribe({
        next: (data: TechniqueMitigationWithLinks) => {
          this.mitigation = data;
          this.mitigation.id = id;
          this.form.addControl('name', new FormControl(this.mitigation.name, [Validators.required, Validators.maxLength(40)]));
```

```
            this.form.addControl('description',                    new
FormControl(this.mitigation.description,        [Validators.required,
Validators.maxLength(1000)]));
          },
          error: () => this.router.navigate(['/mitigations'])
        });
      }
      else {
        this.mitigation = { id: "", name: "", description: "", links:
[] }
        this.applicability   =   {   os:   "",   osMinVersion:   "",
osMaxVersion:    "",    chipset:    "",    fingerprintScanner:    "",
faceRecognition: "" }
        this.form = this.formBuilder.group({
          name:               ['',               [Validators.required,
Validators.maxLength(40)]],
          description:          ['',          [Validators.required,
Validators.maxLength(1000)]],
          os: [''],
          osMinVersion: ['', [Validators.maxLength(10)]],
          osMaxVersion: ['', [Validators.maxLength(10)]],
          chipset: ['', [Validators.maxLength(40)]],
          fingerprintScanner: [''],
          faceRecognition: ['']
        });
      }
      this.title  =  this.modeEdit  ?  "Редагування  пом'якшення"  :
"Додавання пом'якшення";
      this.getDevicePredefinedValues();
    }

    addLink(): void {
      const dialogConfig = new MatDialogConfig();
      dialogConfig.disableClose = true;
      dialogConfig.autoFocus = false;
      dialogConfig.data = {
        selectedLinks: this.mitigation.links
      }
      const    dialogRef    =    this.dialog.open(LinksEditComponent,
dialogConfig);

dialogRef.afterClosed().pipe(takeUntil(this.destroy)).subscribe((dat
a: TechniqueMitigation[]) => this.mitigation.links = [...data]);
    }

    getDevicePredefinedValues(): void {

this.deviceService.getDevicePredefinedValues().pipe(takeUntil(this.d
estroy))
        .subscribe({
          next: data => this.devicePredefinedValues = data,
          error: () => this.mitigation.links = []
        });
    }

    removeLink(id: string): void {
      this.mitigation.links = this.mitigation.links.filter(v => v.id
!= id);
```

```
    }

    onSubmitForm(): void {
      this.alertService.clear();
      if (this.form.valid) {
        if (!this.modeEdit) {
          this.applicability = { os: this.form.value.os,
osMinVersion: this.form.value.osMinVersion, osMaxVersion:
this.form.value.osMaxVersion,
            chipset: this.form.value.chipset, fingerprintScanner:
this.form.value.fingerprintScanner, faceRecognition:
this.form.value.faceRecognition }
            console.log(this.applicability);

this.techniqueMitigationService.createMitigation(this.mitigation).pi
pe(takeUntil(this.destroy)).subscribe({
            next: () => {
              this.alertService.success("Загрозу додано.", true,
true);
              this.router.navigate(['../'], { relativeTo:
this.activatedRoute });
            },
            error: () => this.alertService.error("Сталася серверна
помилка.", false, false)
          });
        }
        else {
          this.applicability = { os: this.form.value.os,
osMinVersion: this.form.value.osMinVersion, osMaxVersion:
this.form.value.osMaxVersion,
            chipset: this.form.value.chipset, fingerprintScanner:
this.form.value.fingerprintScanner, faceRecognition:
this.form.value.faceRecognition }
            console.log(this.applicability);

this.techniqueMitigationService.editMitigation(this.mitigation).pipe
(takeUntil(this.destroy)).subscribe({
            next: () => {
              this.alertService.success("Загрозу оновлено.", true,
true);
              this.router.navigateByUrl("/mitigations");
            },
            error: () => this.alertService.error("There was a server
error. Please try again later.", false, false)
          });
        }
      }
    }
  }
```

Додаток З

Копія презентації

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНІ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО"
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
КОМП'ЮТЕРНИХ СИСТЕМ

ВЕБЗАСТОСУНОК ДЛЯ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТОЇВ

Виконав: Максименко Дмитро Юрійович

Керівник: Цуркан Василь Васильович

Київ – 2023

АКТУАЛЬНІСТЬ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

З кожним роком спостерігається тенденція до збільшення як кількості, так і складності мобільних пристроїв, які щоденно використовуються. Це призводить до підвищення і кількості потенційно цікавих для зловмисників цілей у мобільних пристроях. Тому важливо забезпечувати протидіяти загрозам безпеці мобільних пристроїв, пом'якшувати їхній вплив.

Отже, моделювання вебзастосунку для формування моделі загроз безпеці мобільних пристроїв є актуальним завданням.

ПОСТАНОВКА ЗАВДАННЯ

Об'єкт дослідження: процес формування моделі загроз безпеці мобільних пристроїв.

Предмет дослідження: програмні застосунки для формування моделі загроз безпеці мобільних пристроїв.

Мета проєкту: програмно реалізувати вебзастосунок для формування моделі загроз безпеці мобільних пристроїв.

Часткові завдання:

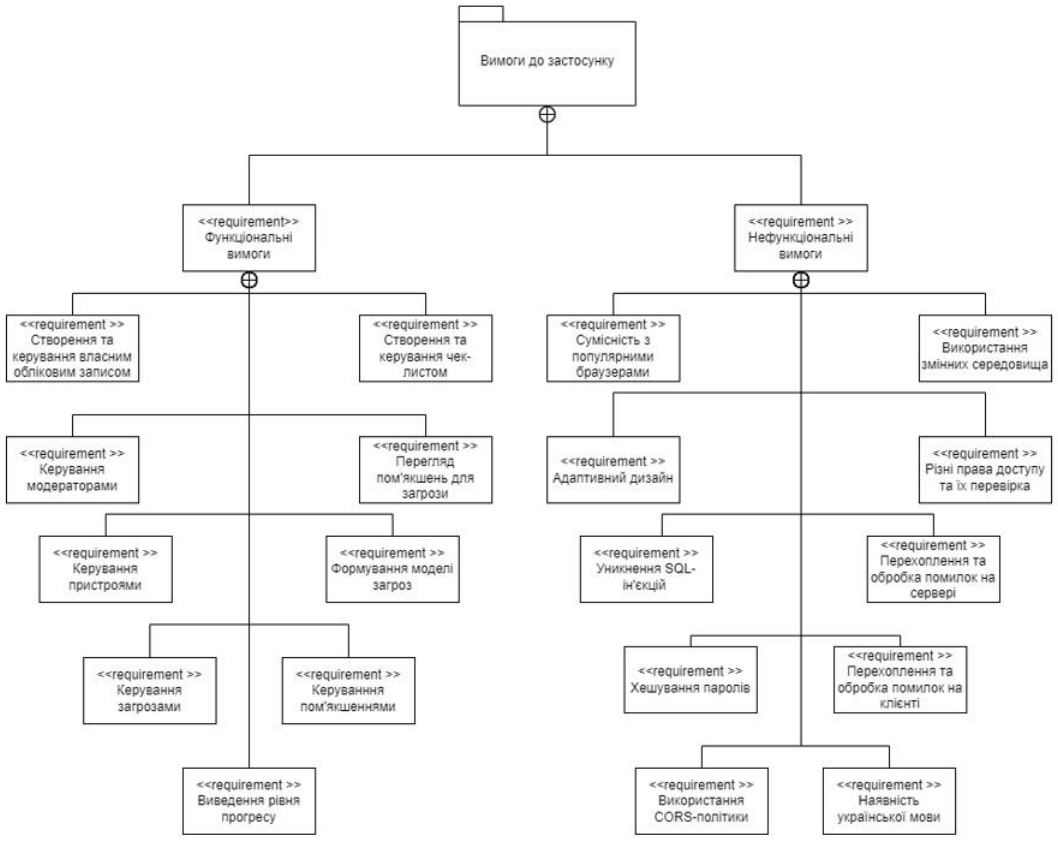
1. Специфікувати вимоги до вебзастосунку.
2. Розробити концептуальну модель вебзастосунку.
3. Розробити об’єктно-орієнтовану модель вебзастосунку.
4. Розробити робочий проєкт вебзастосунку.

АНАЛІЗ ВІДОМИХ ПРИКЛАДІВ ПРОГРАМНИХ ЗАСТОСУНКІВ

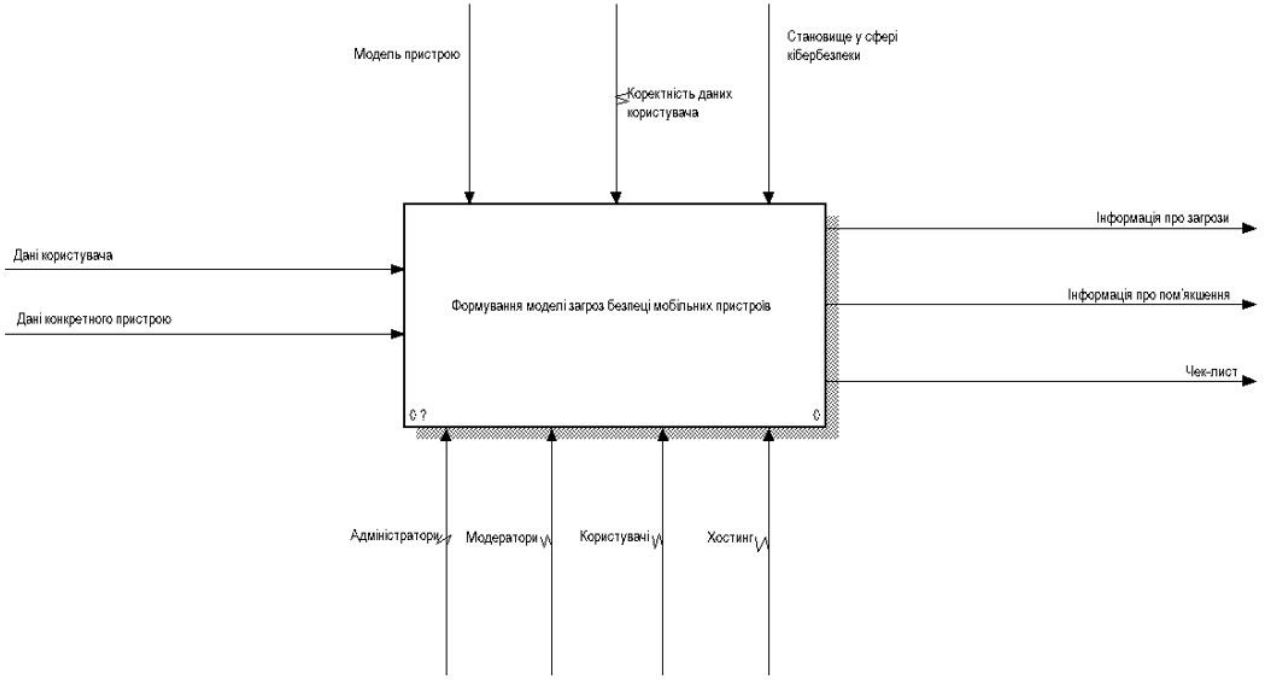
Критерій \ Застосунок	Microsoft Threat Modelling Tool	OWASP Mobile Application Security	Threat Modeler
Пристосованість та спеціалізованість на мобільних пристроях	-	+	+
Не потребує створення діаграм	-	+	-
Автоматично генерує загрози	+	-	+
Автоматично пропонує пом'якшення (заходи безпеки)	+	-	+
Надає можливість проставляти відмітки для відслідковування прогресу забезпечення безпеки	+	+	-
Є відкритим застосунком для будь-яких користувачів	+	+	-
Надає можливість формування моделі загроз для конкретного пристрою	-	-	+

Символи "+" та "–" означають, що наведена функціональна можливість реалізована або ні відповідно.

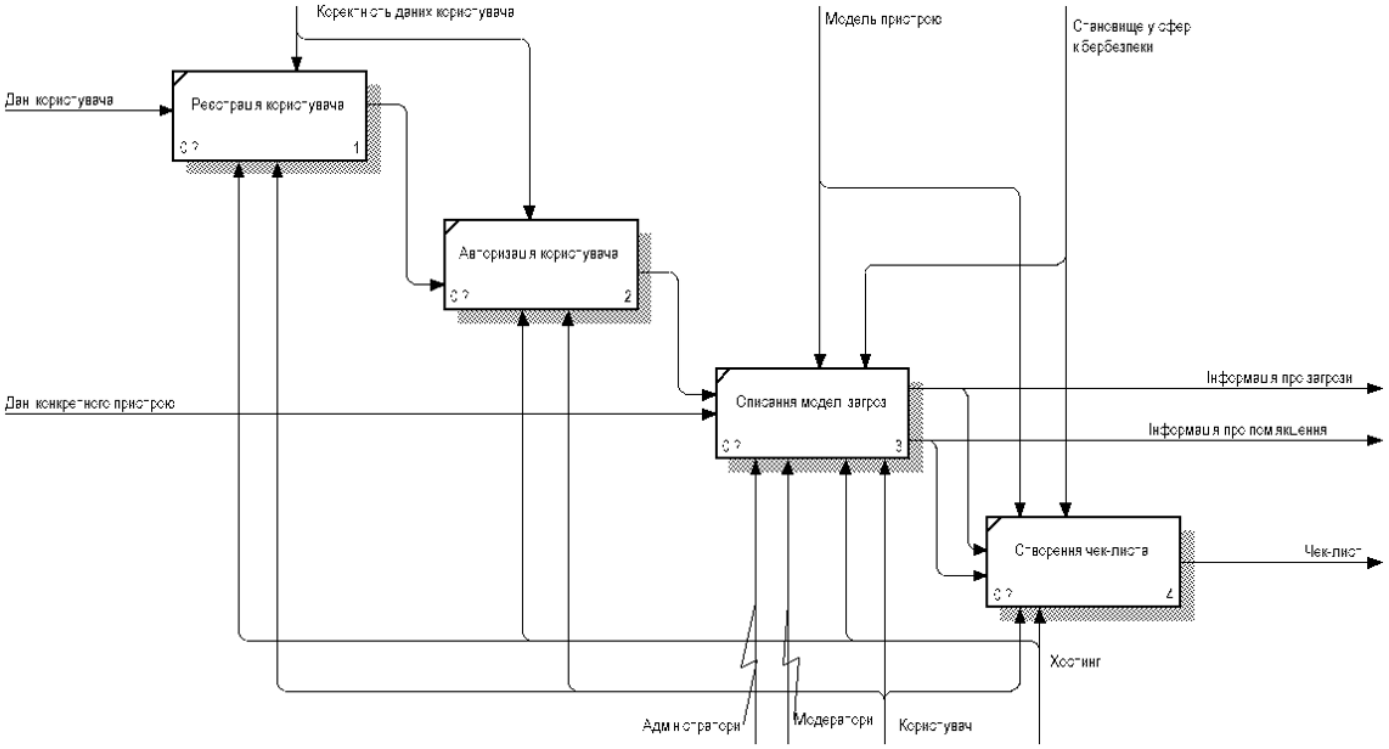
ВИМОГИ ДО ВЕБЗАСТОСУНКУ ДЛЯ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ



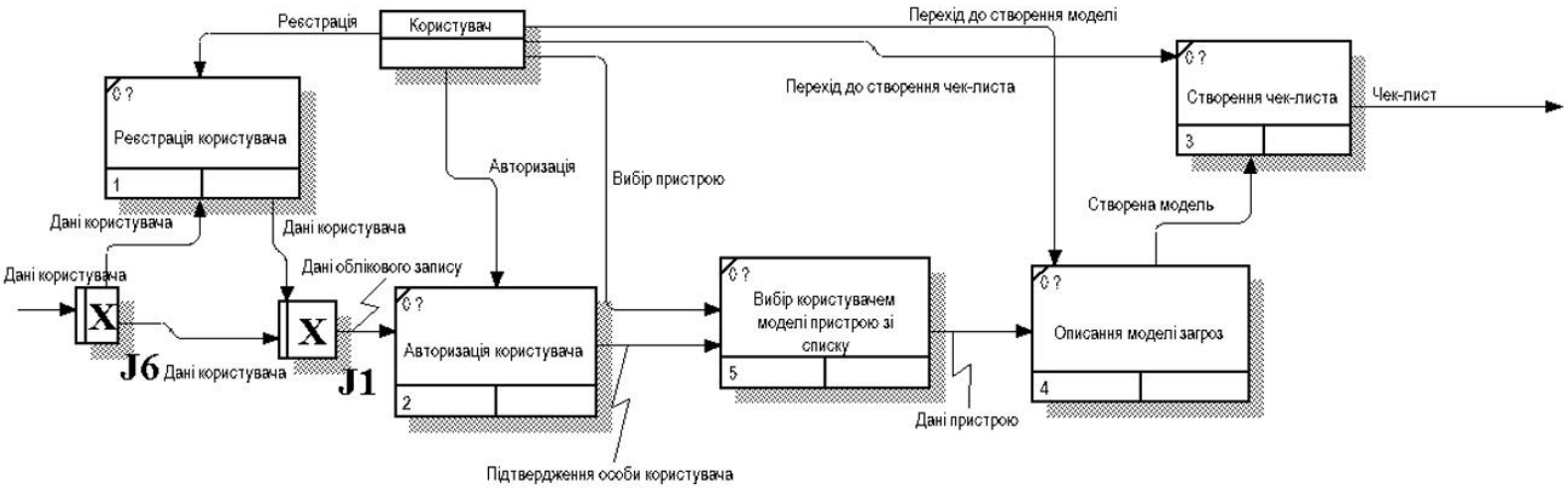
КОНЦЕПТУАЛЬНА МОДЕЛЬ ВЕБЗАСТОСУНКУ. РІВЕНЬ ПРЕДСТАВЛЕННЯ ФУНКЦІЙ



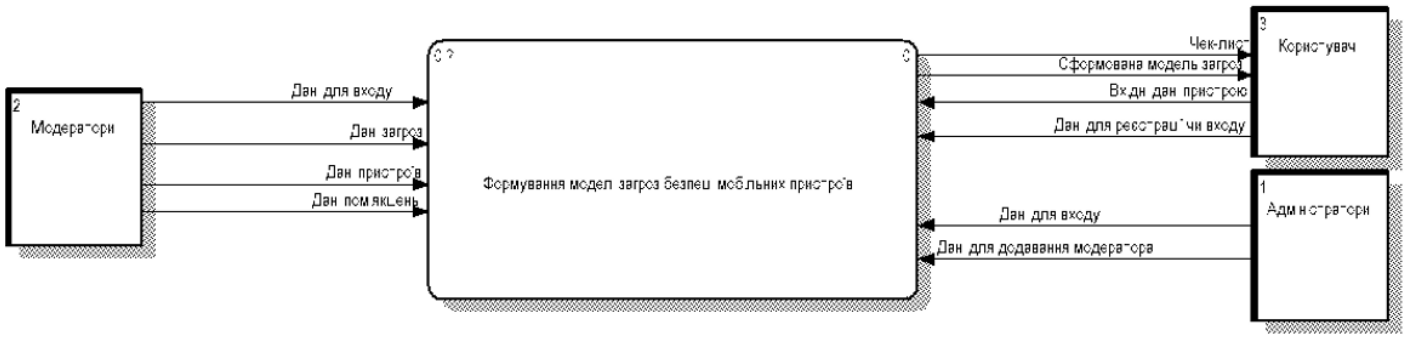
ДЕКОМПОЗИЦІЯ ФУНКЦІОНАЛЬНОЇ МОДЕЛІ ВЕБЗАСТОСУНКУ



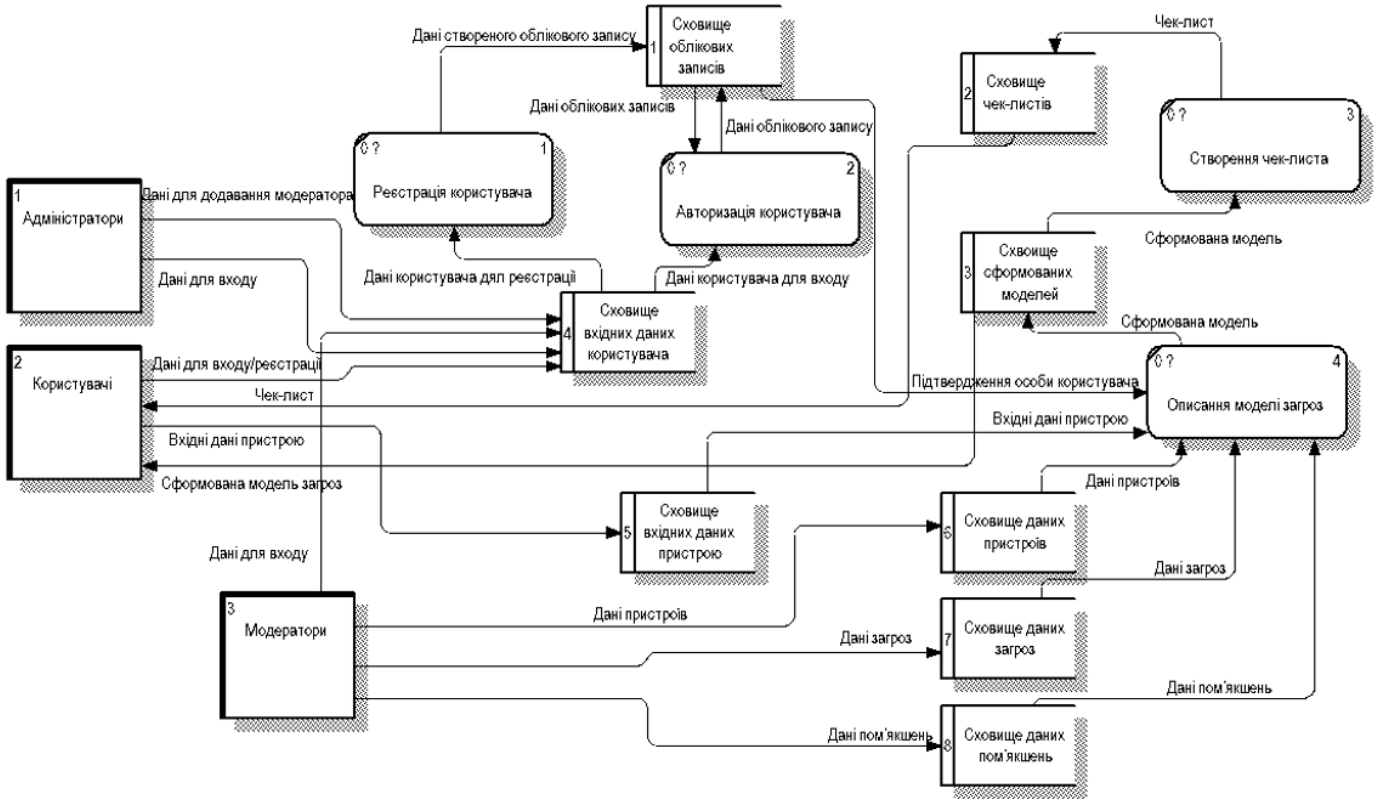
КОНЦЕПТУАЛЬНА МОДЕЛЬ ВЕБЗАСТОСУНКУ. РІВЕНЬ ПРЕДСТАВЛЕННЯ ПРОЦЕСІВ



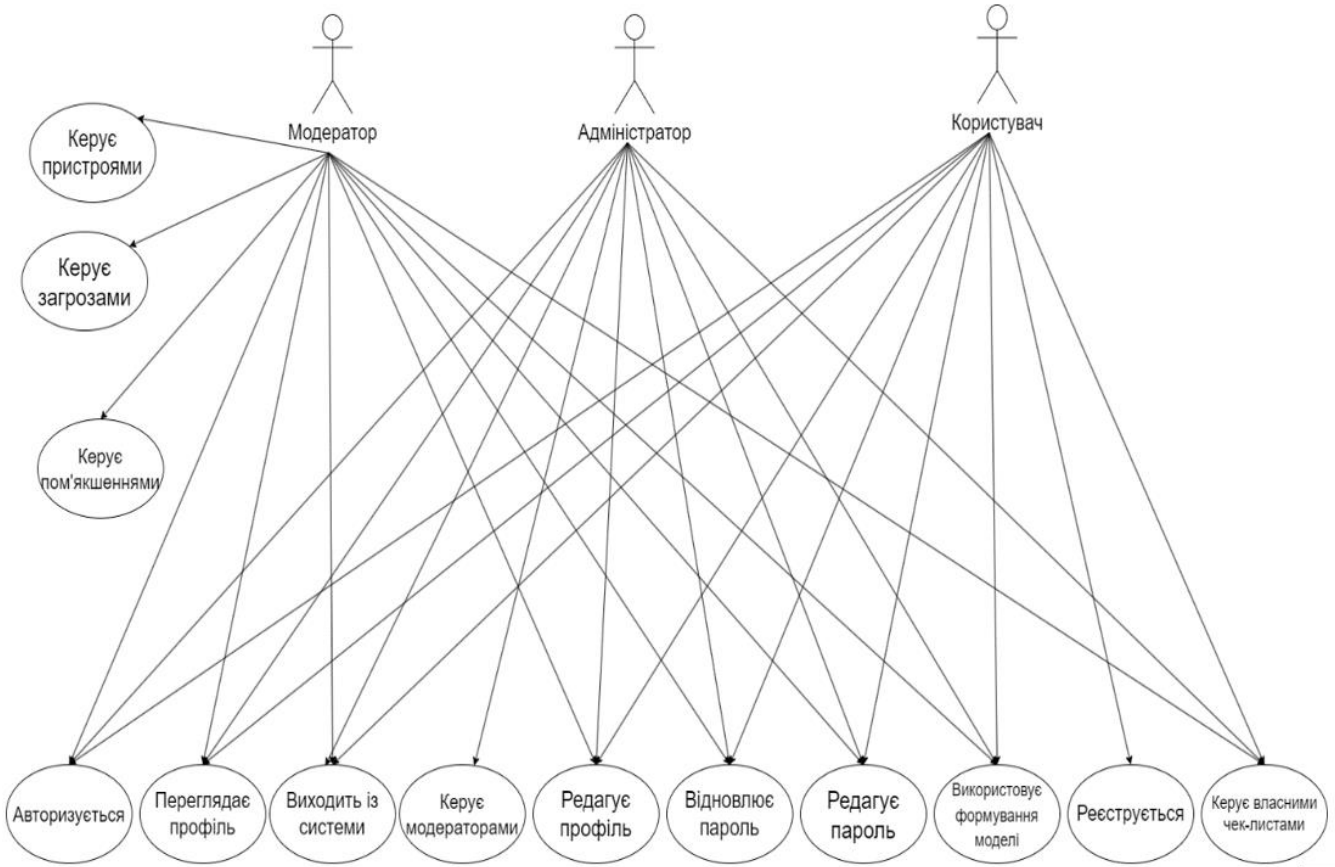
КОНЦЕПТУАЛЬНА МОДЕЛЬ ВЕБЗАСТОСУНКУ. РІВЕНЬ ПОТОКІВ ДАНИХ



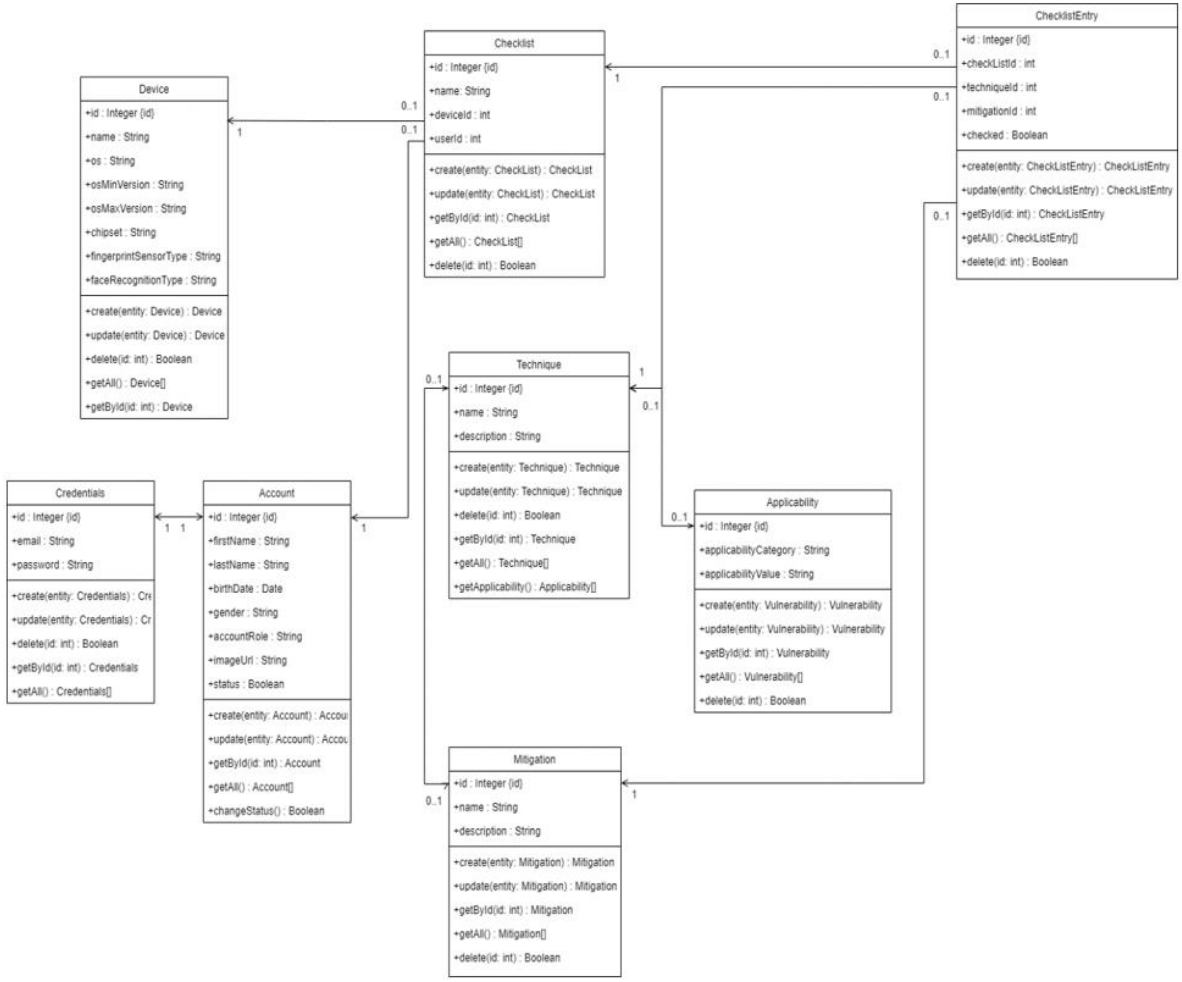
ДЕКОМПОЗИЦІЯ МОДЕЛІ ПОТОКІВ ДАНИХ ВЕБЗАСТОСУНКУ



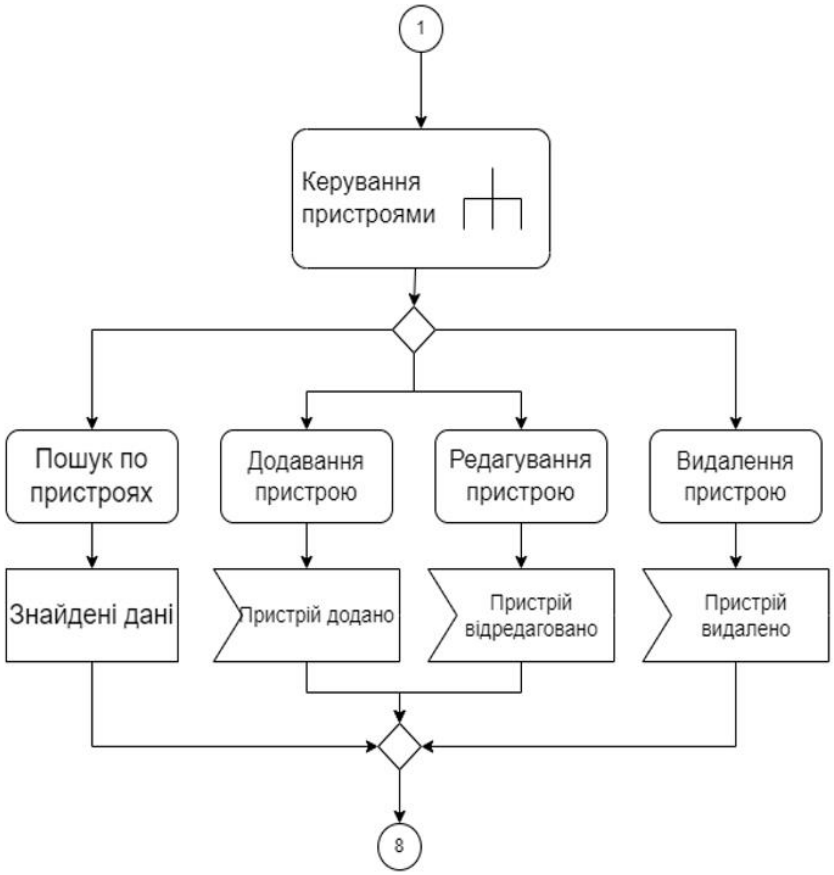
ВАРІАНТИ ВИКОРИСТАННЯ ВЕБЗАСТОСУНКУ



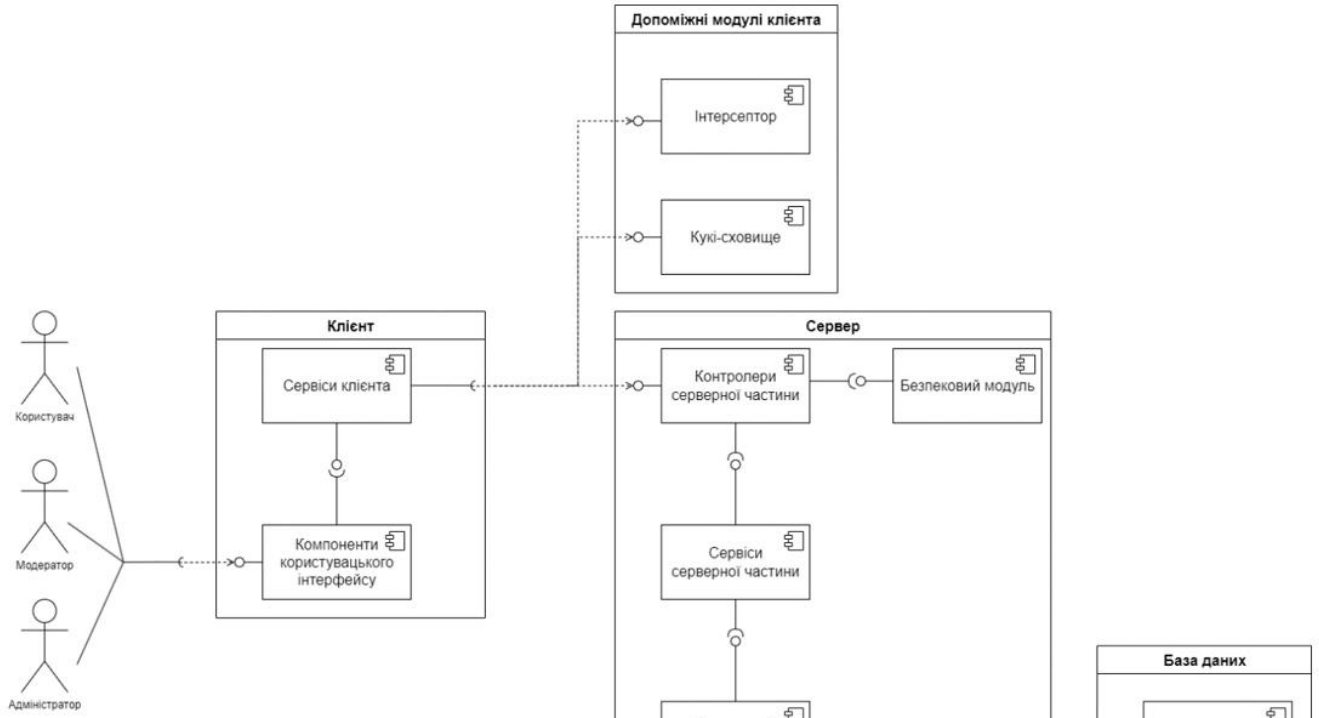
СТАТИЧНА ЛОГІЧНА СТРУКТУРА ВЕБЗАСТОСУНКУ



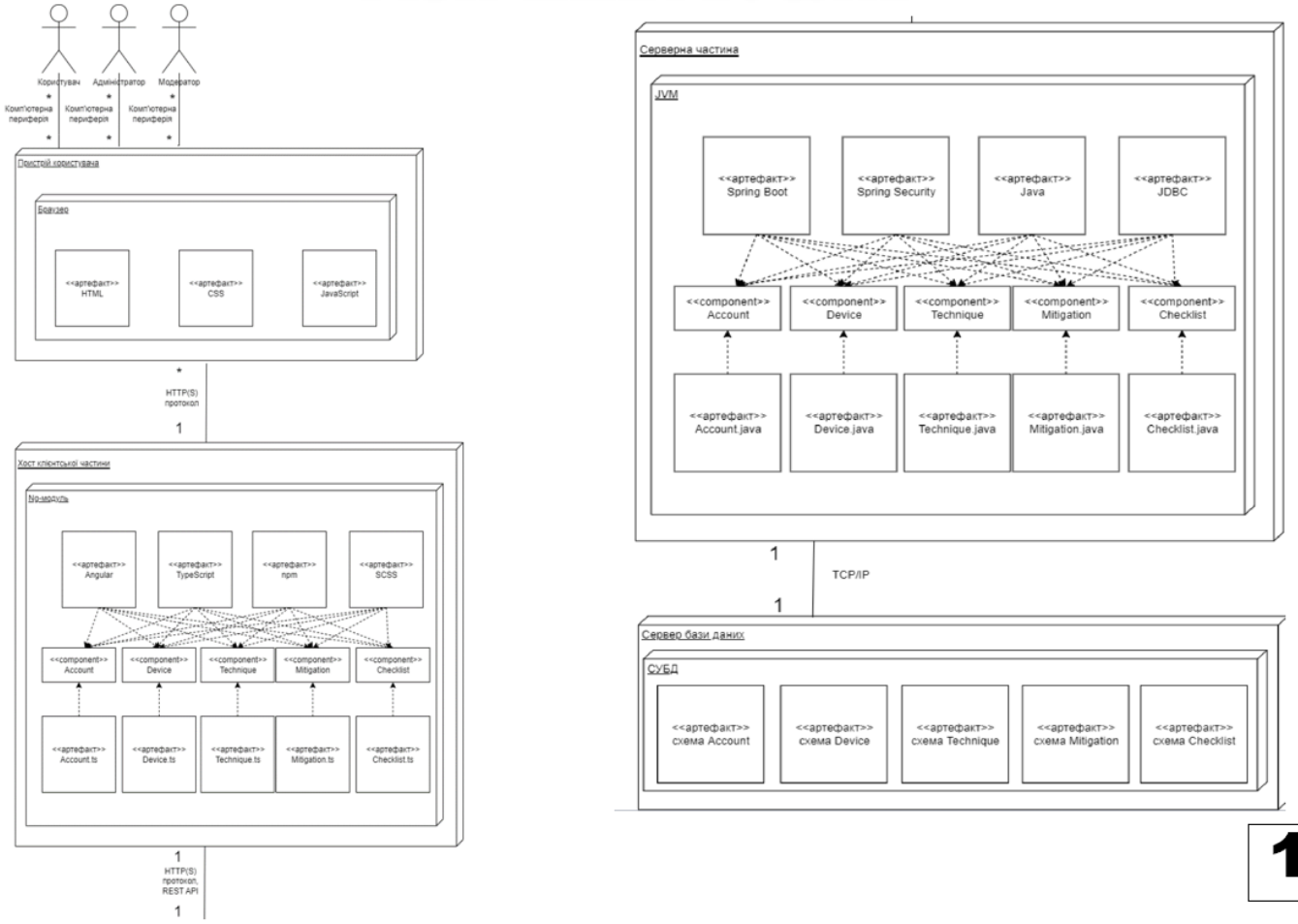
ФРАГМЕНТ ДИНАМІЧНОЇ ЛОГІЧНОЇ СТРУКТУРИ ВЕБЗАСТОСУНКУ



ФІЗИЧНА СТРУКТУРА ВЕБЗАСТОСУНКУ. ВІЗУВАЛІЗУВАННЯ ЗАГАЛЬНОЇ СТРУКТУРИ

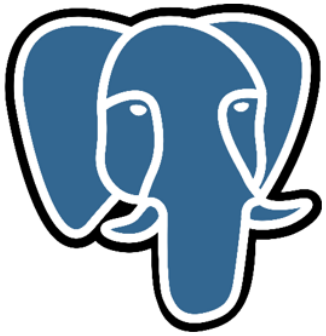


ФІЗИЧНА СТРУКТУРА ВЕБЗАСТОСУНКУ. ВІЗУАЛІЗУВАННЯ ФІЗИЧНИХ ВУЗЛІВ

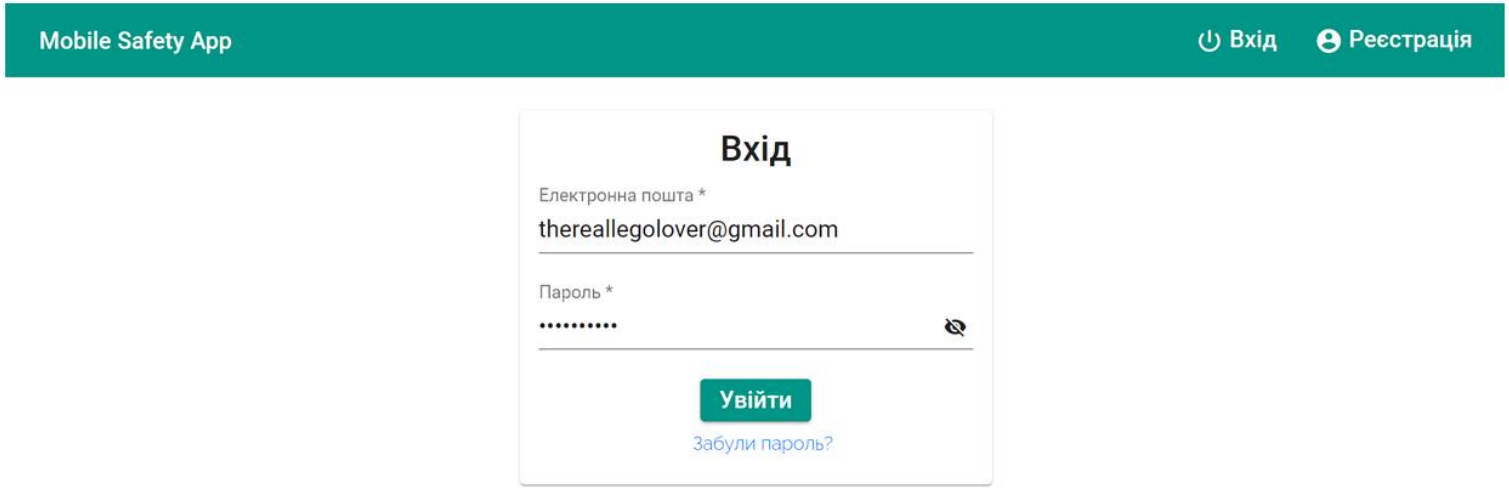


ЗАСОБИ ПРОГРАМНОГО РЕАЛІЗУВАННЯ ВЕБЗАСТОСУНКУ

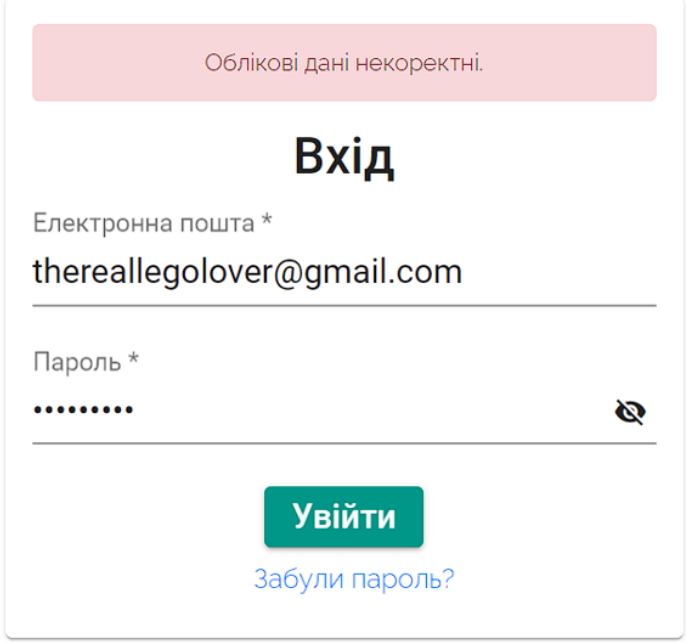
1. Мови програмування:
   - Java;
   - TypeScript.
2. Фреймворки:
   - Spring;
   - Angular.
3. Сховище даних:
   - PostgreSQL.



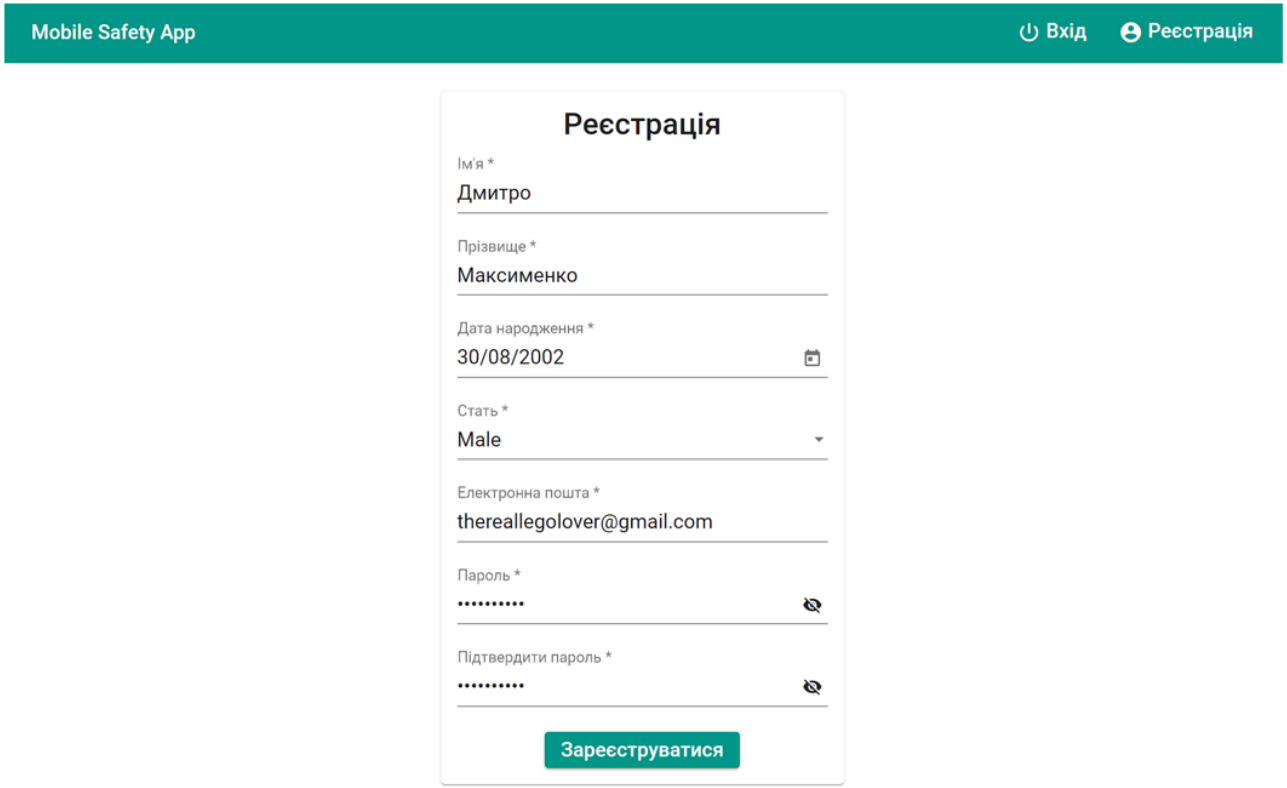
СКРИНШОТ 1.
СТОРІНКА АВТОРИЗАЦІЇ



СКРИНШОТ 2. ПРИКЛАД ПОМИЛКИ ПРИ АВТОРИЗАЦІЇ



СКРИНШОТ 3. ПРИКЛАД РЕЄСТРАЦІЇ



СКРИНШОТ 4. ПРИКЛАД ФОРМУВАННЯ МОДЕЛІ

Пристрій *

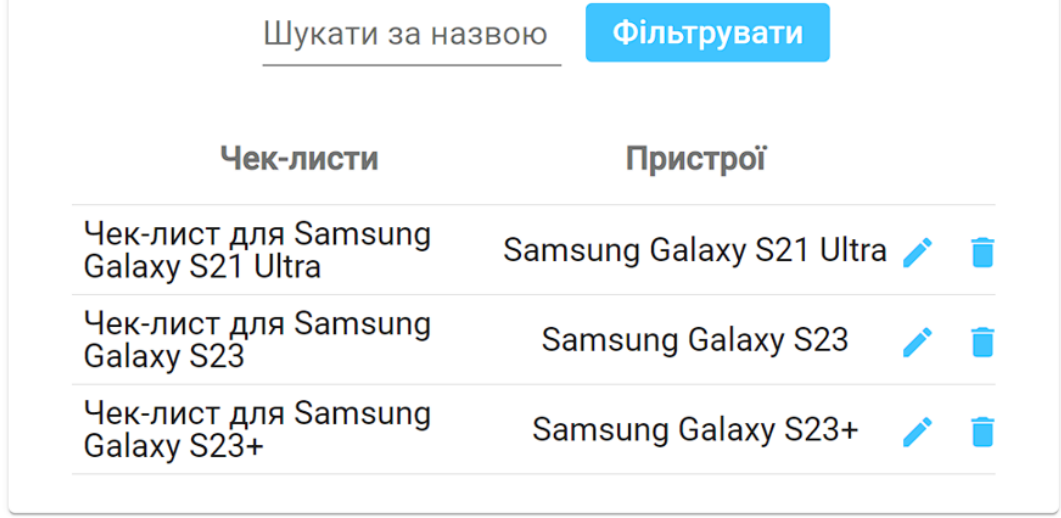
Samsung Galaxy S23+

Сформувати модель

Загрози	Пом'якшення
Розблокування фотографією	Двофакторна автентифікація або вимкнення
Доступ до пристрою без пароля	Додати пароль до пристрою

СКРИНШОТ 5. ПРИКЛАД ПЕРЕГЛЯДУ ДЕТАЛЕЙ В МОДЕЛІ

СКРИНШОТ 6. ПРИКЛАД ПЕРЕГЛЯДУ ОСОБИСТИХ ЧЕК-ЛИСТІВ



СКРИНШОТ 7. ПРИКЛАД ПЕРЕГЛЯДУ ЧЕК-ЛИСТА

Чек-лист для Samsung Galaxy S23+

Для пристрою: Samsung Galaxy S23+

Загрози	Пом'якшення	
Розблокування фотографією	Двофакторна автентифікація або вимкнення	☐
Доступ до пристрою без пароля	Додати пароль до пристрою	☑

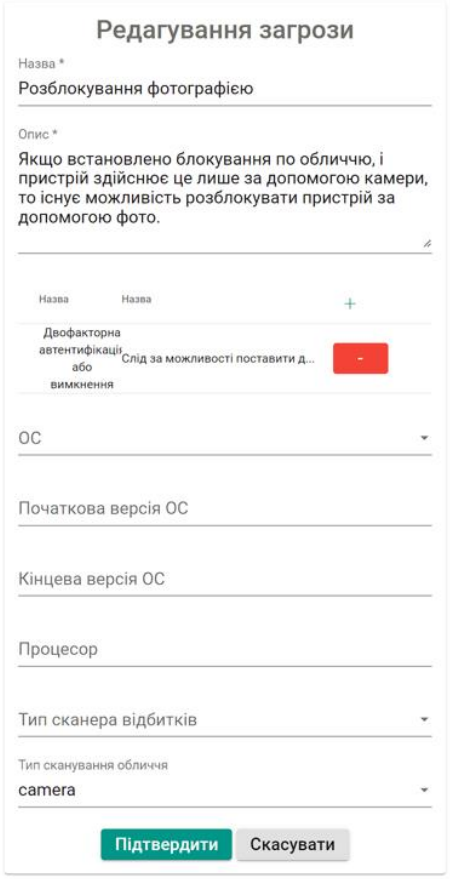
СКРИНШОТ 8. ПРИКЛАД ПЕРЕГЛЯДУ ПРИСТРОЇВ

Шукати за назвою | Фільтрувати

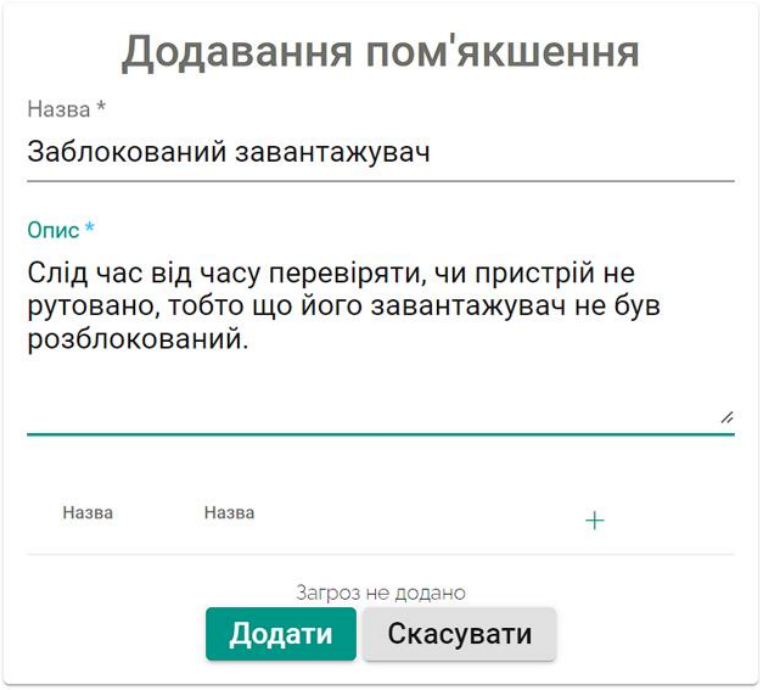
Назва	Операційна система	Мінімальна версія ОС	Максимальна версія ОС	Процесор	Сканер відбитків	Розпізнавання обличчя	+
Apple iPhone 14	iOS	16	21	Apple A15 Bionic			
Apple iPhone 14 Plus	iOS	16	21	Apple A15 Bionic			
Apple iPhone 14 Pro	iOS	16	21	Apple A16 Bionic		SL 3D	
Apple iPhone 14 Pro Max	iOS	16	21	Apple A16 Bionic		3D TOF	
Samsung Galaxy S21 Ultra	Android	11	15	Exynos 2100	ultrasonic	infrared	
Samsung Galaxy S23	Android	13	17	Qualcomm Snapdragon 8 Gen 2	ultrasonic	camera	
Samsung Galaxy S23 Ultra	Android	13	17	Qualcomm Snapdragon 8 Gen 2	ultrasonic	camera	
Samsung Galaxy S23+	Android	13	17	Qualcomm Snapdragon 8 Gen 2	ultrasonic	camera	
Xiaomi 13	Android	13	16	Qualcomm Snapdragon 8 Gen 2	optical	camera	
Xiaomi 13 Pro	Android	13	16	Qualcomm Snapdragon 8 Gen 2	optical	camera	

Items per page: 12 1 – 10 of 10

СКРИНШОТ 9. ПРИКЛАД РЕДАГУВАННЯ ЗАГРОЗИ



СКРИНШОТ 10. ПРИКЛАД СТВОРЕННЯ ПОМ'ЯКШЕННЯ



ВИСНОВКИ

1. Проаналізовано програмні застосунки для формування моделі загроз безпеці мобільних пристроїв. Виявлено їхні переваги та недоліки. На їхній основі встановлено вимоги до вебзастосунку та специфіковано діаграмою у нотації SysML.
2. Формалізовано діяльність формування моделі загроз безпеці мобільних пристроїв, описано процеси та потоки даних шляхом концептуального моделювання вебзастосунку у нотаціях IDEF0, IDEF3, DFD.
3. Побудовано об'єктно-орієнтовану модель вебзастосунку для формування моделі загроз безпеці мобільних пристроїв. Нею відображено варіанти використання, його логічна та фізична структури.
4. З урахуванням п. 2 і 3 програмно реалізовано, протестовано та продемонстровано вебзастосунок для формування моделі загроз безпеці мобільних пристроїв.

РЕЗУЛЬТАТИ ПЕРЕВІРКИ ДИПЛОМНОГО ПРОЄКТУ СИСТЕМОЮ АНТИПЛАГІАТУ UNICHECK

UNICHECK

Ім'я користувача:
Цуркан Василь Васильович ред.

ID перевірки:
1015457945

Дата перевірки:
06.06.2023 13:27:33 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2023 13:32:04 EEST

ID користувача:
76910

Назва документа: **Дипломний проєкт_Максименко Д.Ю. (ПЗ)**

Кількість сторінок: 87 **Кількість слів:** 13537 **Кількість символів:** 102652 **Розмір файлу:** 2.43 MB **ID файлу:** 1015117376

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

0.54%
Схожість

Найбільша схожість: 0.12% з джерелом з Бібліотеки (ID файлу: 1015083094)

Дякую за увагу!

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем
забезпечення комп'ютерних систем

ЗАТВЕРДЖЕНО
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

Програма та методика тестування

ДП.045440-04-51

ПОГОДЖЕНО

Керівник проєкту:

_____ Василь ЦУРКАН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро МАКСИМЕНКО

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебзастосунок для формування моделі загроз безпеці мобільних пристроїв. Призначений для формування моделі загроз, яка включає загрози та заходи безпеки, на основі моделі пристрою та передбачає додавання чек-листів на основі сформованих моделей загроз.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено:

- справність застосунку як такого в цілому;
- працездатність застосунку в реальних умовах;
- коректність роботи функціональних можливостей;
- перевірку введених даних;
- оброблення помилок;
- відсутність критичних недоліків у розробленому програмному забезпеченні.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом димного тестування. Димне тестування складається з простих тестів, які перевіряють основну функціональність вебзастосунку. Це такі тести, які мають здійснюватися швидко і мають на меті перевірити на справність основних можливостей вебзастосунку, коректність їхньої роботи та відповідність очікуваним результатам.

Таке тестування може бути корисним одразу після внесення значних змін до програмного коду застосунку для того, щоб упевнитися, що вебзастосунок загалом працює коректно.

Використовується ручне тестування, тобто таке тестування, яке здійснюється безпосередньо людиною шляхом взаємодії з розробленим застосунком та наочної перевірки коректності його роботи.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Справність застосунку перевіряється з використанням таких засобів:

- веббраузер Google Chrome та його розробницька панель, у якій можна переглянути консоль, у якій виводяться помилки, взаємодію через мережу, тобто, в даному випадку, запити до сервера та відповіді від нього;
- веббраузер Microsoft Edge та його розробницька панель;
- консоль клієнтської частини, куди виводяться помилки;
- консоль серверної частини, куди виводяться помилки.

Тестування має такий порядок:

1. Складення сценаріїв тестування.
2. Тестування роботи вебзастосунку при введенні некоректних даних облікового запису.
3. Тестування роботи вебзастосунку при введенні даних некоректного формату чи довжини.
4. Тестування роботи вебзастосунку при виникненні помилок.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖЕНО

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ ФОРМУВАННЯ МОДЕЛІ ЗАГРОЗ БЕЗПЕЦІ МОБІЛЬНИХ ПРИСТРОЇВ

Керівництво користувача

ДП.045440-05-34

ПОГОДЖЕНО

Керівник проєкту:

_____ Василь ЦУРКАН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро МАКСИМЕНКО

2023

ЗМІСТ

1. ОПИС СТРУКТУРИ ВЕБЗАСТОСУНКУ

Вебзастосунок призначений для формування моделі загроз безпеці мобільних пристроїв. Характеризується розподілом на ролі серед користувачів, що зумовлено потребою наявності можливості не лише власне формування моделі загроз, а й керування інформацією в системі, аби мати можливість легко підтримувати її в актуальному стані та доповнювати.

Тож вебзастосунок передбачає три ролі для авторизованих користувачів: звичаний користувач, модератор та адміністратор.

Кожна сторінка вебзастосунку складається з двох головних елементів: панелі меню та вмісту сторінки. Вміст сторінки змінюється залежно від дій користувача та переходом між сторінками. Тоді як панель меню залежить лише від того, авторизований користувач чи ні, та його ролі, тому переважно вона лишається статична.

Панель меню містить кнопку з назвою додатку, яка переходить на основну сторінку, кнопки для переходу на сторінки з основними функціями застосунку та кнопки базових функцій – перегляду профілю/виходу або реєстрації/входу залежно від того, чи авторизований користувач. Залежно від розміру екрану кнопки сторінок основних та базових функцій або розташовані в ряд на самій панелі (рис. 1), або містяться у випадному меню, яке відкривається при натисканні на три риски у правому верхньому куті (рис. 2).

Рис. 1. Панель меню на великому екрані

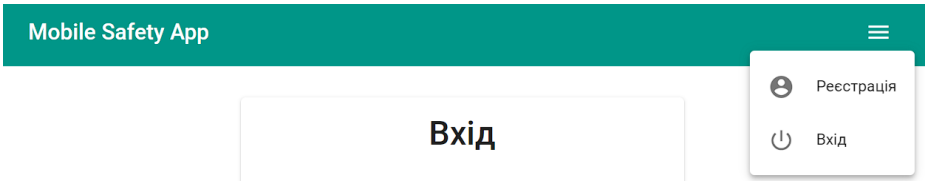


Рис. 2. Панель меню на малому екрані

Отже, для неавторизованого користувача передбачено такі сторінки:

- сторінка входу, яка доступна відразу при відкритті вебзастосунку, або з панелі меню;
- сторінка реєстрації, яка доступна з панелі меню;
- сторінка відновлення паролю, яка доступна зі сторінки входу.

Для авторизованого користувача незалежно від ролі доступні такі сторінки:

- сторінка формування моделі загроз, доступна з панелі меню;
- сторінка чек-листів, доступна з панелі меню;
- сторінка перегляду конкретного чек-листа, доступна зі сторінки чек-листів;
- сторінка профілю, доступна з панелі меню.

Для модератора додатково доступні такі сторінки:

- сторінка пристроїв, доступна з панелі меню;
- сторінка загроз, доступна з панелі меню;
- сторінка додавання загрози, доступна зі сторінки загроз;
- сторінка редагування загрози, доступна зі сторінки загроз;
- сторінка пом’якшень, доступна з панелі меню;
- сторінка додавання пом’якшення, доступна зі сторінки пом’якшень;
- сторінка редагування пом’якшення, доступна зі сторінки пом’якшень.

Для адміністратора додатково до сторінок для всіх авторизованих користувачів доступна:

- сторінка модераторів, доступна з панелі меню.

Додаткові сторінки:

- сторінка відновлення паролю;
- сторінка створення паролю для модератора.

2. ОПИС СТОРІНОК ДЛЯ НЕАВТОРИЗОВАНОГО КОРИСТУВАЧА

Перша сторінка, яка відкривається неавторизованому користувачеві, це сторінка входу (рис. 3). Аби увійти, слід ввести електронну пошту та пароль облікового запису та натиснути кнопку увійти. Для того, щоб перейти на сторінку реєстрації, слід натиснути на кнопку реєстрації у правому верхньому куті. Аби перейти на сторінку відновлення паролю, слід натиснути на посилання під кнопкою входу. Аби повернутися на сторінку входу, слід натиснути на кнопку входу в правому верхньому куті.

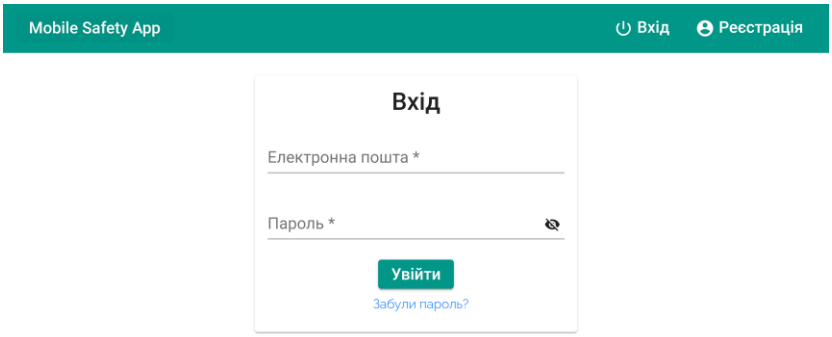


Рис. 3. Сторінка входу

Для того, щоб зареєструватися (рис. 4) слід заповнити всі поля коректними даними.

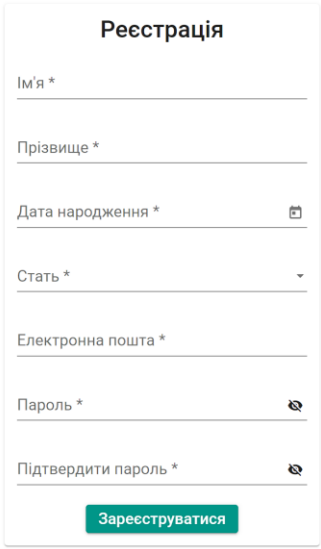


Рис. 4. Сторінка реєстрації

Ім’я має мати щонайменше 2 символи, прізвище також, електронна пошта має мати формат електронної пошти, а пароль має містити щонайменше 8 символів, одну велику літеру, одну малу літеру та цифру, також обидва паролі мають збігатися. У випадку успішної реєстрації буде виведено сповіщення та відбудеться автоматичний перехід до сторінки входу.

Для відновлення паролю слід на сторінці відновлення (рис. 5) ввести електронну пошту, до якої прив’язаний обліковий запис, та натиснути кнопку.

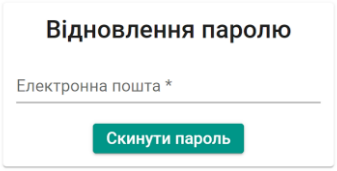


Рис. 5. Сторінка відновлення паролю

На вказану пошту буде відправлено листа (про що буде виведено сповіщення як на рис. 6), перейшовши на посилання з якого відкриється сторінка, де треба буде вказати новий пароль і підтвердити (рис. 7).

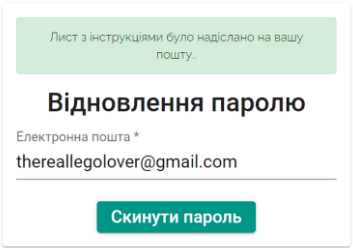


Рис. 6. Сповіщення про лист для відновлення паролю

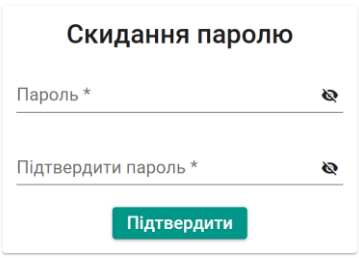


Рис. 7. Форма скидання паролю

3. ОПИС СТОРІНОК ДЛЯ БУДЬ-ЯКОГО АВТОРИЗОВАНОГО КОРИСТУВАЧА

Перша сторінка, яка відкривається після авторизації, це сторінка профілю (рис. 8). З неї можна за кнопками внизу перейти на сторінку редагування та сторінку зміни паролю.

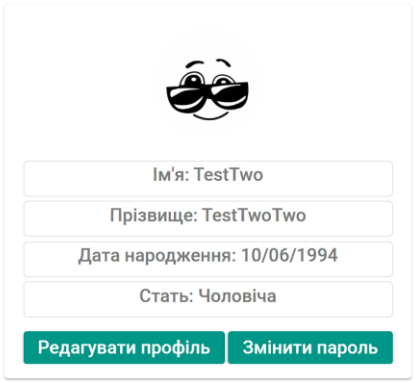


Рис. 8. Сторінка профілю

На сторінці редагування профілю (рис. 9) слід ввести дані, які відповідають реєстраційним обмеженням і натиснути кнопку збереження. Кнопка «Назад» скасовує редагування.

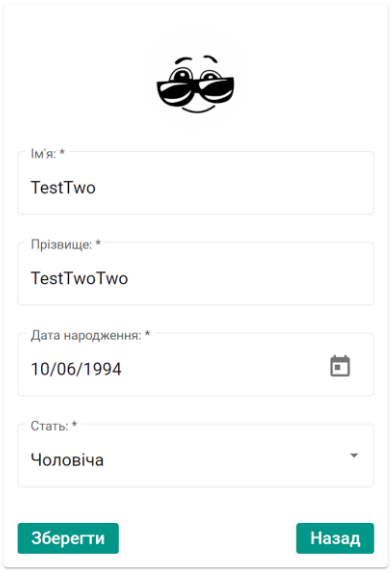


Рис. 9. Сторінка редагування профілю

Для зміни паролю на сторінці зміни паролю (рис. 10) слід ввести старий пароль, та двічі новий, і натиснути кнопку «Зберегти». Якщо пароль буде успішно змінено, виведеться сповіщення. Кнопка «Назад» скасовує зміну.

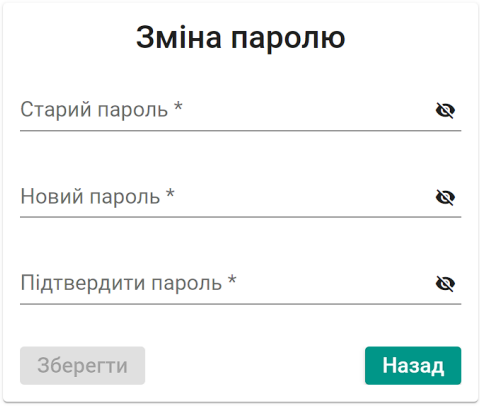


Рис. 10. Сторінка зміни паролю

Для формування моделі загроз слід перейти на сторінку формування через панель меню, почати вводити в поле вводу назву пристрою та обрати пристрій з випадного списку (рис. 11), після чого натиснути кнопку формування моделі.

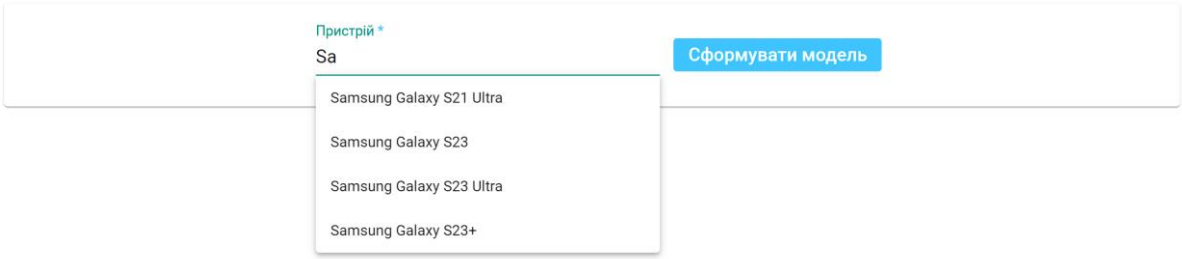


Рис. 11. Формування моделі

Для перегляду детальної інформації про загрозу (рис. 12) чи пом’якшення слід натиснути на назву загрози чи пом’якшення у сформованій моделі. Аби закрити вікно з детальною інформацією, слід натиснути на хрестик у кутку діалогового вікна, яке з’явиться.

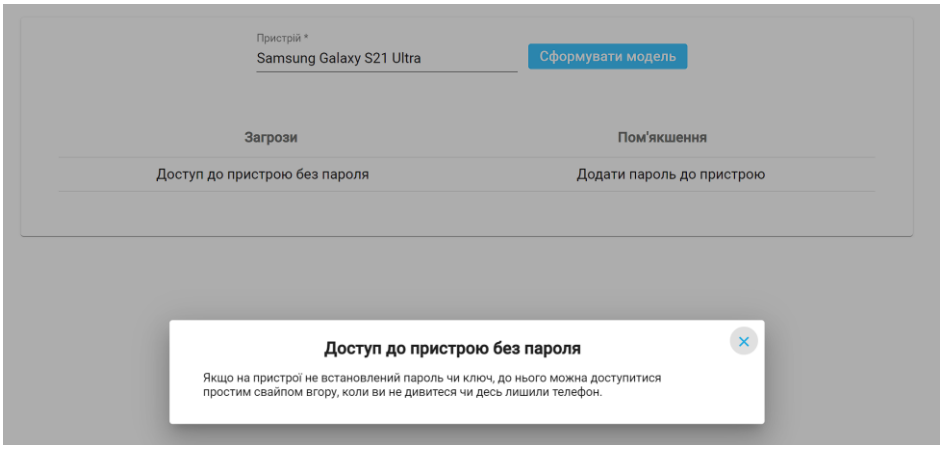


Рис. 12. Детальна інформація про загрозу

Для переходу до чек-листів слід обрати відповідний пункт на панелі меню. На сторінці чек-листів (рис. 13) доступні такі функції: щоб шукати чек-лист, потрібно ввести в поле пошуку частину назви і натиснути кнопку «Фільтрувати»; щоб видалити чек-лист, слід натиснути на іконку смітника навпроти нього; щоб перейти до редагування назви чек-листа, слід натиснути на іконку олівця навпроти нього, після чого відкриється діалогове вікно (рис. 14), де потрібно внести зміни та натиснути кнопку «Зберегти», кнопка «Закрити» скасовує зміни.

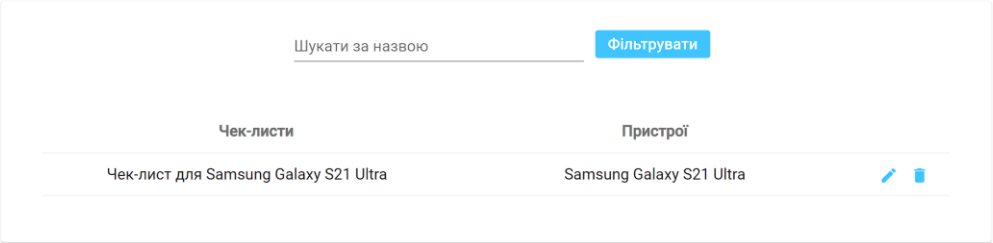


Рис. 13. Сторінка чек-листів

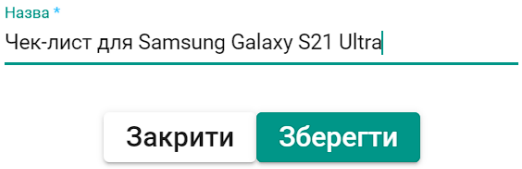


Рис. 14. Діалогове вікно зміни назви чек-листа

Якщо на сторінці чек-листів натиснути на назву чек-листа, відбудеться перехід на сторінку чек-листа (рис. 15). Справа в квадратиках можна проставляти відмітки, з кожною доданою чи знятою відміткою автоматично рухатиметься показник прогресу (лінія) вгорі.

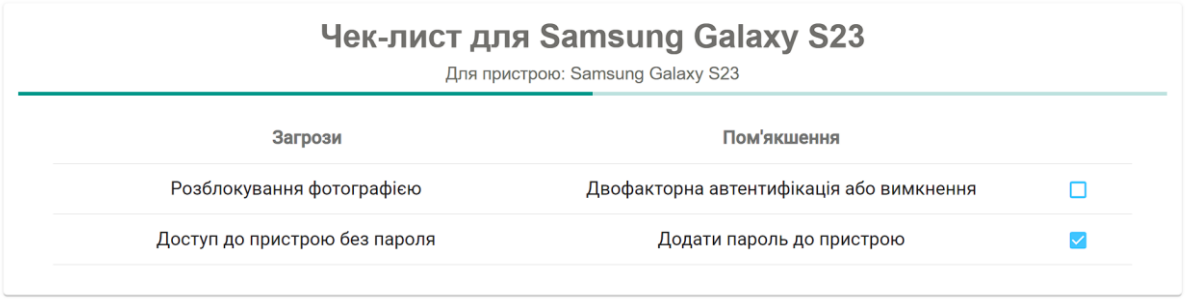


Рис. 15. Сторінка чек-листа

4. ОПИС СТОРІНОК ДЛЯ МОДЕРАТОРА

Для переходу на сторінку пристроїв (рис. 16) слід натиснути на відповідну кнопку на панелі меню. Для пошуку в полі ввести частину назви та натиснути кнопку. Для сортування за назвою слід змінювати напрям стрілки біля заголовку колонки з назвами мобільних пристроїв. Для видалення слід натиснути на іконку смітника. Для переходу на іншу сторінку слід користуватися стрілками в правому нижньому куті. Для вибору кількості записів на сторінці слід користуватися випадним списком у правому нижньому куті.

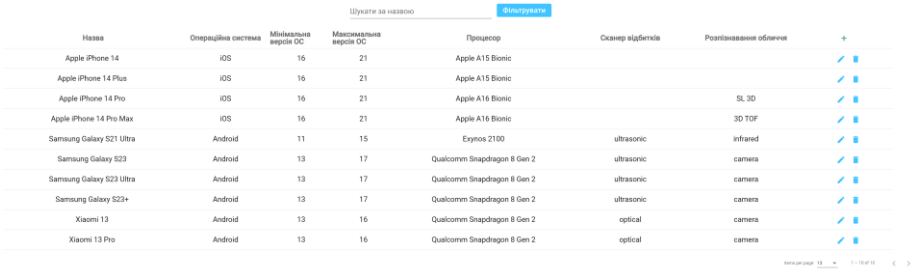


Рис. 16. Сторінка пристроїв

Якщо натиснути на олівець навпроти пристрою, з’явиться діалогове вікно редагування (рис. 17). Обов’язковими є всі поля, крім останніх двох. Для назв пристрою та процесора потрібно щонайменше 2 символи. Для збереження слід натиснути відповідну кнопку, кнопка «Закрити» скасовує зміни.

Рис. 17. Діалогове вікно редагування пристрою

Якщо натиснути на кнопку + справа вгорі, з’явиться діалогове вікно створення пристрою, повністю аналогічне вікну редагування (рис. 18).

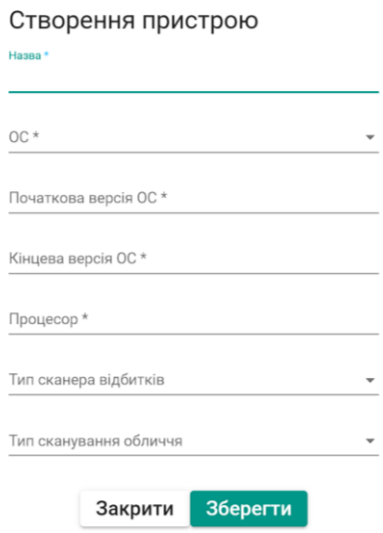


Рис. 18. Діалогове вікно створення пристрою

Для переходу на сторінку загроз (рис. 19) слід натиснути на відповідну кнопку на панелі меню. Для пошуку в полі ввести частину назви та натиснути кнопку. Для сортування за назвою слід змінювати напрям стрілки біля заголовку колонки з назвами пристроїв. Для видалення слід натиснути на іконку смітника. Для переходу на іншу сторінку слід користуватися стрілками в правому нижньому куті. Для вибору кількості записів на сторінці слід користуватися випадним списком у правому нижньому куті.

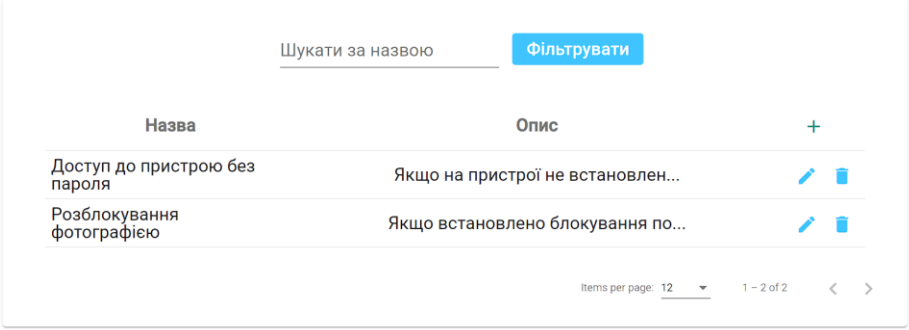


Рис. 19. Сторінка загроз

Якщо натиснути на олівець навпроти загрози, з’явиться сторінка редагування з таким полями (рис. 20).

- назва;
- опис;
- операційна система (ОС);
- початкова версія ОС;
- кінцева версія ОС;
- процесор;
- тип сканера відбитків;
- тип сканування обличчя.

Обов’язковими є поля назви та опису. Для збереження слід натиснути відповідну кнопку, кнопка «скасувати» скасовує зміни.

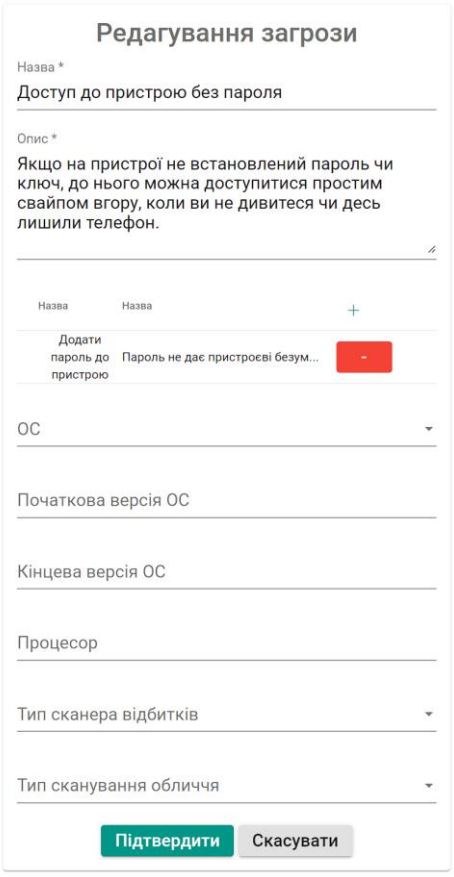


Рис. 20. Сторінка редагування загрози

Якщо натиснути на кнопку + справа вгорі, з’явиться сторінка створення загрози, повністю аналогічна сторінці редагування (рис. 21).

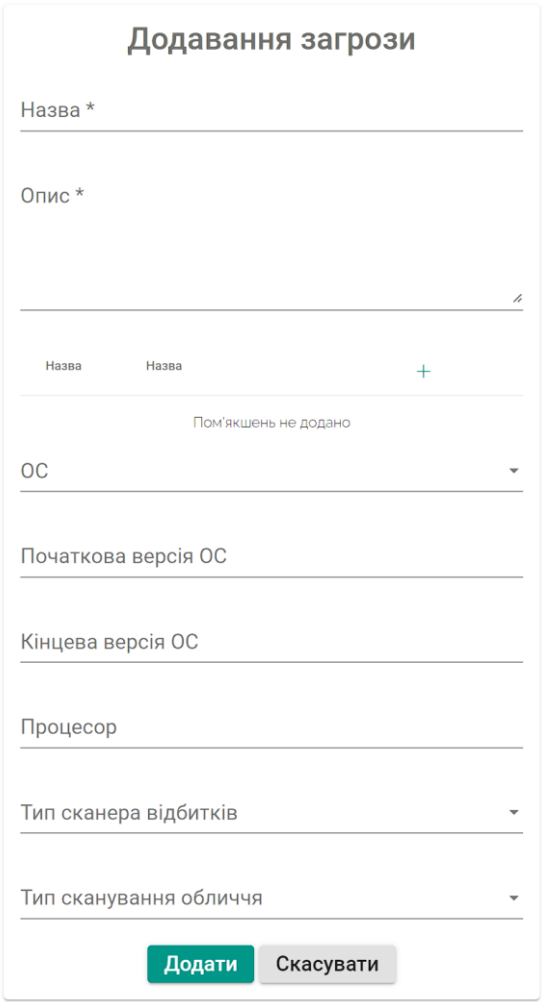


Рис. 21. Сторінка створення загрози

Для переходу на сторінку пом’якшень (рис. 22) слід натиснути на відповідну кнопку на панелі меню. Для пошуку у полі ввести частину назви та натиснути кнопку. Для сортування за назвою слід змінювати напрям стрілки біля заголовку колонки з назвами пристроїв. Для видалення слід натиснути на іконку смітника. Для переходу на іншу сторінку слід користуватися стрілками в правому нижньому куті. Для вибору кількості записів на сторінці слід користуватися випадним списком у правому нижньому куті.

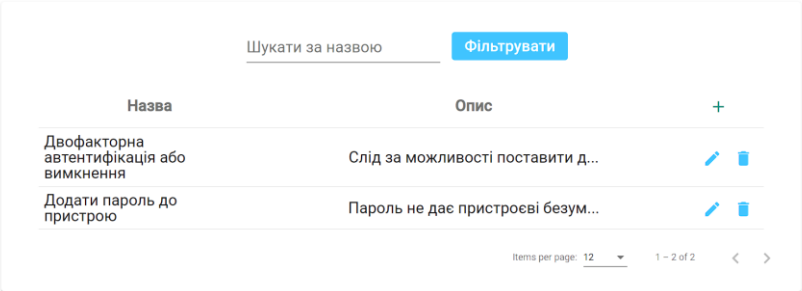


Рис. 22. Сторінка пом’якшень

Якщо натиснути на олівець навпроти пом’якшення, з’явиться сторінка редагування (рис. 23). Обов’язковими є поля назви та опису. Для збереження слід натиснути відповідну кнопку, кнопка «скасувати» скасовує зміни.

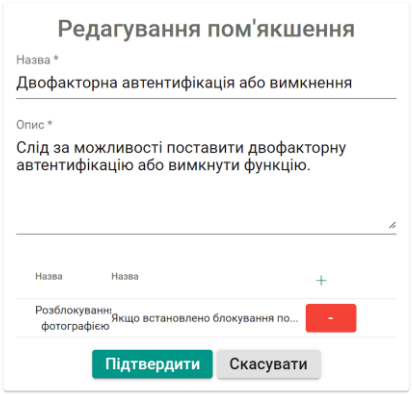


Рис. 23. Сторінка редагування пом’якшення

Якщо натиснути на кнопку + справа вгорі, з’явиться сторінка створення пом’якшення, повністю аналогічна сторінці редагування (рис. 24).

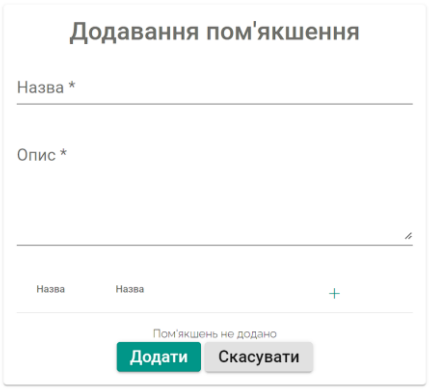


Рис. 24. Сторінка створення пом’якшення

5. ОПИС СТОРІНОК ДЛЯ АДМІНІСТРАТОРА

Для переходу на сторінку модераторів (рис. 25) слід натиснути на відповідну кнопку на панелі меню. Для пошуку заповнити пошукові поля та натиснути кнопку. Для сортування за іменем слід обрати відповідну опцію вгорі перед пошуком. Для переходу на іншу сторінку слід користуватися стрілками в правому нижньому куті. Для вибору кількості записів на сторінці слід користуватися випадним списком у правому нижньому куті.

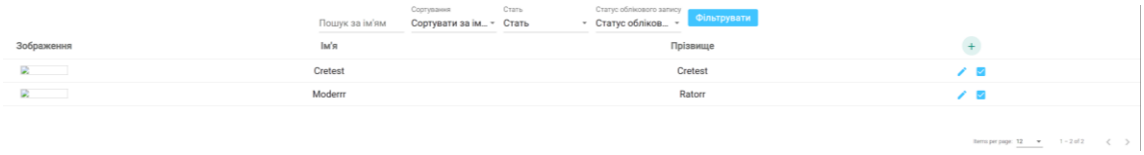


Рис. 25. Сторінка модераторів

Якщо натиснути на олівець навпроти модератора, з’явиться діалогове вікно редагування (рис. 26). Обов’язковими є всі поля. Вимоги відповідають реєстраційним вимогам, посилання на зображення має бути у форматі, який відповідає одному з основних форматів зображення. Для збереження слід натиснути відповідну кнопку, кнопка «Закрити» скасовує зміни.

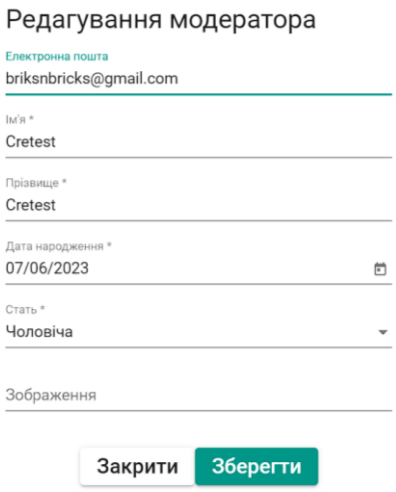


Рис. 26. Діалогове вікно редагування модератора

Якщо натиснути на кнопку + справа вгорі, з’явиться діалогове вікно створення модератора, повністю аналогічне вікну редагування (рис. 27).

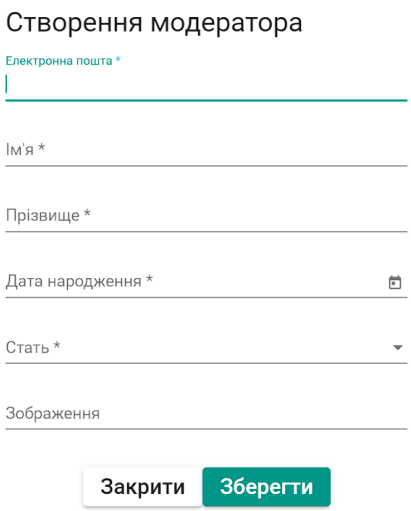


Рис. 27. Діалогове вікно створення модератора