НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ Дмитро ЛАНДЕ
“___”_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»

зі спеціальності 125 «Кібербезпека»

на тему: Керування доступом на основі атрибутів в інформаційний системах класу CRM та ERP

Виконала здобувач ступеня магістра 2 курсу, групи ФБ-01мп
(шифр групи)

_____Лях Дар’я Олександрівна _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н. доц. Коломицев Михайло Володимирович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Здобувач ступеня магістра _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію здобувачу ступеня магістра

Лях Дар'я Олександрівна
(прізвище, ім'я, по батькові)

1. Тема дисертації Керування доступом на основі атрибутів в інформаційний системах класу CRM та ERP

науковий керівник дисертації к.т.н. доц. Коломицев Михайло Володимирович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «05» ___11___ 2021 р. № 3683-с

2. Термін подання здобувачем дисертації 06.12.2021 р.

3. Об'єкт дослідження: політики керування доступом CRM та ERP систем.

4. Вихідні дані: дослідження та літературні джерела за заданою темою

5. Перелік завдань, які потрібно розробити: розглянути існуючі політики керування доступом; проаналізувати наявні політики керування доступом CRM та ERP систем; розробити власну систему атрибутів специфічну для CRM та ERP систем; розробити структуру рішення для впровадження

політики керування доступом на основі атрибутів; реалізувати рішення для впровадження політики керування доступом на основі атрибутів для Microsoft Dynamics CRM.

6. Орієнтовний перелік ілюстративного матеріалу Керування доступом на основі атрибутів в інформаційний системах класу CRM та ERP

7. Орієнтовний перелік публікацій «Керування доступом на основі атрибутів в інформаційний системах класу CRM та ERP» у науково-технічному журналі «Захист інформації».

8. Дата видачі завдання 01.09.2021 р

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Ознайомлення з літературою	1.09.2020 – 18.09.2021	
2	Збір та аналіз даних	13.09.2021 – 22.09.2021	
3	Розгляд політик керування доступом	18.09.2021 – 03.10.2021	
4	Дослідження системи керування доступом Microsoft Dynamics CRM	4.10.2021 – 10.10.2021	
5	Дослідження системи керування доступом Microsoft Dynamics AX	11.10.2021 – 18.10.2021	
6	Розробка власної системи атрибутів	19.10.2021 – 25.10.2021	
7	Розробка рішення для керування доступом на основі атрибутів в Microsoft Dynamics CRM	20.10.2021 – 09.11.2021	
8	Розробка стартап проекту	10.11.2021 – 19.11.2021	
9	Оформлення дипломної роботи	20.11.2021 – 05.12.2021	
10	Подання роботи на попередній захист	29.11.2021	
11	Подання роботи на захист	06.12.2021	

Здобувач ступеня магістра _____ Дар’я ЛЯХ
(підпис) (власне ім’я, ПРІЗВИЩЕ)

Науковий керівник _____ Михайло КОЛОМИЦЕВ
(підпис) (власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Робота обсягом 113 сторінки містить 37 ілюстрацій, 26 таблиці, 25 літературних посилань та 1 додаток.

Метою роботи є дослідження системи керування доступом інформаційних системи класу CRM та ERP та покращення існуючої системи за допомогою впровадження керування доступом на основі атрибутів.

Об’єкт дослідження: політики керування доступом CRM та ERP систем.

Предмет дослідження: забезпечення належного рівня керування доступом у CRM та ERP системах.

Методи дослідження: опрацювання літератури та інших джерел за даною темою, розгляд існуючих політик керування доступом, аналіз політик керування доступом CRM та ERP систем, розробка власної системи атрибутів для CRM та ERP систем, а також структури рішення для впровадження політики керування доступом на основі атрибутів для Microsoft Dynamics CRM та його практична реалізація.

Результати роботи можуть бути використані для впровадження рішення політики керування доступом на основі атрибутів у Microsoft Dynamics CRM та реалізації подібного рішення для ERP систем.

Ключові слова:

КЕРУВАННЯ ДОСТУПОМ, ERP, CRM, ABAC

ABSTRACT

The work includes 113 pages, 37 figures, 26 tables and 25 literary references and 1 appendix.

The purpose of this qualification work is researching the access control system for information systems in CRM and ERP class information systems and improve the existing system by implementation the attribute-based access control.

The object of research is the access control policies of CRM and ERP systems.

The subject of research is to provide an adequate level of access control in CRM and ERP systems.

Research methods: elaboration of literature and other sources on this topic, analysis of existing access control policies, analysis of access control policies for CRM and ERP systems, development of our own system of attributes for CRM and ERP systems, as well as our own solution for implementing an access control policy based on attributes for Microsoft Dynamics CRM.

The results of the work can be used to implement an access control policy solution based on Microsoft Dynamics CRM attributes and implement a similar solution for ERP systems.

Keywords:

ACCESS CONTROL, ERP, CRM, ABAC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACL – Access Control List

DAC – Discretionary Access Control

MAC – Mandatory Access Control

RBAC – Role-based Access Control

ABAC – Attribute-based access control

ERP – Enterprise Resource Planning

CRM – Customer Relationship Management

PHI – Protected Health Information

PCI – Payment Card Industry

BI – business intelligence

HR – human resources

IT – Information Technology

AAD – Azure Active Directory

NIST – National Institute of Standards and Technology

ВСТУП

Коли компанії проходять цифрову трансформацію, вони, як правило, намагаються підвищити ефективність у критичних бізнес-процесах, а також високий рівень наочності цих процесів. З усіх цих процесів є багато таких, які є фундаментальною основою того, як працює підприємство. Тому дуже важливо мати більший контроль над цими робочими процесами, ніж минулі роки.

Інвестування в такі технології, як програмне забезпечення для планування ресурсів підприємства (ERP) та програмне забезпечення управління відносинами з клієнтами (CRM), допомагає підвищити ефективність у критичних бізнес-процесах, а також високий рівень наочності цих процесів.[1]

Системи ERP і CRM наповнені конфіденційними даними та інформацією. Вони містять дані про фінанси організації, її співробітників, процеси, клієнтів та багато іншого. Якщо ці дані не будуть належним чином захищені, це може призвести до значних наслідків: втрата грошей та завдання шкоди репутації компанії.

1. *Актуальність даної кваліфікаційної роботи* випливає з росту популярності використання CRM та ERP систем. Ключовим аспектом безпеки, відповідальність за реалізацію якого лежить не на боці постачальника, а на боці клієнта є політика керування доступом. Для багатьох клієнтів є важливою можливість реалізації більш гнучких та динамічних налаштувань, які не може задовольнити існуюча система керування доступом.

З кожним роком все більше компаній замислюються про перехід до хмарних баз даних і тоді перед керівниками компаній постає питання безпеки зберігання своїх конфіденційних даних у хмарі. Особливо різко постає питання захисту інформації від зовнішніх атак.

Метою кваліфікаційної роботи є дослідження системи керування доступом інформаційних системи класу CRM та ERP та покращення існуючої системи за допомогою впровадження керування доступом на основі атрибутів.

Завдання роботи:

2. розглянути існуючі політики керування доступом;
3. проаналізувати наявні політики керування доступом CRM та ERP систем;
4. розробити власну систему атрибутів специфічну для CRM та ERP систем;
5. розробити структуру рішення для впровадження політики керування доступом на основі атрибутів;
6. реалізувати рішення для впровадження політики керування доступом на основі атрибутів для Microsoft Dynamics CRM.

Об'єкт дослідження: політики керування доступом CRM та ERP систем.

Предмет дослідження: забезпечення належного рівня керування доступом у CRM та ERP системах.

Методи дослідження: опрацювання літератури та інших джерел за даною темою, розгляд існуючих методів керування доступом, аналіз політик керування доступом CRM та ERP систем, розробка власної системи атрибутів для CRM та ERP систем, а також власної структури рішення для впровадження політики керування доступом на основі атрибутів для Microsoft Dynamics CRM та практична реалізація цього рішення.

Наукова новизна полягає в реалізації керування доступом на основі атрибутів для CRM системи на базі платформи Microsoft Dynamics 365.

Практичне значення одержаних результатів: результати даної роботи можуть бути використані в якості готового рішення по впровадженню системи керування доступом на основі атрибутів до екземпляру системи Microsoft

Dynamics CRM, а також для розробки аналогічного рішення для Microsoft Dynamics AX.

Публікації: «Керування доступом на основі атрибутів в інформаційний системах класу CRM та ERP» у науково-технічному журналі «Захист інформації».

1 АНАЛІЗ ПОЛІТИК КЕРУВАННЯ ДОСТУПОМ

1.1 Загальний огляд

Політика керування доступом – це сукупність функцій, які визначають, чи запрошена операція над спільним об’єктом (ресурсом) є законною чи ні. Іншими словами, керування доступом – це метод захисту, який класифікує авторизованих та неавторизованих користувачів та захищає спільні ресурси на основі списку контролю доступу (ACL) або політики безпеки. Політики контролю доступу використовують правила розмежування доступу, щоб визначити, який користувач може отримати які типи доступу до спільного ресурсу. Вони керують всіма правами доступу та конфліктами доступу щодо спільних ресурсів. [2]

Термін «об'єкт» відноситься до спільного ресурсу у розподіленому середовищі, над якими виконуються якісь операції. Моделі контролю доступу використовують термін «суб'єкт» для позначення користувача чи процесу, який виконується для окремого користувача або організації та здійснює операцію над об'єктом. На рис. 1.1 показаний механізм роботи моделі контролю доступу.

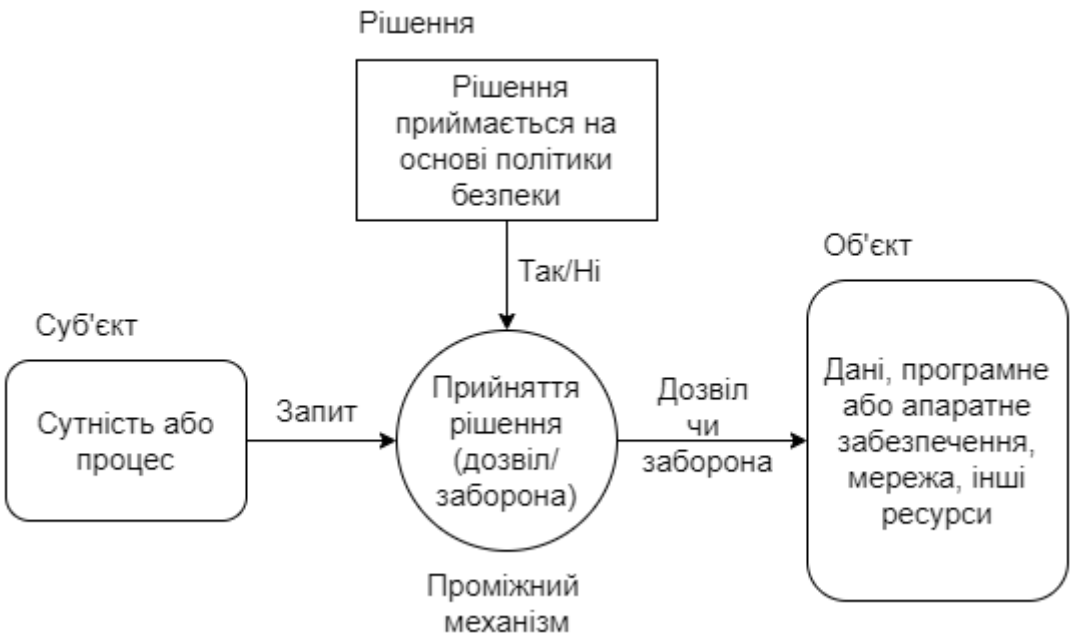

Рисунок 1.1 – Механізм роботи моделі керування доступом

Коли суб'єкти намагаються одержати доступ до об'єктів, механізми, що реалізують керування доступом, на основі політики безпеки можуть прийняти рішення про законність запиту. Використовуючи деякий набір атрибутів доступу відповідно до встановленої політики безпеки, можна реалізувати необхідне керування доступом.

Вже було розроблено досить багато моделей контролю доступу, і кожна з них має свої переваги в деяких ситуація. Ключовими механізмами є дискретний (вибірковий) (DAC – Discretionary Access Control), мандатний (MAC – Mandatory Access Control), на основі ролей (RBAC – Role-based access control) та на основі атрибутів (ABAC – Attribute-based access control).

У DAC політику доступу до об'єкта визначає користувач – власник цього об’єкта. Він надає права іншим користувачам. За такого підходу кожен керований ресурс має власника. На практиці у багатьох організаціях користувачі не володіють інформаційними ресурсами, а єдиним власником є сама організація. Також не завжди доцільно, щоб користувачі мали право

надавати доступ для інших користувачів. Таким чином, дискреційна політика не завжди коректно відображає залежності між суб'єктами та об'єктами доступу. При використанні DAC складно адмініструвати досить велику кількість ресурсів та неможливо визначати та підтримувати складні політики доступу.

У мандатній політиці доступ визначає система, а не власник ресурсу. Ця політика використовується у багаторівневих системах керування доступом, де кожен комп'ютер здатний опрацьовувати багато рівнів міток секретності, що асоційовані з ресурсами. Для отримання дозволу на доступ до ресурсу суб'єкт повинен мати рівень доступу не менший, ніж мітка, що асоційована з визначеним об'єктом безпеки. Головною проблемою, яку вирішує мандатна політика, – контроль доступу до конфіденційної інформації. MAC часто використовується у військових системах, де необхідно контролювати доступ до секретних документів та кількість рівнів секретності порівняно невелика. Для використання у промислових системах абстракція з мітками доступу не достатньо гнучка і не відображає реалій промислових бізнес-процесів.[3]

У політиці рольового керування доступом, права доступу також визначає система. Ролі у RBAC відповідає множина прав доступів до ресурсів. Механізм RBAC надає можливість визначати доступ до складних бізнес-операцій, таких як транзакцій. Користувачі системи отримують права через зв'язок з певними ролями. Керування доступом з використанням ролей матиме найбільший ефект, якщо на підприємстві багато користувачів з однаковими наборами прав доступу, саме які і зводяться до визначених ролей. Зміна прав доступу для ролі відразу змінює права для усіх асоційованих з даною роллю користувачів.[2] Керування доступом на базі ролей сьогодні є одним з найкращих підходів до керування доступом. У RBAC права доступу призначаються адміністратором вручну та є статичними.

Оскільки RBAC використовує статичні асоціації (роль-дозвіл та суб'єкт ролі), він не може задовольнити потреби безпеки обчислювальної технології, що вимагає динамічних асоціацій та незалежних від ролі політик безпеки. ABAC вирішує цю проблему і здатний виконувати динамічні відносини та генерувати незалежні від ролей політики безпеки. Політики безпеки ABAC генеруються за допомогою атрибутів категорій суб’єкт, об’єкт та середовище (час, важливість запиту тощо). І RBAC, і ABAC відповідають потребам безпеки великих програм або організацій, але ABAC відповідає складним вимогам безпеки сучасних обчислювальних технологій. Це забезпечує кращу безпеку, навіть якщо зв'язок між суб'єктами та об'єктами зростає в геометричній прогресії.[2]

Далі буде детальніше розглянуто кожну зі згаданих вище моделей керування доступом.

1.2 Дискреційна політика керування доступом

Ця модель характеризується розмежуванням доступу між певними суб'єктами та об'єктами. Суб'єкт з деяким правом доступу може наділити цим правом будь-якого іншого суб'єкта. Для кожної пари суб'єкту та об'єкту повинно бути явно задані допустимі типи доступів, такі як запис, читання, тощо, що є санкціонованими для даного суб'єкта до даного об'єкту. Кожний об'єкт системи має пов'язаний з ним суб'єкт, який є власником та встановлює права доступу до об'єкта.

У системі є тільки один обраний привілейований суб'єкт, який має можливість встановлювати права власності для всіх суб'єктів системи. Також можливі змішані варіанти реалізації системи, коли одночасно у системі присутні як власники, які можуть встановлювати права доступу до власних

об'єктів, так і привілейований суб’єкт, який може змінити права для будь-якого об'єкта, або змінити його власника. Цей випадок реалізується у багатьох існуючих операційних системах. Дискреційна модель є основною реалізації розмежованої політики доступу до ресурсів при обробці конфіденційних даних відповідно до вимог до системи захисту інформації.

Основою дискреційного керування доступом є матриця доступу. Матриця доступу – це матриця D розміру S*O, рядки якої відповідають суб’єктам, а стовпчики – об’єктам. Кожен її елемент D{s,o}=K визначає права доступу суб’єкта s до об’єкта o, де K є множиною всіх прав доступу. Кожен суб’єкт s це активна сутності, здебільшого користувач або процес. Об’єкт o – це ресурс, який потребує захисту. Наприклад, запис бази даних, сегмент оперативної пам’яті, файл.[4]

Таблиця 1.1 Приклад матриці керування доступом

Об’єкт Суб’єкт	O_1	O_2	O_3	O_4
S_1	Read			Read, Write
S_2		Read		
S_3	Read, Write		Execute	
S_4		Read, Execute		

Підмножина об’єктів, до яких заданий суб’єкт має деякі права доступу є доменом цього суб’єкта. Матриця доступу є дуже розрідженою і неефективною з погляду використання пам’яті. Тому, замість неї на практиці

використовують списки повноважень та списки доступу. Список повноважень пов'язаний з кожним суб’єктом системи й містить ідентифікатори об’єктів разом з повноваженнями цього суб’єкта щодо цих об’єктів. Список повноважень, таким чином, описує рядок матриці доступу. Список доступу навпаки пов'язаний з кожним захищеним об’єктом у системі і містить ідентифікатори різних суб’єктів з їх правами доступу до деякого об’єкту. А отже список доступу відповідає стовпчику матриці доступу.[3]

Розділяють два варіанти вибіркового керування доступом: довірче та адміністративне. У першому випадку керування доступом всі права на зміну прав доступу до об’єкта надаються суб’єктові, який є власником цього об’єкта. А отже, якщо список прав доступу суб’єкта до об’єкта містить право власника, то цей суб’єкт отримає повний контроль над стовпчиком матриці доступу цього об’єкту. При адміністративному керуванні доступом система керування визначає можливість доступу суб’єкту до об’єкту, на основі міток або атрибутів доступу, які може встановлювати або змінювати лише призначений привілейований користувач.

Переваги дискреційної політики:

- проста реалізація та широка розповсюдженість;
- зручність для користувачів — користувачі можуть самостійно керувати своїми даними;
- гнучкість — користувачі можуть налаштовувати параметри доступу до даних без адміністраторів;
- простота обслуговування — додавання нових об’єктів і користувачів не займає багато часу у адміністратора.[5]

Проте вважається, що дана політика є недосконалою через деякі важливі недоліки.

Недоліки цієї політики:

- статичні правила не враховують динамічність змін стану системи;

- при запиті доступу суб’єкта до об’єкта необхідно кожного разу визначати права доступу та аналізувати їх вплив на безпеку системи, що в свою чергу знижує її прозорість;
- не може забезпечити надійну безпеку, оскільки користувачі можуть ділитися своїми даними як завгодно;[5]
- відсутнє централізоване керування доступом, тому, щоб дізнатися параметри доступу, потрібно перевірити кожен ACL;
- загалом для систем з дискреційною політикою задача перевірки захищеності є алгоритмічно нерозв’язною. Доведення того, що система з дискреційною політикою, є захищеною у заданому стані, має проводитися для кожного стану та для кожної конкретної системи.

Дана політика безпеки вважається підходящою для використання в багаторівневих захищених військових застосунках.[5]

1.3 Мандатна політика керування доступом

Мандатна політика керування доступом (MAC) є найсуворішою з усіх політик контролю доступу. Дизайн MAC був визначений, і в основному використовується урядом.

MAC використовує ієрархічний підхід до контролю доступу до ресурсів. У MAC доступ до всіх об’єктів ресурсів контролюється параметрами, визначеними системним адміністратором. Таким чином, весь доступ до об’єктів ресурсу суворо контролюється системою на основі налаштувань системного адміністратора. Користувачі не можуть змінити контроль доступу до ресурсу відповідно до вимог MAC.[7]

Мандатна політика керування доступом починається з міток безпеки, які призначаються всім об’єктам ресурсів у системі. Ці мітки безпеки містять дві

частини інформації - класифікацію (цілком секретна, конфіденційна тощо) та категорію (яка, по суті, є вказівкою на рівень управління, відділ чи проект, для якого доступний об'єкт).

Аналогічно, кожен обліковий запис користувача в системі також має властивості класифікації та категорії з того самого набору властивостей, що застосовуються до об’єктів ресурсу. Коли користувач намагається отримати доступ до ресурсу в рамках обов’язкового контролю доступу, система перевіряє класифікацію та категорії користувача та порівнює їх із властивостями мітки безпеки об’єкта. Якщо облікові дані користувача відповідають властивостям мітки безпеки MAC, доступ до об’єкта дозволений. Важливо відзначити, що і класифікація, і категорії повинні збігатися. Наприклад, користувач із абсолютно секретною класифікацією не може отримати доступ до ресурсу, якщо він також не є членом однієї з необхідних категорій для цього об’єкта.[7]

Запобігання витокам інформації від об’єктів з високим рівнем доступу до об’єктів з низьким рівнем доступу є найважливішим завданням мандатної політики безпеки.[8] Найбільш популярним описом мандатної політики керування доступом є модель Белла-ЛаПадула (рисунок 1.2). Ця модель гарантує, що суб’єкт може прочитати з інформацією лише тоді, коли має для цього достатні повноваження, і будь-який суб’єкт (крім адміністратора, який має повноваження встановлювати рівні конфіденційності об’єктів) не може перенести дані з об’єкта з вищим рівнем конфіденційності у об’єкт з нижчим рівнем конфіденційності.[8]

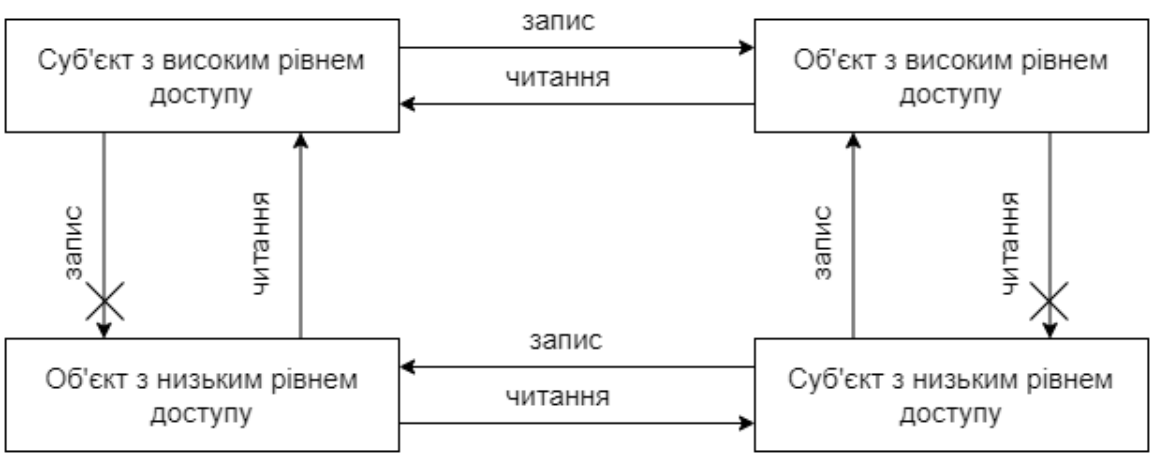

Рисунок 1.2 – Модель Белла-ЛаПадула

Переваги мандатної політики:

- прозорість правил у порівнянні з правилами дискреційної політики. Системи, побудовані на основі мандатної політики є більш надійними в порівнянні з системами, побудованими на дискреційній політиці безпеки.
- у загальному випадку для систем з мандатною політикою задача перевірки захищеності – алгоритмічно розв’язна та захищеність системи з мандатною політикою доведена.
- адміністратор визначає доступ до об’єктів, і користувачі не можуть редагувати цей доступ, а отже користувачі не можуть розсекретити дані або надати доступ до секретних даних.[5]

Недоліки мандатної політики:

- високі вимоги до обчислювальних ресурсів та складність її практичної реалізації;
- ручне налаштування рівнів безпеки та дозволів вимагає постійної роботи адміністраторів;
- MAC не масштабується автоматично;
- користувачі повинні запитувати доступ до кожного нового фрагменту даних; вони не можуть налаштувати параметри доступу для власних даних.[5]

Дана політика безпеки частіше використовується в організаціях, які потребують підвищеного акценту на конфіденційності та класифікації даних (тобто військові установах).[6]

1.4 Рольова політика керування доступом

Основною ідеєю управління доступом на основі ролей є ідея про зв'язуванні дозволів доступу до ролей, котрі призначаються кожному користувачеві.[8] По суті, RBAC призначає дозволи певним ролям в організації. Потім користувачам призначають цю конкретну роль. Наприклад, бухгалтеру в компанії буде призначено роль бухгалтера, який отримає доступ до всіх ресурсів, дозволених для всіх бухгалтерів у системі. Аналогічно, на роль розробника може бути призначений інженер-програміст.[7] Такий принцип керування доступом є складовою багатьох сучасних комп'ютерних систем. Як правило, такий підхід застосовується в системах захисту СУБД.[9] Встановлення ролей дозволяє визначити більш чіткі та зрозумілі для користувачів комп'ютерної системи правила керування доступом.

Рольовий підхід часто використовується в системах для користувачів, для яких чітко визначені їх посадові повноваження та обов'язки. Незважаючи на те, що роль є сукупністю прав доступу на об'єкти системи, рольове керування доступом аж ніяк не є окремим випадком дискреційного керування доступом, так як його правила визначають порядок надання доступу суб'єктам комп'ютерної системи в залежності від наявних (або відсутніх) у нього ролей в кожен момент часу, що є характерним для систем мандатного керування доступом.[9] З іншого боку, правила керування доступом на основі ролей є більш гнучкими, ніж при мандатному підході. Так як привілеї не призначаються безпосередньо користувачам і отримуються ними тільки через

роль (або ролі), керування індивідуальними правами користувача по суті зводиться до призначення йому ролей. Це спрощує такі операції, як додавання користувача або зміна користувачем підрозділу.[9]

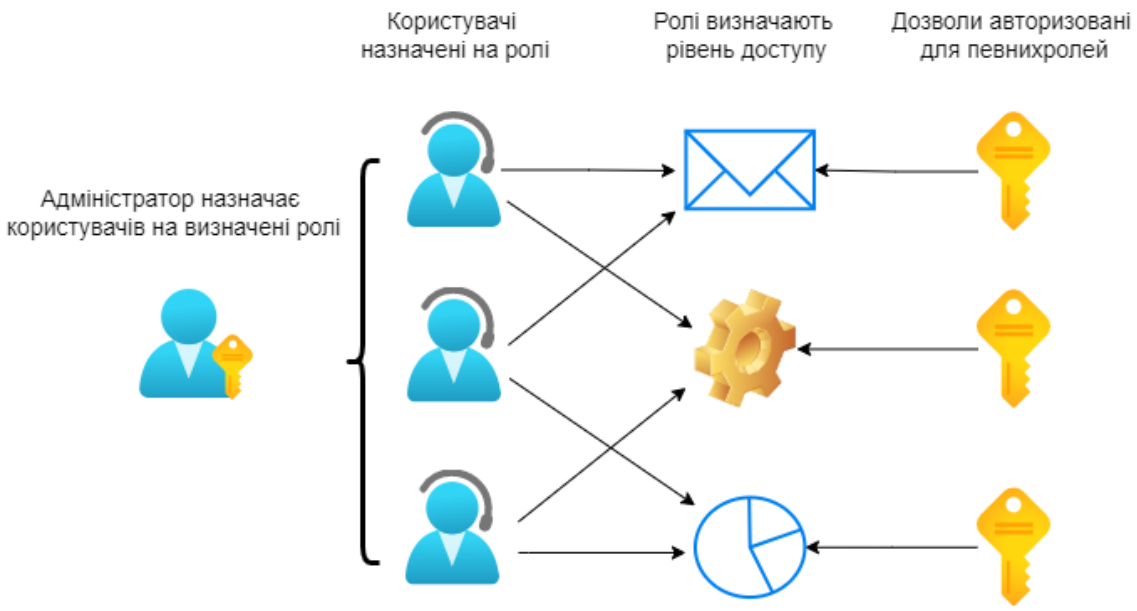

Рисунок 1.3 – Структура рольової політики керування доступом

Переваги рольової моделі:

- Зменшення адміністративної роботи та IT-підтримки. За допомогою RBAC ви можете зменшити потребу в паперовій роботі та зміні пароля, коли працівник приймається на роботу або змінює свою роль. Замість цього, ви можете використовувати RBAC для швидкого додавання та перемикання ролей та глобальної реалізації їх у операційних системах, платформах та додатках. Це також зменшує ймовірність помилок під час призначення дозволів користувачам.
- Максимізація операційної ефективності. Замість того, щоб намагатися адмініструвати контроль доступу нижнього рівня, усі ролі можуть бути узгоджені з організаційною структурою бізнесу, і користувачі зможуть виконувати свою роботу більш ефективно та автономно.

- Покращення відповідності. Усі організації підпадають під дію федеральних, державних та місцевих правил. Завдяки системі RBAC компанії можуть легше виконувати законодавчі та нормативні вимоги щодо конфіденційності, оскільки IT-відділи та керівники мають можливість керувати тим, як дані доступні та використовуються. Це особливо важливо для медичних та фінансових установ, які керують великою кількістю конфіденційних даних, таких як PHI та PCI.[10]

Недоліки рольової політики безпеки:

- У великих організаціях, де велика кількість користувачів, множина прав доступу яких часто змінюються і залежать від поточних задач, які виконуються. Керування доступом з використанням RBAC у даному випадку призводить до накопичення кількості ролей, якими важко адмініструвати;
- У разі додавання до поточної системи нових додаткових підсистем кількість ролей буде зростати в арифметичній прогресії, що в майбутньому унеможливить ефективне адміністрування;
- Зміна змісту ролі вимагає корекції її визначення. Ці функції мали б виконувати бізнес-працівники, які приймають рішення, видають задачі для виконання. Але для корекції дозволів доступів до ресурсів потрібно значні технічні знання, якими бізнес-працівники не володіють. Тому для підтримки RBAC потрібна команда технічних працівників – системних адміністраторів, кількість яких буде збільшуватися із зростанням складності системи;[9]
- З часом права доступу для співробітників буде розширюватися, адже під час виконання нових задач будуть додаватися нові права. Зворотний процес (вилучення прав) зазвичай не виконується або виконується із запізненням. В результаті працівники отримують необґрунтовано великі права доступу, що зменшує загальний рівень захищеності системи та суперечить принципу мінімальних прав доступу.[3]

1.5 Політика керування доступом на основі атрибутів

Списки керування доступом прості, але зі зростанням кількості користувачів та об’єктів, до яких можна отримати доступ, разом із ними зростає і кількість записів списку ACL. Якщо у вас мільйон користувачів і мільйон об’єктів, то в гіршому випадку ви можете отримати мільярд записів ACL із переліком індивідуальних дозволів кожного користувача для кожного об’єкта. Хоча такий підхід може працювати з меншою кількістю користувачів, з ростом бази користувачів він стає ще більшою проблемою. Ця проблема особливо погана, якщо дозволи централізовано управляються системним адміністратором, а не визначаються окремими користувачами. Якщо дозволи не знімаються, коли вони більше не потрібні,

Хоча RBAC є дуже успішною політикою контролю доступу, яка широко розповсюджена, у багатьох випадках бажана політика контролю доступу не може бути виражена простими призначеннями ролей. Розглянемо приклад агента кол-центру. Окрім того, щоб не дозволити агенту отримати доступ до записів клієнтів поза межами встановленого ними робочого часу, ви також можете заборонити їм доступ до цих записів, якщо вони насправді не здійснюють дзвінок із цим клієнтом. Дозволити кожному агенту отримувати доступ до всіх записів клієнтів у робочий час - це ще більше прав, ніж він дійсно потребує для виконання своєї роботи, що порушує принцип найменших привілеїв. Можливо, ви можете визначити, з яким клієнтом розмовляє агент дзвінків, за його номером телефону (ідентифікатор абонента), або, можливо, клієнт вводить номер рахунку за допомогою клавіатури перед тим, як з’єднатися з агентом. Ви хотіли б дозволити агенту доступ лише до файлу цього клієнта протягом усього часу розмови, можливо, дозволивши йому через п’ять хвилин після того, як він пройде, завершити написання будь-яких

нотаток. Для обробки таких видів рішень щодо динамічного контролю доступу була розроблена альтернатива RBAC, відома як ABAC: контроль доступу на основі атрибутів.

Згідно NIST «ABAC — це методологія логічного контролю доступу, де авторизація на виконання набору операцій визначається шляхом оцінки атрибутів, пов’язаних із суб’єктом, об’єктом, запитуваними операціями та, в деяких випадках, умовами середовища щодо політики, правил чи відносин, які описують допустимі операції. для заданого набору атрибутів».[12] Отже в ABAC рішення щодо контролю доступу приймаються динамічно для кожного запиту, використовуючи колекції атрибутів, згрупованих у чотири категорії:

- Атрибути про суб’єкт; тобто користувач, який робить запит: приклади атрибутів - роль, ідентифікатор, статус, членство у командах, групах, членство у відділах та організаціях, посада тощо.
- Атрибути ресурсу або об’єкта, до якого здійснюється доступ, наприклад, URI ресурсу або мітка безпеки.
- Атрибути дії, яку намагається виконати користувач, наприклад оновити запис.
- Атрибути про середовище або контекст, у якому відбувається операція. Це може включати місцевий час доби або місцезнаходження користувача, який виконує дію.

Тоді вихід ABAC є рішенням дозволу або заборони.

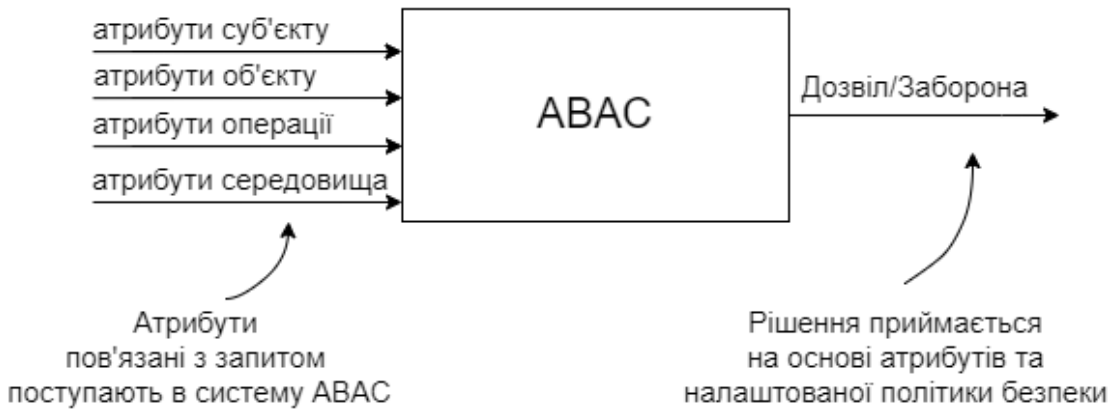

Рисунок 1.4 – Прийняття рішення у моделі на основі атрибутів

Переваги політики доступом на основі атрибутів:

- надає доступ не на основі ролі користувача, а на основі атрибутів кожного компонента системи. Таким чином можна описати бізнес-правило будь-якої складності. Навіть якщо вам потрібно зробити певні дані доступними лише в робочий час, це можна легко зробити за допомогою однієї простої політики.
- правила ABAC можуть оцінювати атрибути суб’єктів та ресурсів, які ще мають бути інвентаризовані системою авторизації.[13]
- менше адміністрування (принаймні, після його налаштування). Оскільки дозволи на доступ автоматично змінюються в міру зміни атрибутів користувача, при введенні нових користувачів стає менше адміністрування.

Недоліки:

- важко налаштувати через те, як повинні бути визначені та підтримувані політики.
- важко провести аудит і визначити дозволи, доступні для конкретного користувача.

Висновки до розділу 1

В даному розділі було проведено аналіз основних наявних політик керування доступом – дискреційної, мандатної, рольової та на основі атрибутів. Оцінено переваги та недоліки кожної з політик та складено порівняльну таблицю політик керування доступом (таблиця 1.2).

Таблиця 1.2 – Аналіз моделей керування доступом

Модель керування доступом	Керування доступом базується на	Деталізація	Гнучкість	Ефективність	Рівень безпеки
DAC	Розсуд власника даних	Добре підходить для невеликих застосунків	Висока	Слабка	Низький
MAC	Класифікації користувачів та даних	Добре підходить для невеликих застосунків	Низька	Слабка	Краще ніж DAC
RBAC	Класифікації ролей	Добре підходить для широкомасштабованих застосунків	Висока	Висока	Низький
ABAC	Оцінці атрибутів	Добре підходить для сучасних комп’ютерних технологій	Дуже висока	Висока	Високий

Якщо підвести підсумок, то у кожної з перерахованих політик є свої переваги, проте ключовим є те, що жодна з описаних політик не стоїть на місці, а динамічно розвивається. Прихильники є у кожної з них, однак, об'єктивно, важко віддати перевагу якійсь одній системі. Вони різні і служать для різних цілей.

В наступному розділі буде розглянуто існуючі системи керування доступом CRM та ERP систем на платформі Microsoft Dynamics 365.

2 АНАЛІЗ МЕХАНІЗМІВ КЕРУВАННЯ ДОСТУПОМ У ІНФОРМАЦІЙНИХ СИСТЕМАХ КЛАСУ ERP ТА CRM

2.1 Загальний огляд

Коли компанії проходять цифрову трансформацію, вони, як правило, намагаються підвищити ефективність у критичних бізнес-процесах, а також високий рівень наочності цих процесів. З усіх цих процесів є багато таких, які є фундаментальною основою того, як працює підприємство. Тому дуже важливо мати більшу видимість та більший контроль над цими робочими процесами, ніж раніше.[1]

Інвестування в такі технології, як програмне забезпечення для планування ресурсів підприємства (ERP) та програмне забезпечення управління відносинами з клієнтами (CRM), допомагає забезпечити цей контроль та видимість, і вони найкраще працюють, коли вони функціонують як згуртований підрозділ.

Системи CRM допомагають керувати даними клієнтів, покупками та контактною інформацією, аналізувати та розуміти їх для збільшення продажів. Підприємства використовують CRM для кращого розуміння своїх клієнтів для прийняття більш обґрунтованих рішень щодо продажу та взаємодії з ними. CRM система збирає інформацію з різних каналів, таких як соціальні мережі, електронні кампанії, чати, телефонні дзвінки, особиста взаємодія, форми зворотного зв’язку тощо. Вона також використовує аналітику даних для аналізу історії покупок клієнтів та їх потреб. Мета полягає в тому, щоб покращити взаємодію з клієнтами та їх задоволеність, перетворивши потенційних клієнтів у клієнтів, а потім підтримуючи та покращуючи відносини з ними. Системи CRM пов'язують різні відділи, включаючи

продажі, маркетинг, HR, обслуговування клієнтів, IT, бізнес-аналітику (BI) та фінанси. Крім того, дані легко доступні будь-якому співробітнику в режимі реального часу, що дає можливість командам приймати швидкі рішення на основі даних. [14]

Загалом, CRM-рішення зуміли об’єднати міжвідомчі команди та збільшити продажі, усунувши розрізненість бізнесу.

Там, де CRM керує клієнтами, ERP використовується для управління бізнесом. ERP — це система підвищення ефективності бізнес-процесів. Як і CRM, ERP дозволяє швидко обмінюватися стандартизованою інформацією між всіма відділами. Усі співробітники вводять інформацію до системи ERP, створюючи в реальному часі загальний знімок всього підприємства. Проблеми в будь-якій області автоматично створюватимуть сповіщення в інших постраждалих областях. Це дозволяє відділам почати вирішувати проблеми вже до того, як вони стануть проблемою в цьому відділі. Коротше кажучи, дозволяючи бізнесу зосереджуватися на даних, а не на операціях, ERP надає метод для спрощення бізнес-процесів.

Програмне забезпечення ERP має на меті спростити та централізувати бізнес-процеси в межах підприємства, що включає історію замовлень, планування та інші робочі процеси, керовані даними. Коли підприємства використовують комплексне програмне забезпечення ERP, це забезпечує легший спосіб керувати критичними частинами бізнесу.

ERP і CRM — це бізнес-додатки, які зберігають та аналізують дані в реляційній базі даних. Обидва вони поставляються або через традиційну локальну модель (on-premise), або через програмне забезпечення як послугу (SaaS), де постачальник керує програмним забезпеченням у своєму власному центрі обробки даних, а клієнти отримують доступ до нього через хмару.[15]

Хоча вся організація буде покладатися на обидві системи ERP та CRM, принципова відмінність між ERP та CRM полягає в тому, що ERP призначена

насамперед для фінансових даних та фінансового відділу, тоді як CRM — це дані про клієнтів, які використовуються відділами продажу та обслуговування клієнтів. Першу зазвичай називають бек-офіс, а другу — фронт-офіс.

Наразі ринок ERP та CRM систем дуже різноманітний. Найбільш популярними є такі ERP системи:

- Oracle ERP Cloud – найкраща програмна система для ERP-аналітики;
- Systems Analysis and Program Development (SAP) ERP – найкраща ERP для масштабованості;
- Microsoft Dynamics 365 – найкращий для користувачів платформи Microsoft;
- Delmiaworks – найкраща ERP для виробничих компаній;
- webERP – найкраща система управління роздрібною торгівлею;
- Dolibarr – найкраща ERP для сторонніх інтеграцій;
- Compiere – найкраща ERP для оптової торгівлі та розповсюдження;
- ERPNext – найкраще безкоштовне програмне забезпечення ERP;
- Odoo – найкраще програмне забезпечення ERP з відкритим кодом.[16]

Та найбільш популярні CPM системи:

- HubSpot CRM
- Microsoft Dynamics 365 CRM
- Oracle CRM
- Salesforce
- Zoho CRM

Кожна з них має свої специфічні особливості системи безпеки. Для подальшого аналізу в роботі було обрано одну з найбільш популярних та знайомих у цілому світі платформу Microsoft Dynamics 365 та її продукти Microsoft Dynamics AX та Microsoft Dynamics CRM.

2.2 Безпека систем класу CRM

Система керування доступом Microsoft Dynamics CRM заснована на рольовій моделі. Для контролю доступу до даних необхідно створити організаційну структуру, яка захищає конфіденційні дані і забезпечує спільну роботу, а саме налаштувати бізнес-підрозділи, ролі безпеки і профілі безпеки поля. Ролі безпеки визначають, які дії користувач може виконати над записом. [17]

Крім того, є кілька підходів до безпеки. Наприклад, доступ до кожного запису зазвичай базується на моделі власності. Наприклад, я можу читати лише свої облікові записи. Але існують інші варіанти, такі як ієрархічна безпека.

Інші вимоги безпеки можуть включати вимоги щодо обмеження доступу до окремих полів або, можливо, нам потрібно надати доступ до певних записів.

Ключові аспекти, на яких базується безпека в Microsoft Dynamics CRM:

- Користувачі
- Команди – об’єднують користувачів та зв’язані з ролями безпеки.
- Ролі безпеки – дозволяють визначити, який доступ буде наданий командам та користувачам. Спочатку створюються ролі, а потім одна або кілька асоціюються з кожним користувачем/командою.
- Бізнес-одиниці (business units) – фундаментальний будівельний блок у моделі безпеки Dynamics. Вони визначають структуру організації.
- Профілі безпеки поля (field security profile) – використовуються, коли потрібно визначити конкретні права доступу до окремих полів.
- Ієрархічна безпека підтримує модель безпеки на основі якоїсь ієрархії. Позиції визначають організаційні посади, які підтримують модель безпеки ієрархії.

Тепер про деякі з аспектів більш детально.

Бізнес-одиниці

Перша конструкція – це бізнес-одиниці, вони дозволяють визначити структуру вашої організації. Користувачі, команди та ролі безпеки будуть пов'язані з бізнес-одиницями, що робить їх фундаментальним будівельним блоком у моделі безпеки.

Бізнес-одиниці можуть мати ієрархію. Це може відображати фактичну ієрархію організації, але це не завжди так. Насправді ієрархія розроблена для того, щоб «керувати», які користувачі отримують доступ до яких записів.

Під час створення користувачів бізнес-одиниця є обов’язковим полем і за замовчуванням буде використовуватися як коренева бізнес-одиниця. Щоб отримати доступ до Dynamics 365, користувачеві також має бути надана принаймні одна роль безпеки або бути призначеним до команди, яка має роль безпеки.

Користувачі

Для доступу суб’єкту до системи перш за все потрібно створити йому користувача. Це означає, що він повинний існувати як користувачі в Office 365. Крім того, йому має бути призначена ліцензія Dynamics 365. Після входу користувача в Dynamics 365 модель безпеки використовується для контролю того, до чого він може отримати доступ. Це досягається за допомогою параметра керування ролями для користувача.

Ролі безпеки

Кожна роль пов'язана з бізнес-одиницею. Ролі містять багато параметрів, які регулюють доступ до сутностей/функцій для поєднання бізнес-одиниці та користувача.

Часто ролі успадковуються від материнського підрозділу. Це означає, що коли використовуються ролі керування для користувача, пов’язаного з дочірнім бізнес-підрозділом, все одно ролі з батьківського підрозділу будуть видимі.

Кожен користувач може мати кілька ролей. Модель безпеки Dynamics 365 працює на найменш обмежувальній моделі безпеки. Це означає, що замість принципу найменших привілеїв буде застосовано «сума» всіх ролей, призначених користувачеві.

Кожна роль складається з кількох параметрів (дій) для кожної таблиці/об’єкта в Dynamics 365. Таблиці зазвичай мають кілька дій, які охоплюють функції, включаючи створення, читання, запис, видалення, додавання, додавання, призначення та спільний доступ.

Кожна сутність може мати вісім різних видів дій:

- Створити;
- Читати;
- Писати;
- Видалити;
- Додати – зв’язати запис цієї таблиці до іншої таблиці;
- Додати до – додати інші записи до цієї таблиці;
- Призначити – дає можливість змінити власника запису на іншу команду чи користувача;
- Спільний доступ – дозволяє поділитися записом з іншими користувачами.

Для кожного доступу регулюється його рівень. Існує 5 рівнів доступу:

- Глобальний – доступ надається до всіх записів в організації;
- Глибокий – доступ надається до записів у бізнес-підрозділі користувача та всіх бізнес-одиниць, підпорядкованих бізнес-підрозділу користувача;
- Локальний – доступ надається до записів у бізнес-підрозділі користувача;
- Основний – доступ надається до записів, якими володіє користувач, до об’єктів, які спільно використовуються з організацією, до об’єктів, до яких

безпосередньо користувач має спільний доступ або команда, членом якої є користувач;

- Відсутній – доступ заборонено.

Окрім дій, що застосовуються до записів таблиць доступ також може надаватися до більш специфічних дій, наприклад таких як:

- Dynamics 365 для мобільних пристроїв;
- Експорт в Excel;
- Активація або деактивація користувача;
- Дії від імені іншого користувача;
- Зміна налаштувань ієрархічної системи безпеки;
- Створення експрес-кампанії;
- Групове видалення;
- Перегляд журналу відстеження;
- Публікація звітів;
- Затвердження статей бази знань;
- та багато інших.

Команди

Команди мають декілька застосувань у Dynamics 365. Команди, по суті, містять список або групи користувачів і можуть бути використані для управління безпекою. Команди також можуть бути корисними при створенні звітів. Крім того, команди можуть бути корисними, коли ви хочете поділитися «активами» з групою користувачів. Активи можуть включати представлення, діаграми, інформаційні панелі або записи.

Бізнес-одиниці та команди

Кожна компанія має команду за замовчуванням. Це «особлива» команда, яку неможливо видалити. Користувачі автоматично додаються до неї по мірі їх призначення до бізнес-одиниці.

Існує два типи команд. Команди власників – це оригінальний тип команди, знайдений у Dynamics 365. Команди власників мають членів, і команда може володіти записами. Наприклад, обліковий запис може належати «Команді міжнародних продажів», а не окремому користувачеві. Команди доступу – це особливий тип команди, що надає можливості обміну записами.

Важливо розуміти, що команди власників можуть виконувати роль безпеки. Члени команди успадкують дозволи, надані роллю(ями) безпеки команди. Якщо ви хочете, щоб записи належали команді, вона повинна мати роль, яка надає доступ до будь-якої сутності, якою вона має володіти.[18]

Ієрархія посад

Адміністратор визначає різні посади в організації та впорядковує їх в ієрархії посад. Потім додає користувачів до будь-якої даної позиції або «позначає» користувача певною позицією. Користувач може бути позначений лише однією позицією в даній ієрархії, однак позицію можна використовувати для кількох користувачів. Користувачі на вищих посадах в ієрархії мають доступ до даних користувачів на нижчих посадах. Прямі вищі позиції мають доступ для читання, запису, оновлення, додавання, додавання до даних нижчих позицій. Непрямі вищі посади, мають доступ лише для читання до даних нижчих позицій.

Профілі безпеки поля

Дозволи на рівні запису надаються на рівні таблиці, але можуть бути певні поля, пов’язані з об’єктом, які містять дані, які є більш конфіденційними, ніж інші поля. У таких ситуаціях можна використовувати безпеку на рівні полів для контролю доступу до певних полів. Безпека на рівні поля керується профілями безпеки. Налаштовуючи їх можна обмежити доступ користувачам чи командам доступ до деяких полів. Можна дозволити або заборонити права на читання, оновлення, або створення даних таких полів.

Аутентифікація

Тільки автентифіковані користувачі, які мають права користувача в програмах Customer Engagement, можуть встановити з’єднання. Вони використовують Microsoft Azure Active Directory (AAD) як основного постачальника ідентифікаційних даних. Щоб отримати доступ до системи, користувачі повинні бути підключені до екземпляра програм Customer Engagement і повинні мати дійсний обліковий запис AAD у авторизованого орендаря (tenant).

2.3 Безпека систем класу ERP

Microsoft Dynamics AX

Microsoft Dynamics AX (Axapta) – багатофункціональна ERP-система для управління ресурсами підприємства для середніх і великих компаній з розподіленою структурою, включаючи холдинги, філії підприємств, дистриб'юторські компанії, міжнародні корпорації та ін. Вона охоплює всі області менеджменту: виробництво та дистрибуцію, ланцюжки поставок і проекти, фінанси та засоби бізнес-аналізу, взаємовідносини з клієнтами та персоналом.[19] Її остання версія називається Dynamics 365 for Finance and Operations.

У Dynamics 365 for Finance and Operations можна використовувати привілеї для групування захищених об’єктів, таких як точки входу та дозволи для таблиць, форм і методів сервера. Крім того, можна об’єднати привілеї в обов’язки і обов’язки в цикли процесу. Обов’язок – це набір привілеїв доступу до програм, необхідних користувачеві для виконання своїх обов’язків. Процесний цикл – це сукупність обов'язків, які представляють бізнес-процес

більш високого рівня. Модель безпеки є ієрархічною, і кожен елемент в ієрархії представляє різний рівень деталізації. [20]

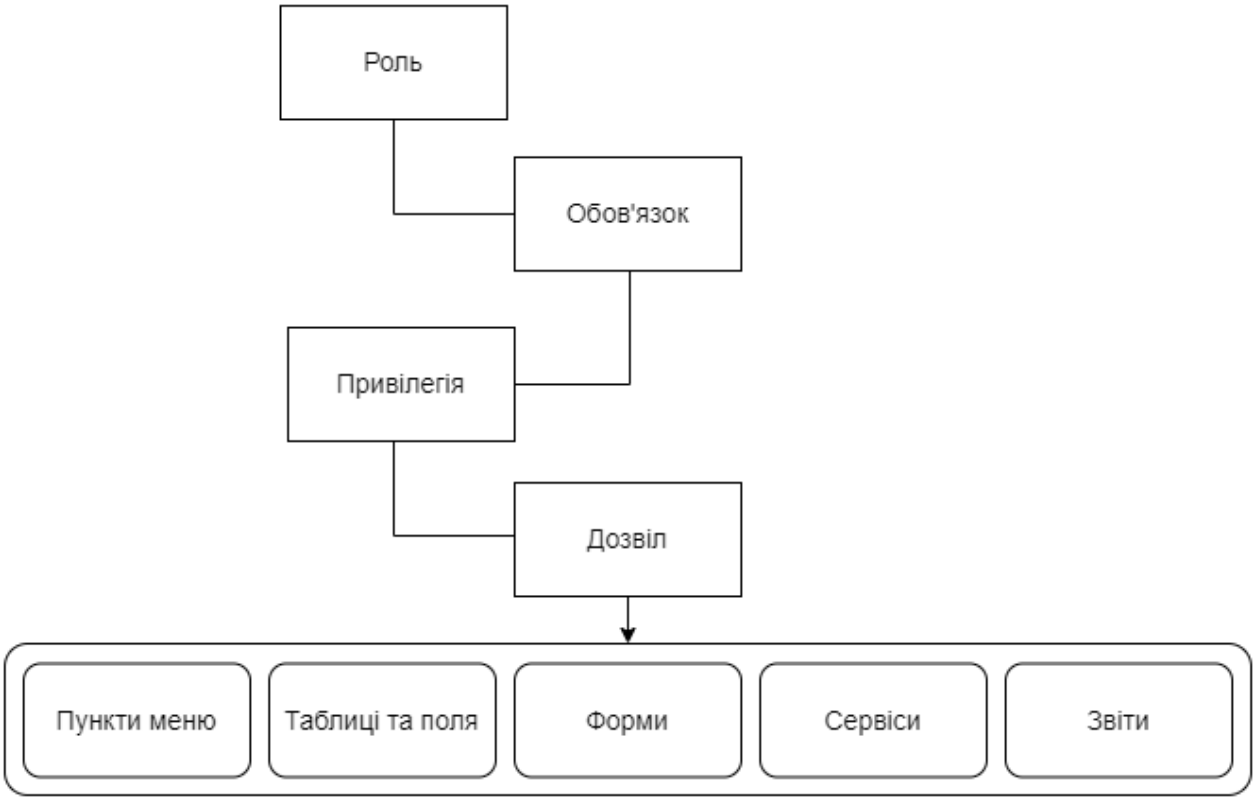

Рисунок 2.1 – Діаграма огляду архітектури безпеки Microsoft Dynamics AX

Система керування доступом Microsoft Dynamics AX заснована на рольовій моделі.[21] У безпеці на основі ролей доступ надається не окремим користувачам, а лише ролям безпеки. Доступ до системи дозволений лише авторизованим користувачам, кожен з яких повинен мати певні ролі. Для представлення різних робочих функцій можна створити ролі безпеки. Певним ролям призначається певний набір привілеїв доступу до програми. Користувачі можуть бути призначені для однієї або кількох ролей безпеки і за допомогою цих призначень ролей отримати дозволи на виконання певних системних функцій. Користувач, якому призначено роль безпеки, має доступ

до набору привілеїв, пов’язаних з цією роллю. Користувач, якому не призначено жодної ролі, не має привілеїв. Усім користувачам потрібно призначити принаймні одну роль безпеки, щоб мати доступ до системи. Ролі безпеки, які призначаються користувачеві, визначають обов’язки, які він може виконувати, і частини інтерфейсу користувача, які користувач може переглядати.

Ролі безпеки

У Dynamics 365 for Finance and Operations безпека на основі ролей узгоджена зі структурою бізнесу. Користувачам призначаються ролі безпеки на основі їхніх обов’язків в організації та їхньої участі в бізнес-процесах. Адміністратор надає доступ до обов'язків, які виконують користувачі у ролі, а не до елементів програми, які користувачі повинні використовувати.[22]

Оскільки правила можна налаштувати для автоматичного призначення ролі, адміністратор не повинен залучатися щоразу, коли відповідальність користувача змінюється. Після налаштування ролей і правил безпеки менеджери бізнесу можуть контролювати щоденний доступ користувачів на основі бізнес-даних.

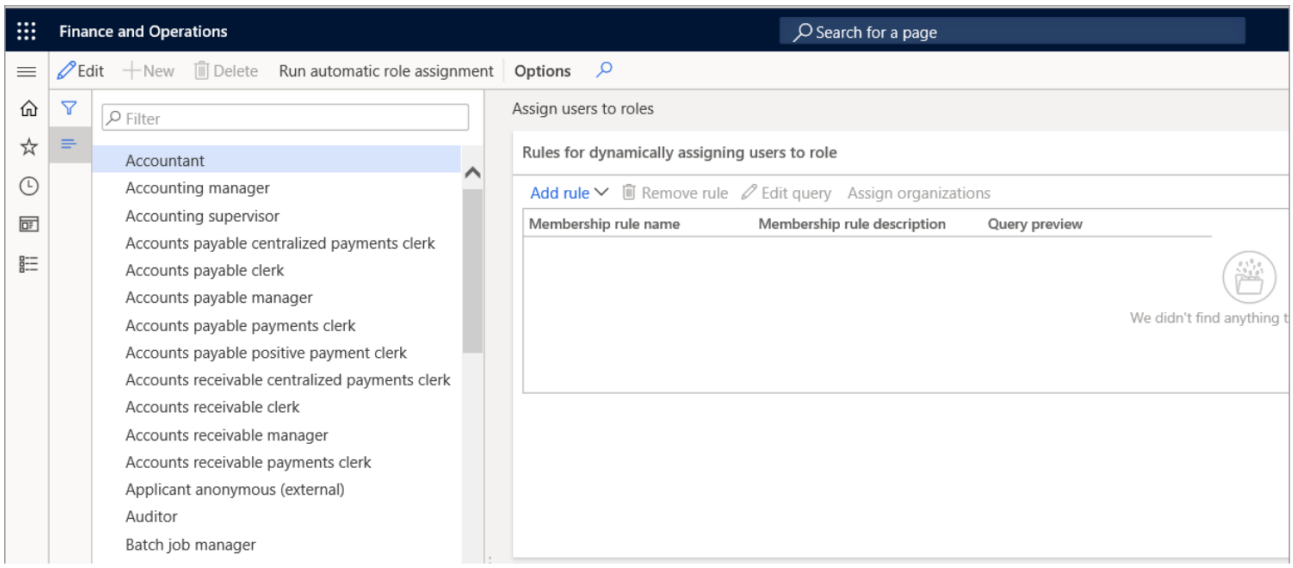

Рисунок 2.2 – Скріншот сторінки додавання користувачів ролям

За замовчуванням надаються зразки ролей безпеки. Усі функції в програмах Finance and Operations пов’язані принаймні з однією зі зразкових ролей безпеки. Адміністратор може призначати користувачам зразки ролей безпеки, змінювати зразки ролей безпеки відповідно до потреб бізнесу або створювати нові ролі безпеки. За замовчуванням зразки ролей не розташовані в ієрархії.

Обов'язки

Обов'язки відповідають частинам бізнес-процесу. Адміністратор призначає обов'язки ролям безпеки. Обов'язок може бути призначений більш ніж на одну роль. У програмах Finance and Operations обов’язки містять привілеї. Наприклад, обов'язок «Ведення банківських операцій» містить генерування депозитних квитанцій та скасування прав на оплату . Ролі безпеки можуть бути призначені як обов’язки, так і привілеї. Проте краще використовувати обов’язки для надання доступу до програм для фінансів та операцій.[22]

Привілеї

Привілеї можна, проте не бажано, призначати безпосередньо ролям. Привілеї визначають рівень доступу, необхідний для виконання роботи, вирішення проблеми або виконання завдання. Він також містить дозволи для окремих об’єктів програми, таких як елементи інтерфейсу користувача та таблиці.

Наприклад, право скасувати платежі містить дозволи на елементи меню, поля та таблиці, необхідні для скасування платежів.

За замовчуванням привілеї надаються для всіх функцій у програмі. Адміністратор може змінювати дозволи, пов’язані з привілеями, або створювати нові привілеї.

Дозволи

Дозволи представляють доступ до окремих захищених об’єктів, таких як пункти меню та таблиці. Привілеї складаються з дозволів і представляють доступ до завдань, таких як публікація журналу та обробка кредитів і колекцій. Обов’язки складаються з привілеїв і являють собою частини бізнес-процесу, наприклад ведення готівкових та банківських операцій. Обов’язки та привілеї можна призначити ролям, щоб надати доступ до системи.[22]

До кожної функції в програмі Finance and Operations, як форма або послуга, можна отримати доступ через точку входу. Пункти меню, елементи веб-вмісту та операції з обслуговуванням називаються пунктами входу.

Аутентифікація

Тільки автентифіковані користувачі, які мають права користувача в програмах Finance and Operations, можуть встановити з’єднання. Програми для фінансів та операцій використовують Microsoft Azure Active Directory (AAD) як основного постачальника ідентифікаційних даних. Щоб отримати доступ до системи, користувачі повинні бути підключені до екземпляра програм Finance and Operations і повинні мати дійсний обліковий запис AAD у авторизованого орендаря.

Авторизація

Авторизація — це контроль доступу до програми Dynamics 365 for Finance and Operations. Дозволи безпеки використовуються для контролю доступу до окремих елементів програми: меню, пунктів меню, кнопок дій і команд, звітів, операцій служб, пунктів меню веб-адреси, елементів керування веб-сайтом і полів у клієнті Dynamics 365 for Finance and Operations.

У програмах Finance and Operations окремі дозволи безпеки об’єднуються в привілеї, а привілеї об’єднуються в обов’язки. Адміністратор надає ролям безпеки доступ до програми, покладаючи на них обов'язки та привілеї.

Dynamics 365 for Finance and Operations використовують контекстну безпеку для визначення доступу до захищених об'єктів. Коли привілей пов’язано з точкою входу (наприклад, пункт меню або операція служби), вказується рівень доступу, наприклад Читання або Видалення. Підсистема авторизації програми Finance and Operations виявляє доступ під час виконання, коли здійснюється доступ до цієї точки входу, і застосовує вказаний рівень доступу до захищеного об’єкта, до якого веде точка входу. Ця функція допомагає гарантувати, що ви не призначаєте більше дозволів для користувача, ніж потрібно.

Аудит

Аудит входу та виходу користувачів увімкнено в програмі Finance and Operations. Система реєструє, коли користувач входить або виходить із програми. Вхід відбувається під час реєстрації, навіть якщо сеанс користувача закінчився. Системний адміністратор або адміністратор безпеки може отримати доступ до журналів аудиту.

Форма зовнішніх ролей дозволяє адміністраторам призначати ролі безпеки для ролі зовнішньої частини, як-от клієнт, постачальник, потенційний клієнт, потенційний постачальник і працівник.

Сплячі облікові записи безпеки користувачів дозволяють ідентифікувати неактивні облікові записи та дізнатися, увімкнені чи вимкнені облікові записи.

Висновки до розділу 2

У другому розділі було розглянуто та проаналізовано існуючі системи керування доступом двох популярних продуктів платформи Microsoft Dynamics 365 - Microsoft Dynamics Customer Engagement та Microsoft Dynamics for Finance and Operations. Було досліджено їх специфічні особливості та можливості налаштувань.

У наступному розділі буде детально розглянуто розроблено власну систему атрибутів специфічну для даних продуктів та розроблено систему для впровадження керування доступом на основі атрибутів в Microsoft Dynamics CRM.

3 РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ДОСТУПОМ НА ОСНОВІ АТРИБУТІВ

3.1 Аналіз проблеми

Рольовий контроль доступу був розроблений для більш простого світу. Формалізований NIST у 1992 році, RBAC швидко став стандартом для підприємств, які керують понад 500 співробітниками. Перевершуючи попередні моделі, RBAC був значною мірою реалізований у системах забезпечення користувачів, що дало підприємствам можливість керувати контролем доступу за ролями, а не індивідуальним ідентифікатором користувача працівника. [23]

ABAC дозволяє підприємству розширювати існуючі ролі за допомогою атрибутів і політик. Впровадження контролю доступу на основі атрибутів, дозволяє безпечно ділитися конфіденційною інформацією та дотримуватись вимог нормативних даних. Додаючи контекст, рішення про надання доступу можна приймати не лише на основі ролі користувача, але й з урахуванням того, з ким він має відношення, до чого потрібен доступ цього користувача, звідки йому потрібен доступ, коли цей доступ потрібен, і як цей користувач отримує доступ до запитуваної інформації. ABAC робить це, використовуючи політики, побудовані на індивідуальних атрибутах. Створюючи політику, яку легко зрозуміти, з контекстом навколо користувача та до того, до чого він має мати доступ, контроль доступу стає набагато більш надійним. Ця функція значно розширює сферу застосування RBAC.

Елементи керування ABAC можна записати у вигляді простих версій політики обміну інформацією. Після написання єдину політику можна розгортати в декількох системах і сотнях пристроїв. На відміну від традиційних елементів керування, які вимагають статичного визначення

дозволів перед спробою доступу, правила ABAC оцінюються динамічно за допомогою атрибутів, представлених під час виконання. Атрибути можуть надходити з кількох джерел – навіть із зовнішніх джерел для організації. Крім того, застосування автоматично адаптується до рівня ризику. Наприклад, якщо змінюється класифікація запису або змінюється членство в команді користувача, права доступу автоматично коригуються. Не потрібно запитувати нові ролі або оновлювати дозволи.

ABAC має набагато більшу кількість можливих керуючих змінних, ніж RBAC. ABAC впроваджено для зниження ризиків через несанкціонований доступ, оскільки він може контролювати безпеку та доступ на більш детальній основі. Наприклад, замість того, щоб люди в кадровій службі завжди могли отримати доступ до інформації про співробітників і нарахування заробітної плати, ABAC може встановлювати додаткові обмеження на їх доступ, наприклад, дозволяти це лише в певний час або для певних філій, що мають відношення до відповідного працівника. Це може зменшити проблеми з безпекою, а також може допомогти в процесі аудиту пізніше.

В CRM та ERP системах використовується рольова політика керування доступом. Проте є чимало вимог, які можуть бути важливими та навіть критичними для підприємств, що використовують CRM та ERP системи, для реалізації яких існуючих систем контролю доступу недостатньою, проте при впровадженні системи доступу заснованої на атрибутах, їх можна задовільнити:

1. Обмеження часу доступу до ресурсів: Існуючі системи не дозволяють обмежити час доступу до певних ресурсів для окремих користувачів, тільки для всієї системи в цілому.

2. Доступ на виконання операції над об’єктом повинен залежати від інформації про об’єкт: В системах немає можливості впроваджувати складні

правила доступу. Наприклад, рядовий співробітник не має права завершувати продаж, якщо сума продажу перевищує ліміт.

3. Доступ на виконання операції над об’єктом повинен залежати від інформації про суб’єкт: В системах немає можливості впроваджувати складні правила доступу. Наприклад, рядовий співробітник не має права завершувати продаж, якщо число затверджень перевищує ліміт.

4. Обмеження доступу до нового функціоналу: При впровадженні CRM та ERP систем в більшості компанії не користуються тільки стандартним функціоналом системи, а також впроваджують власні додаткові процеси які можуть наприклад змінювати якісь дані, чи інтегруватися з зовнішньою системою. Існуючою рольовою політикою не має можливості керувати доступом до цих процесів.

5. Права повинні динамічно змінюватися при зміні посади співробітника: Наразі ролі можуть бути назначені тільки відповідальним співробітником, а не змінені автоматично.

Для можливості більш гнучких налаштувань безпеки необхідно реалізувати систему доступу на основі атрибутів.

Найкраще буде не замінити наявну рольову модель безпеки моделлю заснованою на атрибутах, а скомбінувати обидві моделі. Поєднання RBAC і ABAC може допомогти адміністраторам отримати найкраще з обох систем. RBAC і ABAC можна використовувати разом ієрархічно, з широким доступом, забезпеченим протоколами RBAC, і більш складним доступом, керованим ABAC. Їх змішування поєднує в собі сильні сторони обох.

3.2 Розробка системи атрибутів для CRM та ERP

Було досліджено популярні атрибути в контексті безпеки бізнес-процесів та проаналізовано можливості систем безпеки Microsoft Dynamics

CRM та Microsoft Dynamics AX. На основі отриманих знань було розроблено власну систему атрибутів для керування доступом для виконання деяких дій в системі, що містить найбільш важливі атрибути проте лише ті, що є та важливі для CRM та ERP систем та які можна отримати в контексті виконання певної операції:

Суб’єкт, що здійснює дію:

- безпосередньо користувач;
- посада користувача;
- бізнес-одиниця користувача;
- ролі користувача;
- команда користувача;
- країна користувача;
- місто реєстрації користувача;
- тощо.

Об’єкт

- таблиця (опціонально, залежить від дії);
- стовпець(опціонально, залежить від дії);
- статус запису (опціонально, залежить від дії);
- власник запису (опціонально, залежить від дії);
- тощо (опціонально, залежить від дії).

Операція:

- назва дії.

Стан:

- день тижня;
- час.

Деякі уточнення стосовно розробленої системи:

- Список атрибутів пов’язаних з суб’єктом чи об’єктом може бути розширений будь-якою інформацію (наприклад, лімітами на виконання операцій). Це зроблено для можливості додавання додаткових атрибутів за необхідності, забезпечує динамічність.
- Дані про таблиці та стовпці необхідні тільки для деяких дій, таких як створення, оновлення, отримання – читання, але не потрібні для, наприклад, такої дії, як кваліфікація інтересу – перетворення інтересу(потенційного клієнта) в контакт (клієнт) після збору про нього всіх необхідних даних і готовий до заключення угоди;
- Дії можуть бути як звичайні: створити, видалити, оновити, так і більш складні – кваліфікувати інтерес, експортувати, асоціювати, а також будь-які кастомні дії.
- В доступному для реалізації рішенні немає можливості отримати інші атрибути стану окрім дати та часу, такі як ip-адресу, пристрій, так як вони не містяться в контексті виконання операції.

Даний набір атрибутів є оптимальним для використання в CRM та ERP системах, проте за вимоги його можливо розширювати додатковими атрибутами, що забезпечує динамічність розробленої системи.

3.3 Реалізація рішення для впровадження ABAC для CRM системи

Microsoft Dynamics CRM дозволяє розробляти та впроваджувати власні кастомні рішення над стандартним функціоналом системи. Спектр можливих налаштувань досить великий. Можна створювати та налаштовувати нові таблиці, колонки, зв’язки, форми, представлення, графіки, ролі безпеки, скрипти, плагіни та багато іншого.

Було розроблено власну структуру рішення для впровадження керування доступом на основі атрибутів до Microsoft Dynamics CRM. Структурна схема рішення відображена на рисунку 3.1. Було вирішено використовувати такі компоненти як, нова сутність (таблиця) «ABAC Rule», а отже правила будуть зберігатися безпосередньо в CRM системі, нова сутність «ABAC Audit» для збереження логування, плагіну, який буде викликатися безпосередньо перед виконанням операцій в системі та кроки плагіну для його виклику при різних подіях. Діаграма взаємодії сутностей, на якій представлені зв'язки між новими сутностями «ABAC Rule» та «ABAC Audit» та стандартними сутностями Microsoft Dynamics CRM, представлена на рисунку 3.2. На рисунку 3.3 відображено процес виконання плагіну, які методи викликаються та в якій послідовності. Опис кожного методу буде представлений далі.

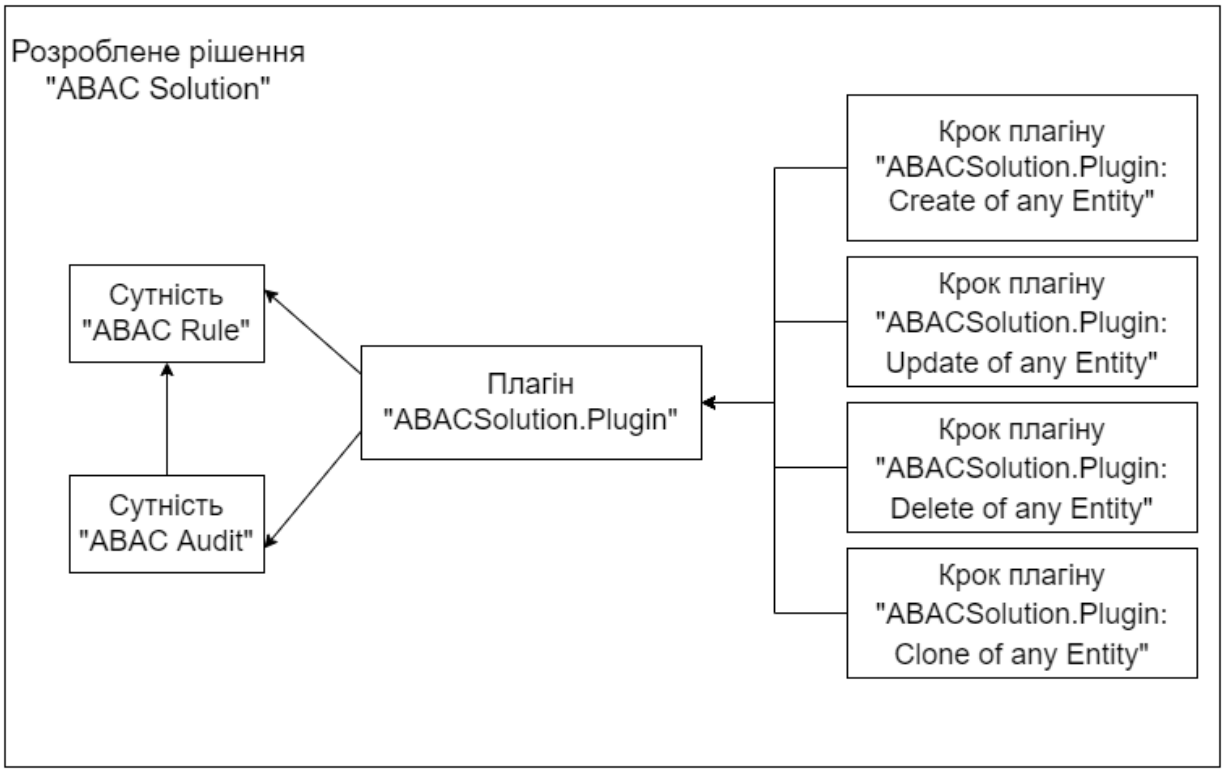

Рисунок 3.1 – Структурна схема рішення

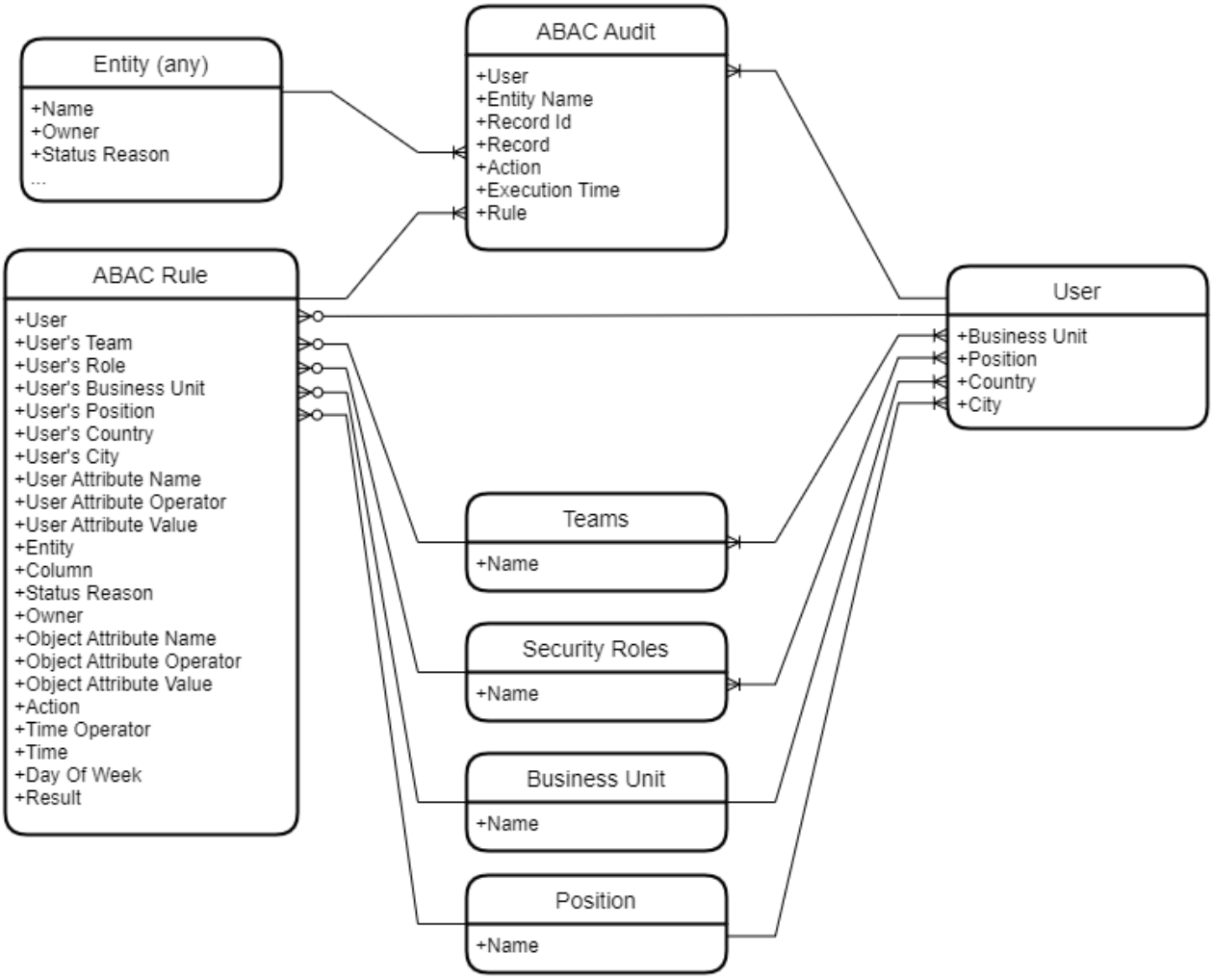

Рисунок 3.2 – Діаграма взаємодії сутностей

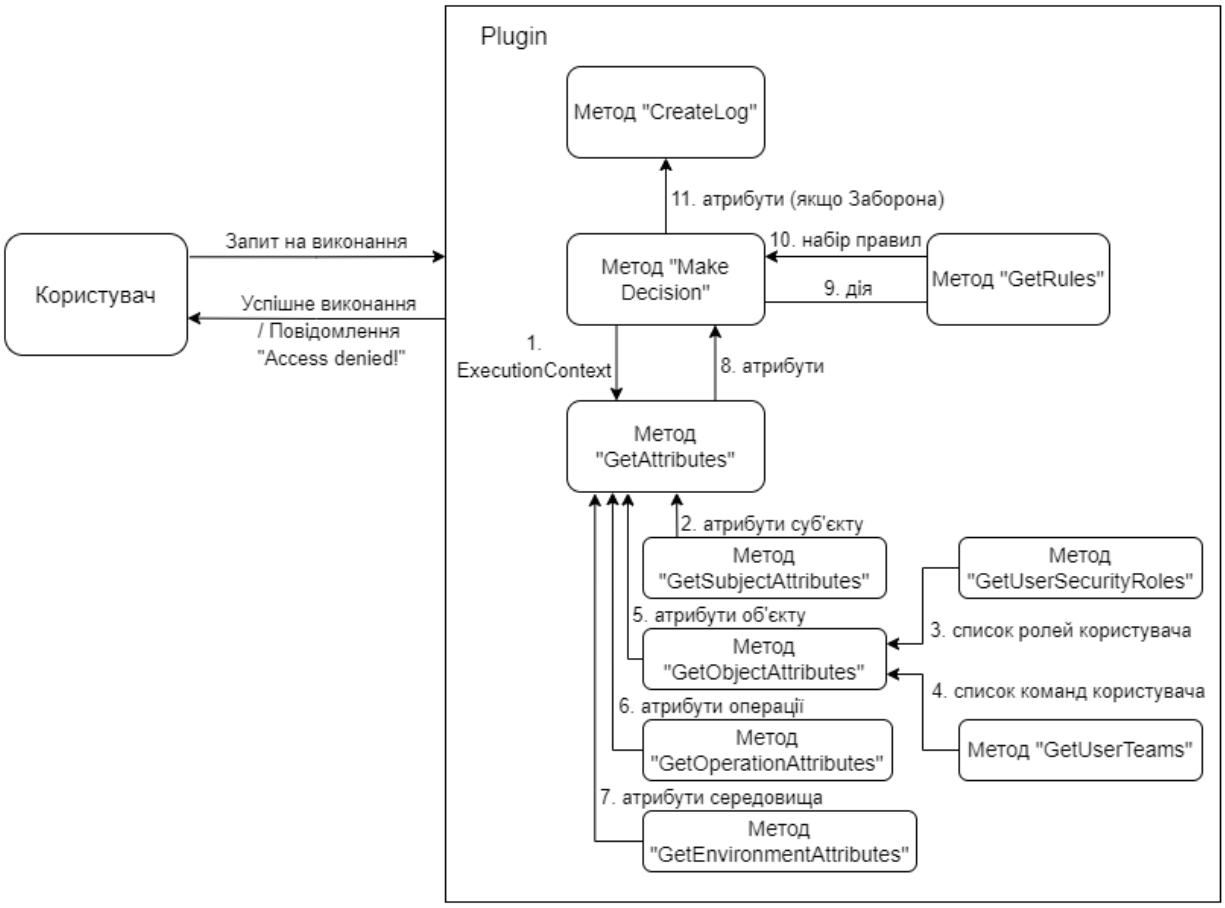

Рисунок 3.3 – Процес виконання плагіну

Кроки, які було виконано під час реалізації рішення:

1. Розгорнуто trial-версію Microsoft Dynamics Customer Engagement;
2. Створено видавця та нове рішення в системі;
3. Здійснено кастомізації системи;
4. Розроблено кастомний плагін;
5. Опубліковано та налаштовано плагін в системі.

Далі буде детальніше описано кожен з виконаних кроків.

Розгортання trial-версії Microsoft Dynamics Customer Engagement

Багато продуктів Microsoft можна використовувати безкоштовно протягом місяця, що дуже корисно як для цілей ознайомлення с потенційним продуктом користувача, так і для навчання. Для розгортання в першу чергу

потрібно перейти за посиланням Get started with a Dynamics 365 free trial [24], натиснути на кнопку «Try for free» для продукту Dynamics 365 Sales та виконати послідовну процедуру реєстрації нового аккаунту. В якості доменного імені нового аккаунту було вказано LiakhCorporation.onmicrosoft.com. Після успішної реєстрації аккаунту було відкрито Power Platform Admin Center, в якому було створено нове trial середовище з url https://org60fd2bc3.crm4.dynamics.com/. Trial-версія Microsoft Dynamics Customer Engagement розгорнута, та перейшовши за посиланням та виконавши процедуру аутентифікації авторизований користувач отримує доступ до системи.

Створення видавця та рішення в системі

Так як при розгортанні автоматично було обрано основну мову системи Російська компоненти інтерфейсу системи будуть вказуватись саме на цій мові. Перейшовши до «Параметры-Настройки-Издатели» можна створити нового Видавця в системі. Видавці потрібні для розмежування для ідентифікації творця нових компонентів в системі. Це необхідно коли над розробкою/кастомізаціями системи одночасно працюють декілька людей або компаній. Було створено видавця з ім’ям Daria Liakh та префіксом dl (рисунок 3.4).

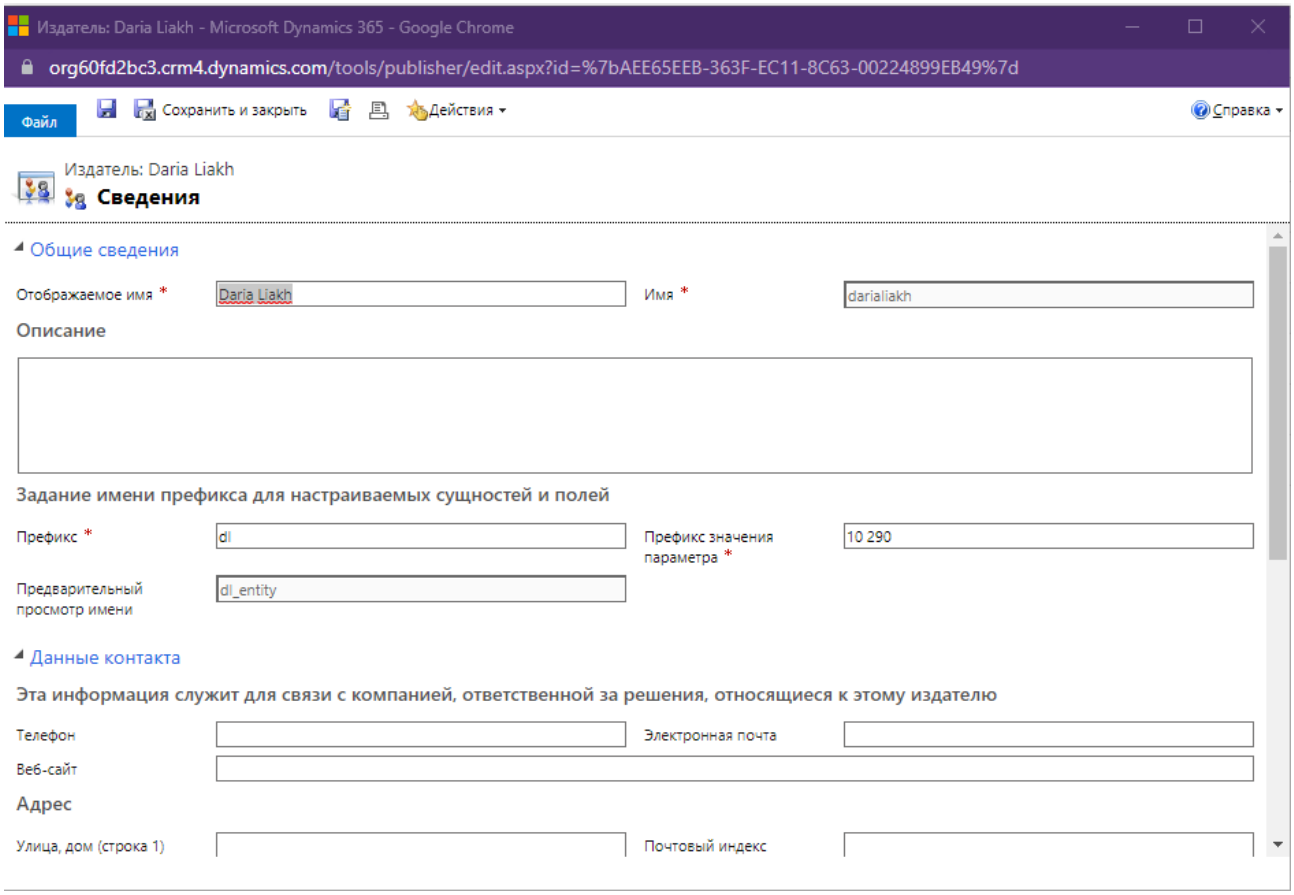

Рисунок 3.4 – Скріншот форми створення видавця

Даний префікс як раз і допомагає виконувати основну функцію видавця, оскільки для нових компонентів системи, якщо вони створюються у рішенні даного видавця, префікс буде за замовчуванням вказуватися на початку імені схеми цього компоненту (наприклад, створена далі таблиця буде мати ім’я sb_abacrule). Для створення рішення необхідно перейти до «Параметры-Решения» та натиснути кнопки «Создать». У вікні браузера для створення рішення було вказано ім’я рішення – «ABAC Solution», видавця – Daria Liakh та номер версії – 1.0.0.0. В описі рішення було вказано – «The solution for implementation attribute-based access control for Microsoft Dynamics Customer Engagement» та натиснуто кнопку «Сохранить» (рисунок 3.5).

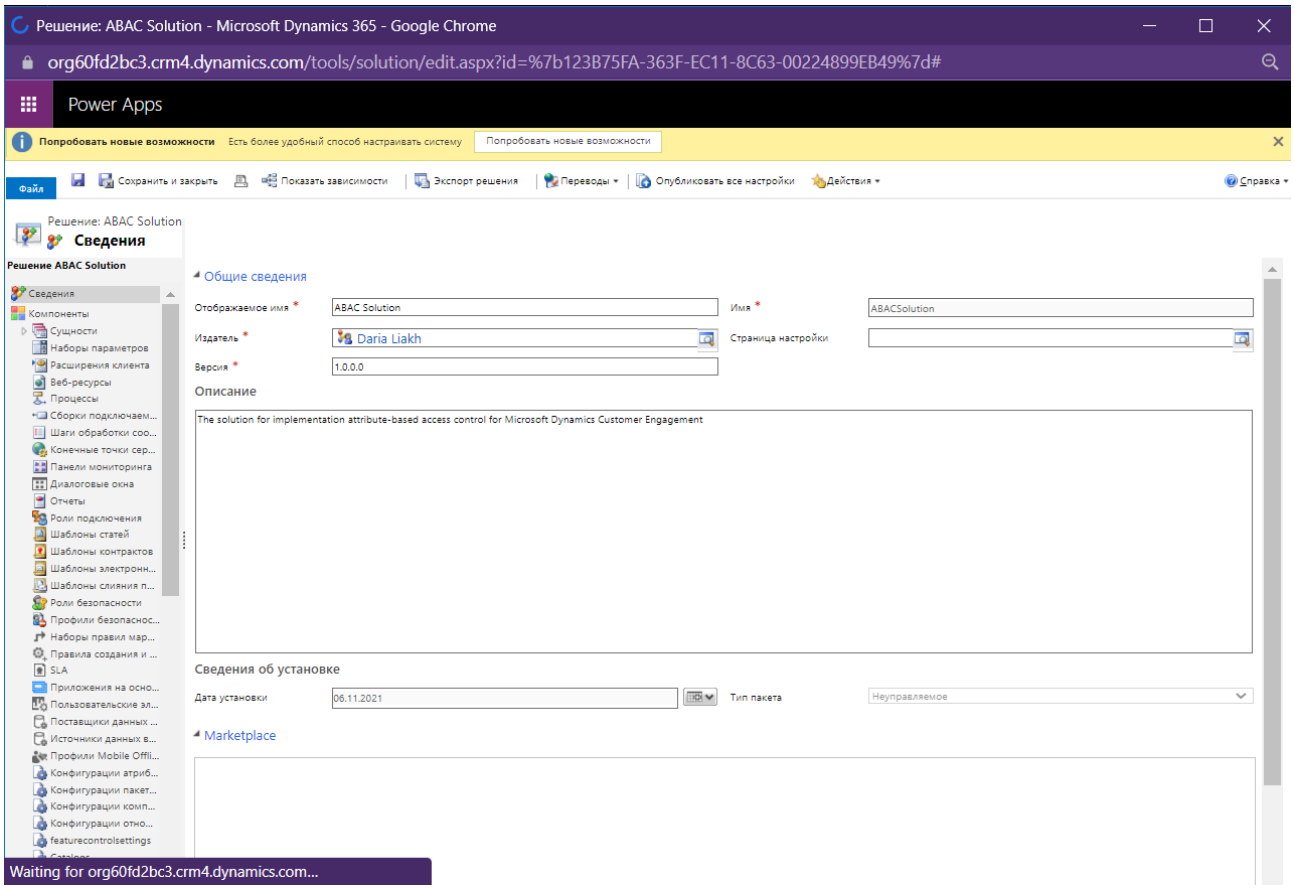

Рисунок 3.5 – Створене рішення «ABAC Solution»

Кастомізації системи

Кастомізації на даному кроці реалізації проводилися в рішенні «ABAC Solution», створеному на минулому кроці. По-перше було створено нову кастомну сутність. Було вказано ім’я, що відображається – «ABAC Rule», ім’я в множині - ABAC Rules та повне ім’я – dl_abacrule. В цій сутності було створено нові поля, інформація про назви та типи полів вказана в Таблиці 3.1.

Таблиця 3.1 – Нові поля сутності «ABAC Rule»

Ім’я, що відображається	Ім’я	Тип поля	Значення
1	2	3	4

User	dl_userid	Пошук (Користувач)	
User’s Team	dl_usersteamid	Пошук (Команда)	
User’s Business Unit	dl_usersbusinessunitid	Пошук (Бізнес одиниця)	
User’s Position	dl_userspositionid	Пошук (Положення)	
User’s Role	dl_userroleid	Пошук (Роль безпеки)	
User’s Country	dl_userscountry	Рядок тексту	
User’s City	dl_userscity	Рядок тексту	
User Attribute Name	dl_userattributename	Рядок тексту	
User Attribute Operator	dl_userattributeoperator	Набір параметрів	102900000 – Equals, 102900001 – Does Not Equal, 102900002 – Is Greater Than, 102900003 – Is Greater Than or Equal To 102900004 – Is Less Than 102900005 – Is Less Than or Equal To 102900006 – Contains

			102900007 – Does Not Contain
User Attribute Value	dl_userattributevalue	Рядок тексту	
Entity	dl_entity	Рядок тексту	
Column	dl_column	Рядок тексту	
Status Reason	dl_statusreasoncode	Рядок тексту	
Owner	dl_ownerid	Пошук (Користувач, Команда)	
Object Attribute Name	dl_objectattributename	Рядок тексту	
Object Attribute Operator	dl_objectattributeoperator	Набір параметрів	102900000 – Equals, 102900001 – Does Not Equal, 102900002 – Is Greater Than, 102900003 – Is Greater Than or Equal To 102900004 – Is Less Than 102900005 – Is Less Than or Equal To 102900006 – Contains 102900007 – Does Not Contain

Object Attribute Value	dl_objectattributevalue	Рядок тексту	
Action	dl_action	Рядок тексту	
Time Operator	dl_timeoperatorcode	Набір параметрів	102900000 – Is Less Than, 102900001 – Is Greater Than, 102900002 – In Period,
Time	dl_time	Рядок тексту	
Day of Week	dl_dayofweekcode	Набір параметрів з вибором декількох варіантів	102900001 – Monday, 102900002 – Tuesday, 102900003 – Wednesday, 102900004 – Thursday, 102900005 – Friday, 102900006 – Saturday, 102900000 – Sunday
Result	dl_resultcode	Набір параметрів	102900000 – Allow, 102900001 – Deny

Далі ці поля було винесено на форму Information сутності ABAC Rule. Поля

поділені на 5 секцій – Subject Atttributes, Object Atttributes, Operation, Environmental Conditions, Result, як зображено на рисунках 3.6-3.8.

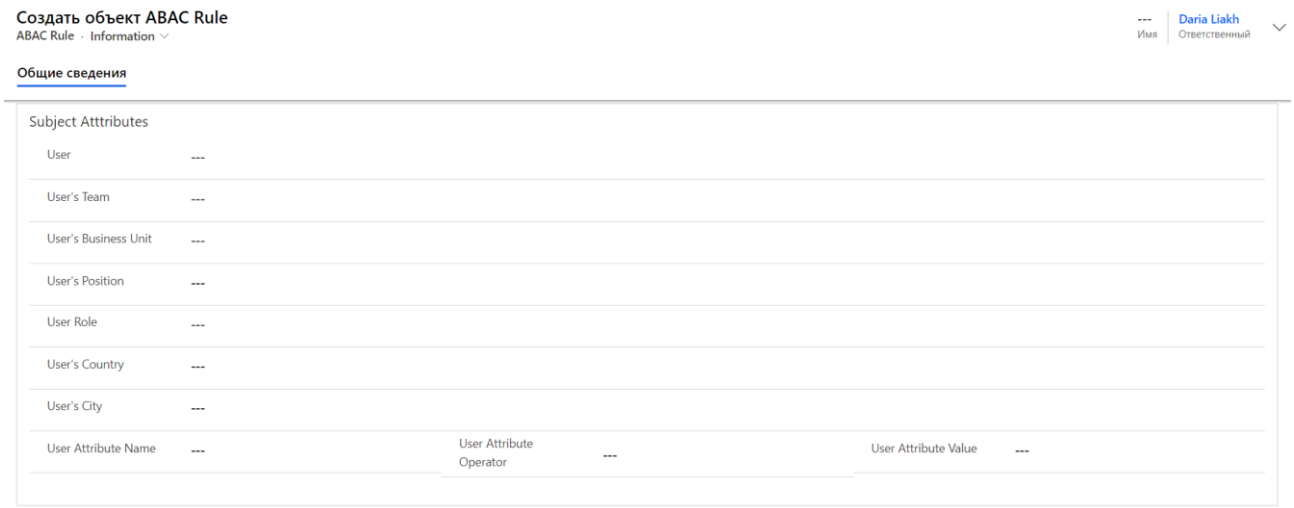

Рисунок 3.6 – Форма запису сутності «ABAC Rule» - частина 1

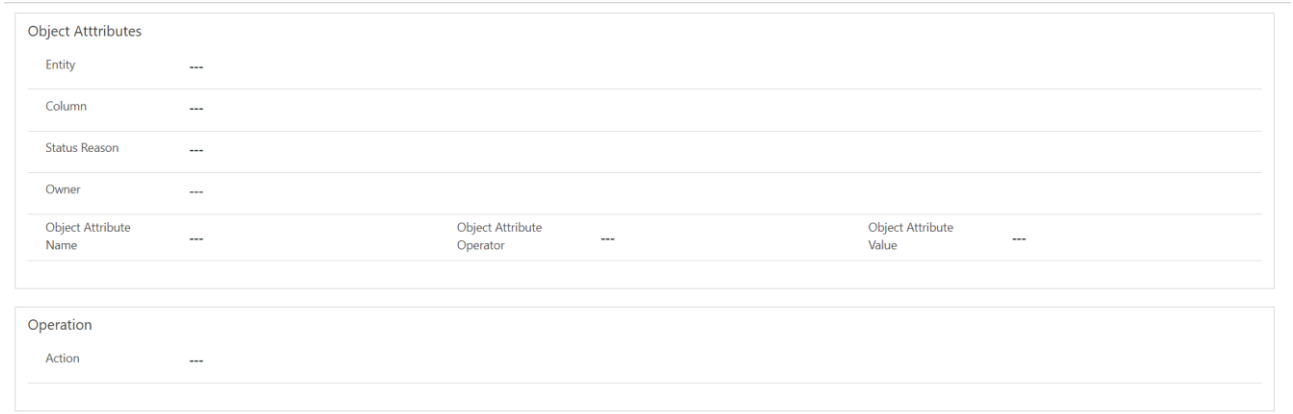

Рисунок 3.7 – Форма запису сутності «ABAC Rule» - частина 2

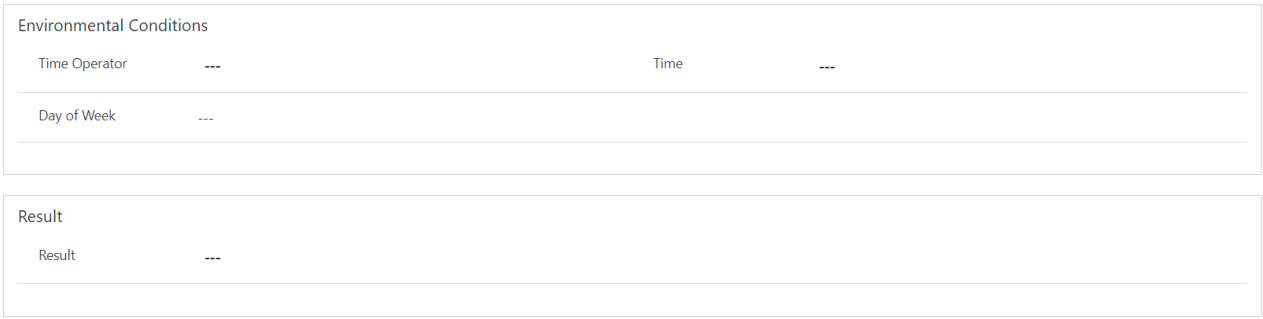

Рисунок 3.8 – Форма запису сутності «ABAC Rule» - частина 3

Аналогічно попередній було також створено сутність сутності «ABAC Audit» (ім’я схеми - dl_abacaudit). Нові створені поля відображено у таблиці 3.2.

Таблиця 3.2 – Нові поля сутності «ABAC Rule»

Ім’я, що відображається	Ім’я	Тип поля	Значення
1	2	3	4
User	dl_userid	Пошук (Користувач)	
EntityName	dl_entityname	Рядок тексту	
Record	dl_idrecord	Рядок тексту	
Record	dl_recordid	Пошук (Всі сутності)	
Action	dl_action	Рядок тексту	
Time	dl_excutiontime	Дата та час	
Result	dl_result	Набір параметрів	102900000 – Allow, 102900001 – Deny

Та винесено поля на нову форму Information в одну секцію як зображено на рисунку 3.9.

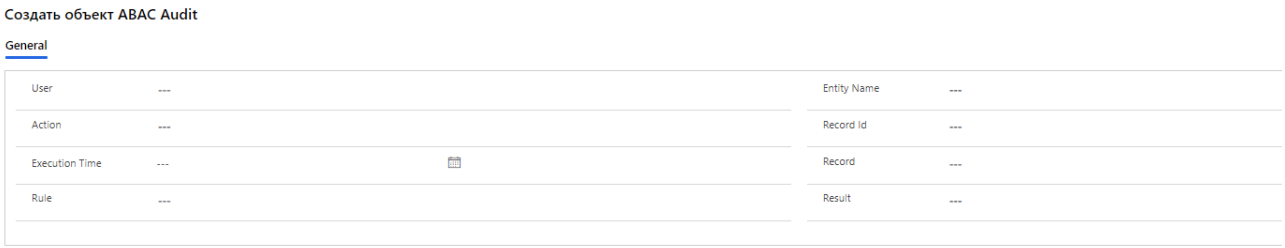

Рисунок 3.9 – Форма запису сутності «ABAC Audit»

Також було створено додаток «ABAC App», до якого були додані наші нові сутності «ABAC Audit» та «ABAC Rule» (рисунок 3.10). Він створений для того, щоб мати доступ до цих сутностей через додаток.

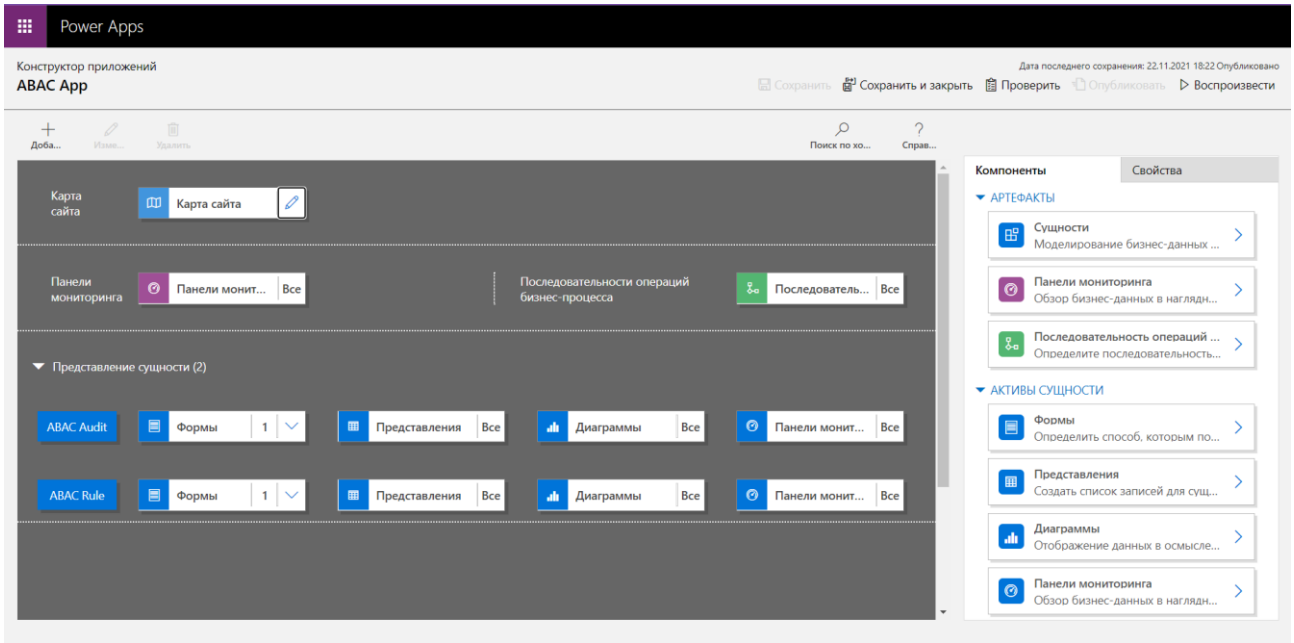

Рисунок 3.10 – Вікно конструктора додатку «ABAC App»

Розробка плагіну

Microsoft Dynamics CRM дає можливість розробляти та впроваджувати логіку, що виконується під час виконання стандартних операцій за допомогою плагінів. Плагіни — це користувацький код, який додається до вбудованого конвеєра подій CRM. Це дозволяє розробникам розширити поведінку CRM за замовчуванням, щоб вони могли адаптувати її саме до потреб вашої організації. Основний принцип роботи плагінів полягає у тому, що кожен раз, коли ви виконуєте дію в Microsoft CRM, за лаштунками виконується процес, відомий як конвеєр виконання подій. Плагін — це код, який повідомляє CRM робити щось додаткове, чого вона ще не робить, коли відбувається певна дія.

Подіями можуть бути такі дії в системі як створення запису, оновлення, видалення, асоціація, призначення та ще багато інших.

Існує декілька етапів, на яких може відбуватися плагін:

- Pre-Validation – виконується перед транзакцією, використовується для додаткових перевірок перед початком операції.
- Pre-Operation – виконується безпосередньо перед транзакцією, використовується наприклад коди при оновленні запису потрібно оновити ще одне поле.
- Post-Operation – виконується після виконання транзакції в базі даних, використовується для виконання додаткової логіки після успішного виконання операції.

Тобто для реалізації перевірки доступу до виконання операції можна створити власний плагін, що буде виконуватися перед початком виконання операції та перевіряти її санкціонованість. Якщо операція заборонена замість успішного виконання операції з'явиться повідомлення з текстом «Access denied!».

Плагін написаний на мові C# .NET з використанням бібліотеки Microsoft.CrmSdk.CoreAssemblies [25] для можливостей взаємодії з CRM. Дана бібліотека надає є інтерфейс IPlugin, який буде наслідувати наш плагін та містить метод Execute. Інтерфейс IPlugin – інтерфейс, який використовується усіма плагінами, можуть бути опубліковані в Microsoft Dynamics CRM. Наслідування цього інтерфейсу забезпечує, що плагін буде містити метод Execute, в якому розміщується основна логіка плагіну, а також отримання контексту виклику, користувача, що виконує операцію, можливо об'єкту, що містить всі зміни, сервісу трасування, тощо. Також бібліотека надає інтерфейс IOrganizationService, який містить набір методів для роботи з сутностями в системі. Він надає можливість виконувати операції створення, оновлення, видалення, отримання одного запису та багатьох записів, виконання

системних запитів, виклики кастомних процесів, а також отримувати метадані сутностей.

Плагіни повинні бути частиною колекції, а тому Для створення проекту колекції в Visual Studio було обрано та створено проект типу Class Library (.NET Framework) та вказано ім’я ABACSolution. Обов’язковою умовою публікації колекції в Dynamics CRM є те, що вона мають бути підписана, тому одразу після створення проекту він був підписаний – було створено ключ – key.snk та було встановлено Nuget-пакет «Microsoft.CrmSdk.CoreAssemblies» для взаємодії з CRM. Далі будо створено файли:

- Plugin.cs – в якому буде вся логіка плагіну;
- ABACRuleModel.cs – для зберігання імен схеми полів сутності «ABAC Rule»;
- ABACAuditModel.cs – для зберігання імен схеми полів сутності «ABAC Audit»;
- AttributeModel.cs – для зберігання атрибутів;
- ResultEnum.cs – перерахування значень поля Result сутності «ABAC Rule»;
- TimeOperatorEnum.cs – перерахування значень поля Time Operator сутності «ABAC Rule»;
- AttributeOperatorEnum.cs – перерахування значень полів UserAttributeOperator та ObjectAttributeOperator сутності «ABAC Rule»;
- JsonSerializer.cs – серіалізатор (для відображення моделі атрибутів в Trace Log – система логування роботи плагінів);
- EntityExtension.cs – міститься метод для злиття двох Entity в одну.

Плагін містить в собі такі методи:

- Execute – головний метод плагіну, здійснює отримання контексту та сервісів с контексту, виклик методів MakeDecision та CreateLog та створення виключення за необхідності;

- MakeDecision – метод для збору атрибутів, правил, перевірки та прийняття рішення про санкціонованість доступу;
- GetAttributes – метод для отримання атрибутів;
- GetSubjectAttributes – метод для отримання атрибутів суб’єкту користувача;
- GetObjectAttributes – метод для отримання атрибутів об’єкту (за наявності);
- GetOperationAttributes – метод для отримання атрибутів операції;
- GetEnvironmentAttributes – метод для отримання атрибутів середовища;
- GetUserSecurityRoles – метод для отримання команд, до яких належить даний користувач;
- GetUserTeams – метод для отримання команд даного користувача;
- GetRules – отримання активних записів правил за назвою дії;
- CreateLog – метод, для створення запису логу.

Логіка плагіну наступна:

- отримання атрибутів суб’єкту по унікальному ідентифікатору користувача з контексту виконання операції (context.UserId);
- отримання атрибутів об’єкту з вхідних параметрів контексту, за умови їх наявності (context.InputParameters["Target"]);
- отримання назви дії (атрибуту операції) з контексту виконання операції (context.MessageName);
- отримання атрибутів середовища – часу та дня тижня (DateTime.Now та DateTime.Now.DayOfWeek);
- отримання правил - активних записів сутності «ABAC Rule» за назвою поточної операції;

- перевірка всіх записів правил по кожному з заповнених значень атрибутів на предмет наявності запису, що забороняє виконання даної операції з поточними атрибутами;
- якщо наявний хоча б один запис заборони буде виключення (throw new Exception("Access denied!")), а також буде створено запис сутності «ABAC Audit», що свідчитиме про спробу отримання несанкціонованого доступу, якщо ні – то операція виконається як і повинна.

Ознайомитися з кодом плагіну можна у Додатку А.

Публікація та налаштування плагіну в системі

Публікація плагіну відбувається не безпосередньо в системі, а за допомогою програми XrmToolBox та її утиліти Plugin Registration. Plugin Registration – інструмент реєстрації плагінів Dynamics CRM. Для реєстрації по перше потрібно відкрити Plugin Registration та ввести облікові дані та вибрати екземпляр Dynamics CRM. Далі натиснувши кнопку Register, обравши Register New Assembly та обравши файл попередньо зібраної бібліотеки динамічної компоновки нашого проекту (ABACSolution.dll) колекція буде зареєстрована в системі CRM (рисунок 3.11).

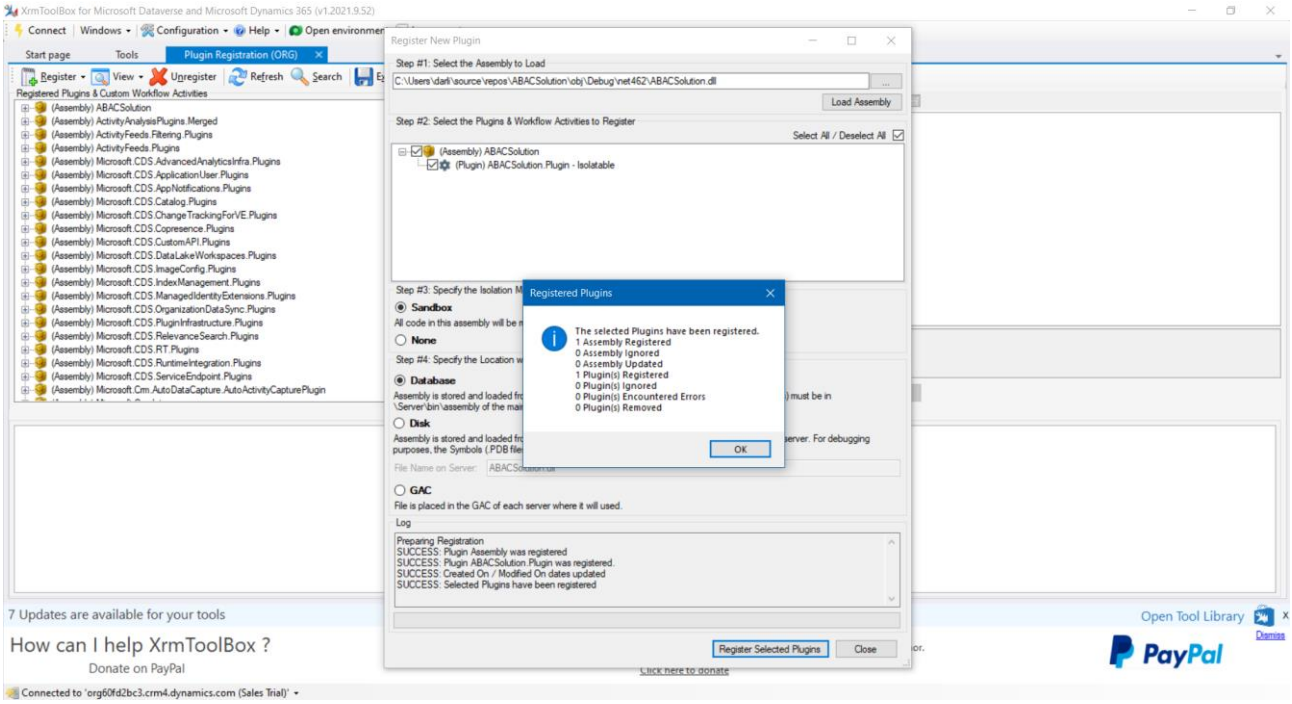

Рисунок 3.11 – Скріншот успішної реєстрації колекції плагінів «ABACSolution»

Проте цього не достатньо для роботи плагіну. Необхідно також зареєструвати кроки плагіну. Для кожної події, для якої необхідна перевірка атрибутів треба зареєструвати окремий крок плагіну. Приклад реєстрації одного з кроків на Рисунку 3.12. В полі Message вказується назва дії, кожен крок реєструється на одну дію. Якщо не вказувати назву сутності в полі Primary Entity плагін буде викликатися для будь-якої сутності в системі. Як вже вказувалося раніше крок реєструється на етапі Pre-Validation та режим виконання плагіну обов’язково синхронний.

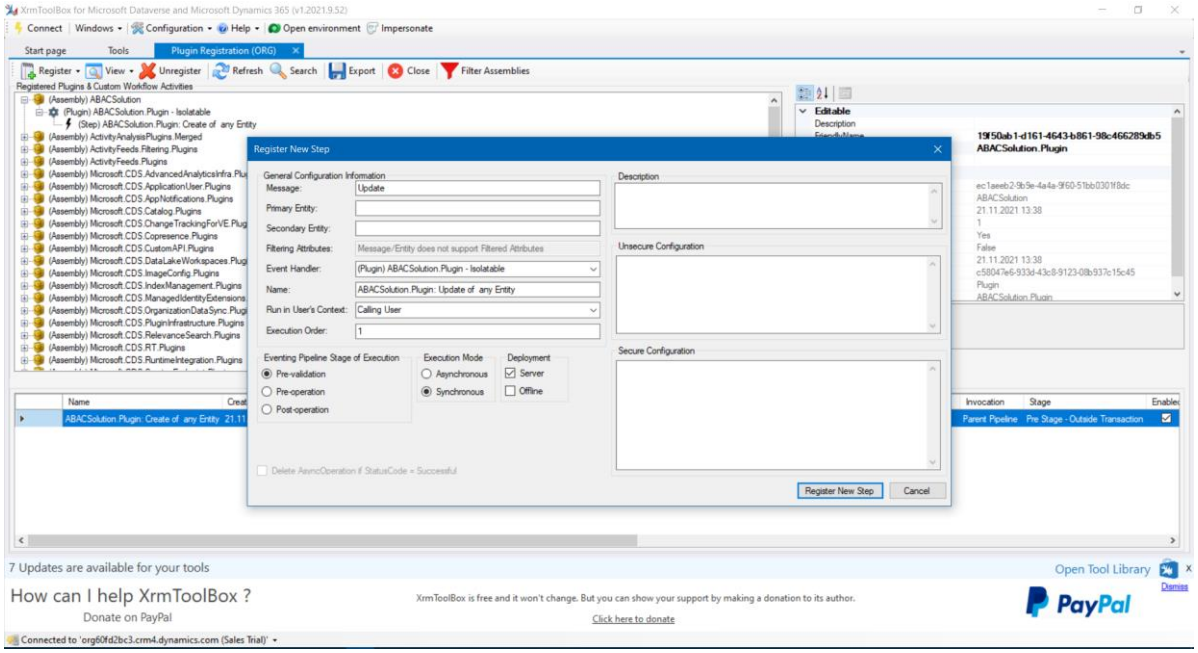

Рисунок 3.12 – Скріншот прикладу реєстрації кроку плагіну

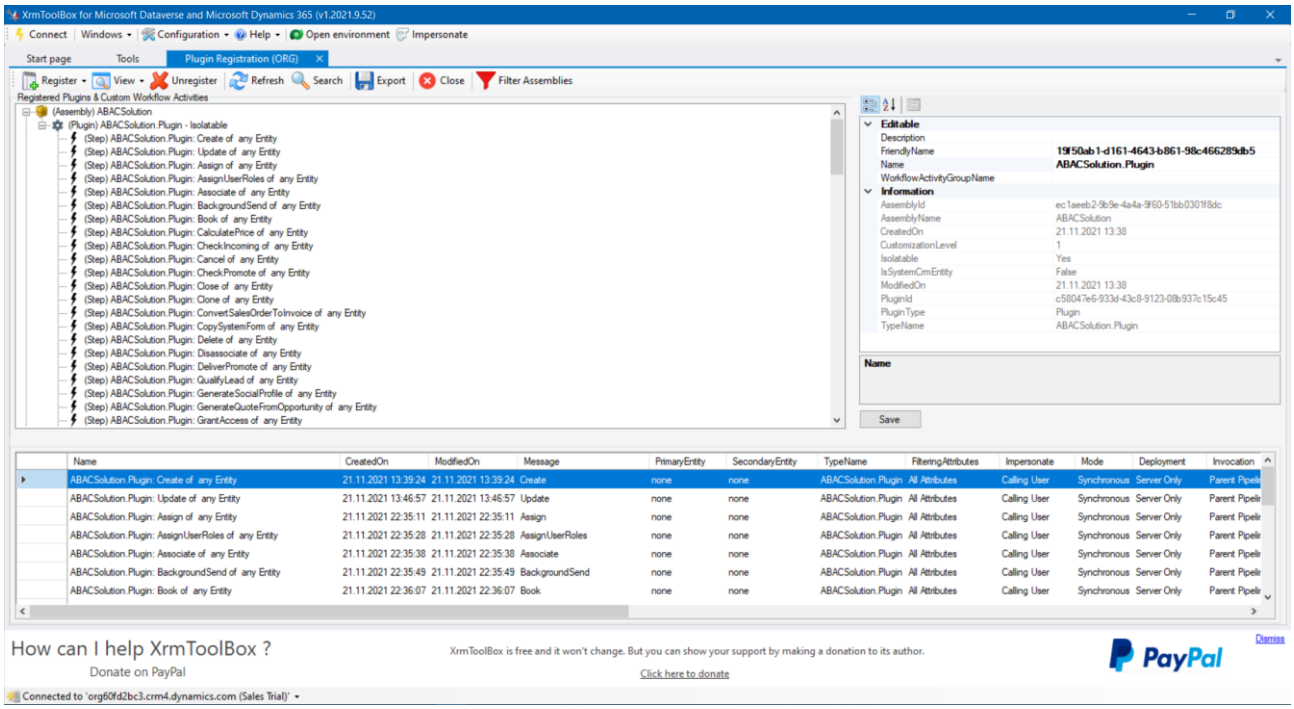

Рисунок 3.13 – Скріншот після реєстрації багатьох кроків плагіну для відпрацювань його на різні події

Після того як всі необхідні кроки було зареєстровано їх було додано у рішення. Це необхідно для того, щоб при переносі рішення до іншої системи кроки також були перенесені.

Розробка рішення завершена. Далі рішення можна вивантажити з системи як Managed (неможливе редагування в іншому екземплярі системи) та виконати імпорт в іншу систему, для якої необхідно впровадження контролю доступу на основі атрибутів, виконати деякі налаштування та можна користуватися рішенням.

Для впровадження керування доступом на основі атрибутів до іншого екземпляру Microsoft Dynamics CRM необхідно виконати такі кроки:

1. Виконати імпорт рішення «ABAC Solution» в систему.
2. За необхідності створити додаткові кроки плагіну для кастомних дій, специфічних для даної системи
3. Створити необхідні записи в таблиці «ABAC Rule», що відповідають правилам політики безпеки.

3.2 Демонстрація роботи рішення

Було реалізовано декілька прикладів, що демонструють роботу рішення для впровадження контролю доступу на основі атрибутів в Microsoft Dynamics CRM.

Приклади правил:

1. Записи дзвінка не повинні створюватися після 19.00 та у вихідні дні – суботу та неділю (Рисунок 3.14);
2. Користувач на позиції «Junior HR» не повинен здійснювати кваліфікацію інтересу (Рисунок 3.15-3.16);

3. Оператор контакт-центру не може видаляти чи відміняти Замовлення, якщо вартість перевищує 50 000 гривень (Рисунок 3.17-3.18).

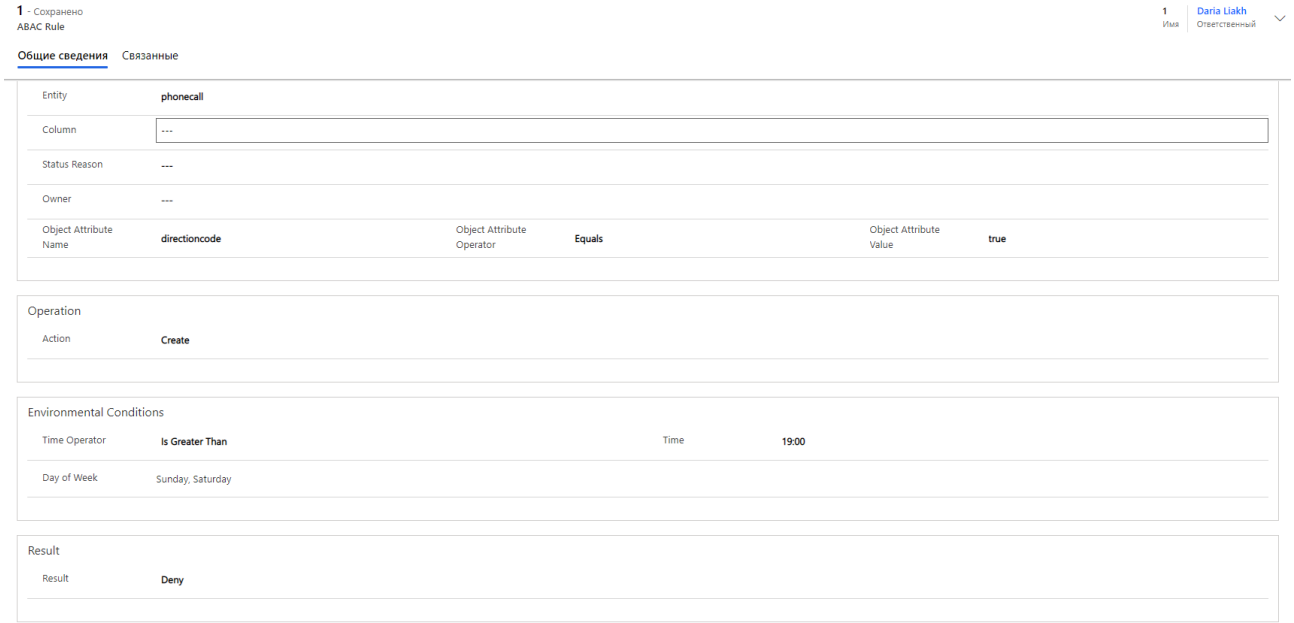

Рисунок 3.14 – Скріншот запису правила номер 1

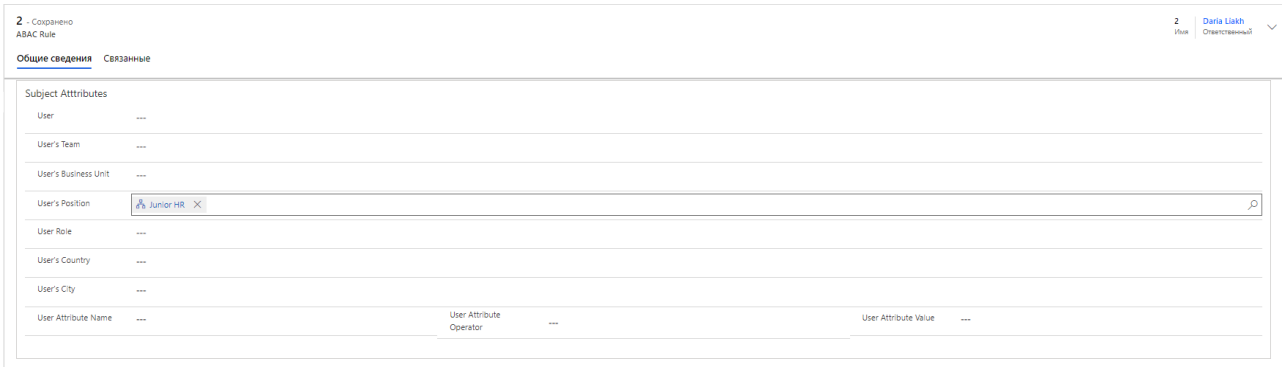

Рисунок 3.15 – Скріншот запису правила номер 2 – частина 1

Entity	---				
Column	---				
Status Reason	---				
Owner	---				
Object Attribute Name	---	Object Attribute Operator	---	Object Attribute Value	---

Operation

Action	QualifyLead

Environmental Conditions

Time Operator	---	Time	---
Day of Week	---		

Result

Result	Deny

Рисунок 3.16 – Скріншот запису правила номер 2 – частина 2

3.1 - Сохранено
ABAC Rule

3.1 Имя | Daria Liakh Ответственный

Общие сведения Связанные

Subject Atttributes

User	---				
User's Team	---				
User's Business Unit	---				
User's Position	Оператор контакт-центру				
User Role	---				
User's Country	---				
User's City	---				
User Attribute Name	---	User Attribute Operator	---	User Attribute Value	---

Object Atttributes

Entity	salesorder				
Column	statecode				
Status Reason	---				
Owner	---				
Object Attribute Name	totalamount	Object Attribute Operator	Is Greater Than or Equal To	Object Attribute Value	50000

Operation

Action	Cancel

Рисунок 3.17 – Скріншот запису правила номер 3 для заборони відміни замовлення

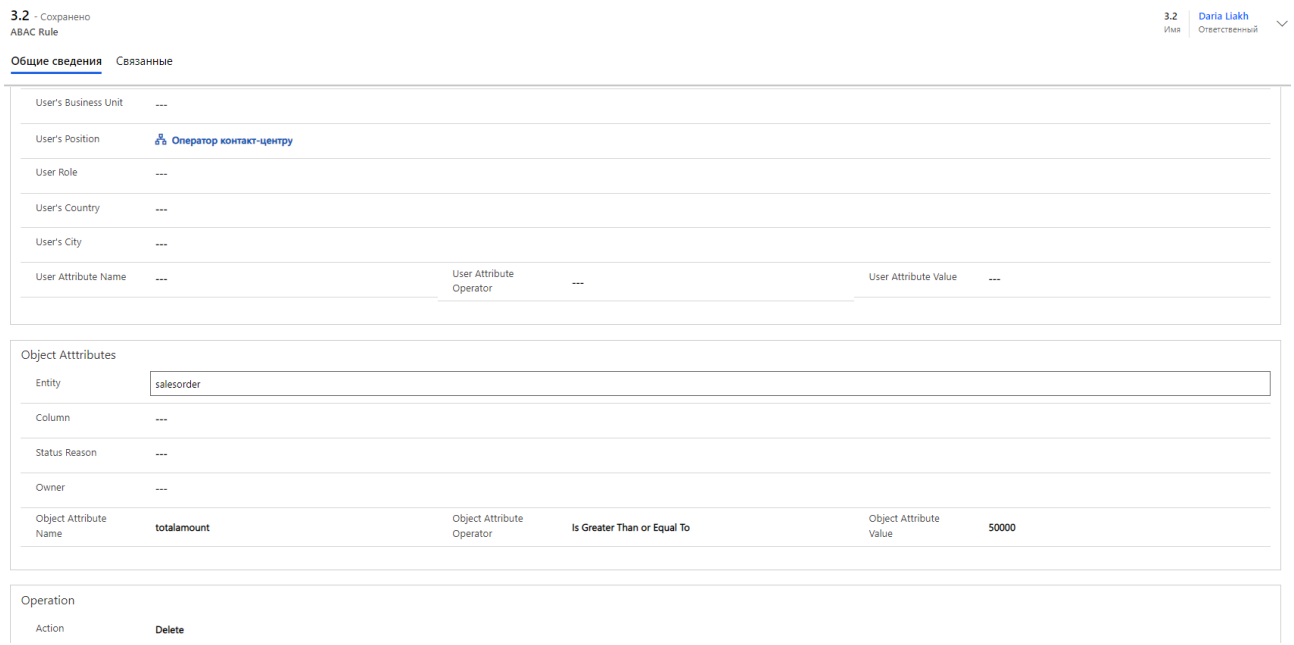

Рисунок 3.18 – Скріншот запису правила номер 3 для заборони видалення запису

Перевірка першого правила відбувається шляхом створення дзвінку у неробочий час після 19.00. При натисканні кнопки збереження нового запису буде виведено повідомлення про помилку(на рисунку 3.19) та запис не буде збережено, та створено запис сутності «ABAC Audit» (рисунок 3.20). Якщо змінити напрямок дзвінку, запис буде успішно створений (рисунок 3.21).

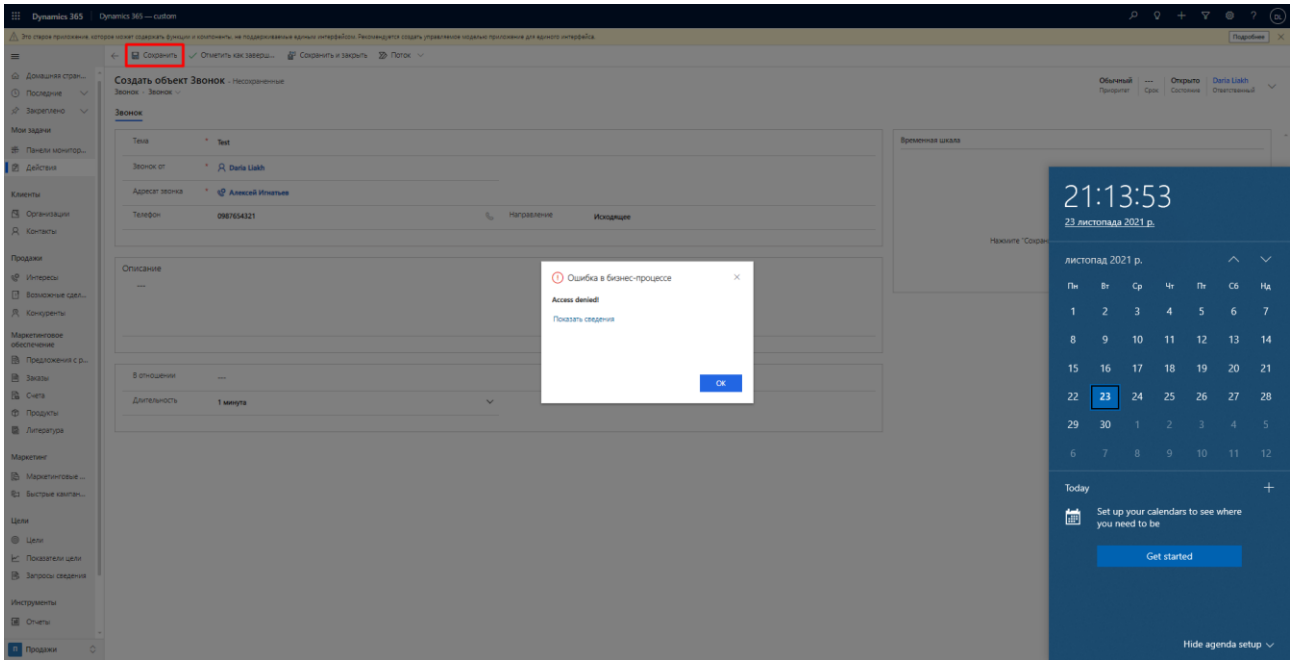

Рисунок 3.19 – Скріншот результату спроби створення вихідного виклику у неробочий час

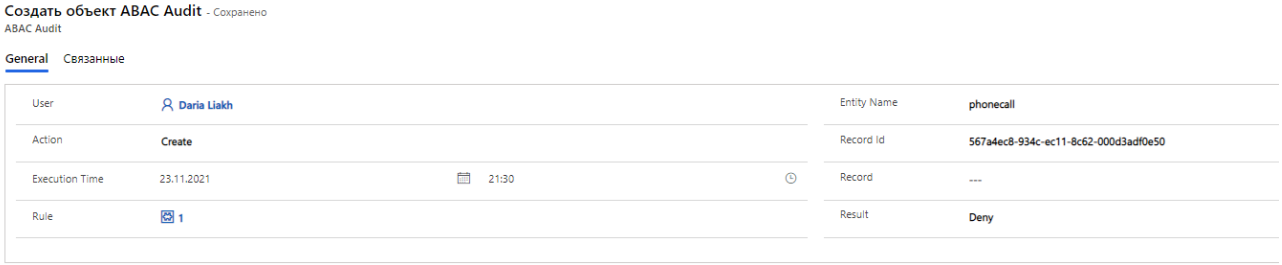

Рисунок 3.20 – Скріншот створеного логу після спроби створення вихідного виклику

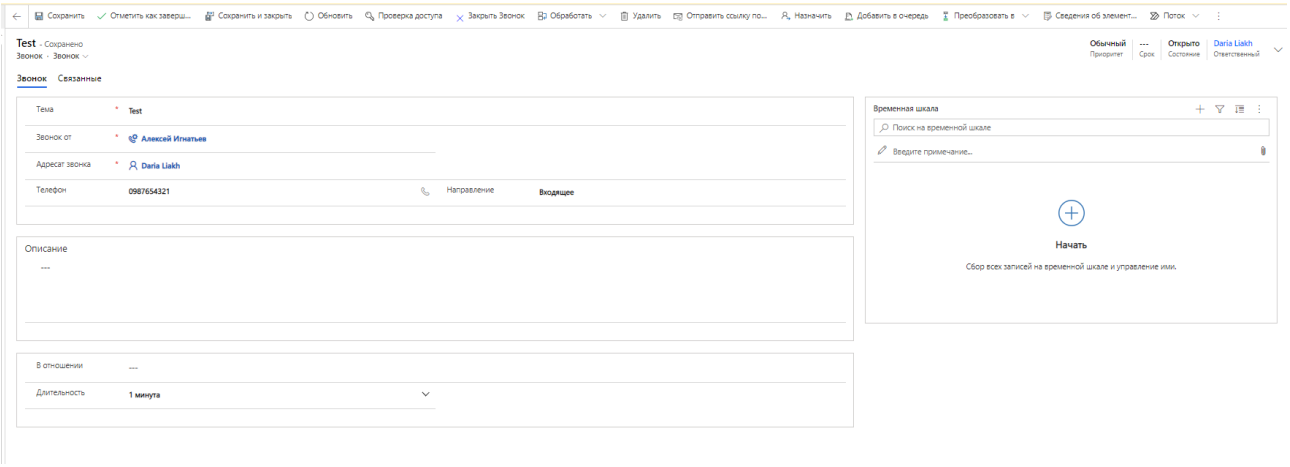

Рисунок 3.21 – Скріншот успішного створення запису вхідного виклику

Для перевірки другого привила власного користувача було призначено на позицію «Junior HR» (рисунок 3.22). Після цього було виконано спробу кваліфукації існуючого у системі інтересу, спроба була неуспішна (рисунок 3.23), з’явилося повідомлення про помилку та був створений запис логу (рисунок 3.24).

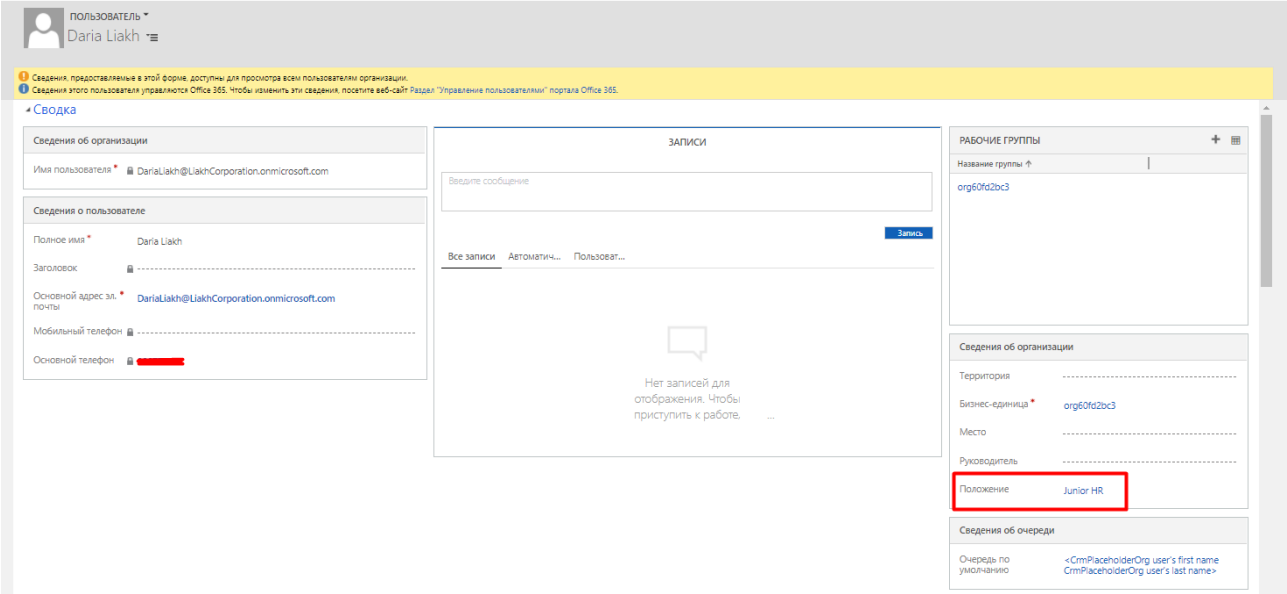

Рисунок 3.22 – Скріншот запису користувача з обраною позицією «Junior HR»

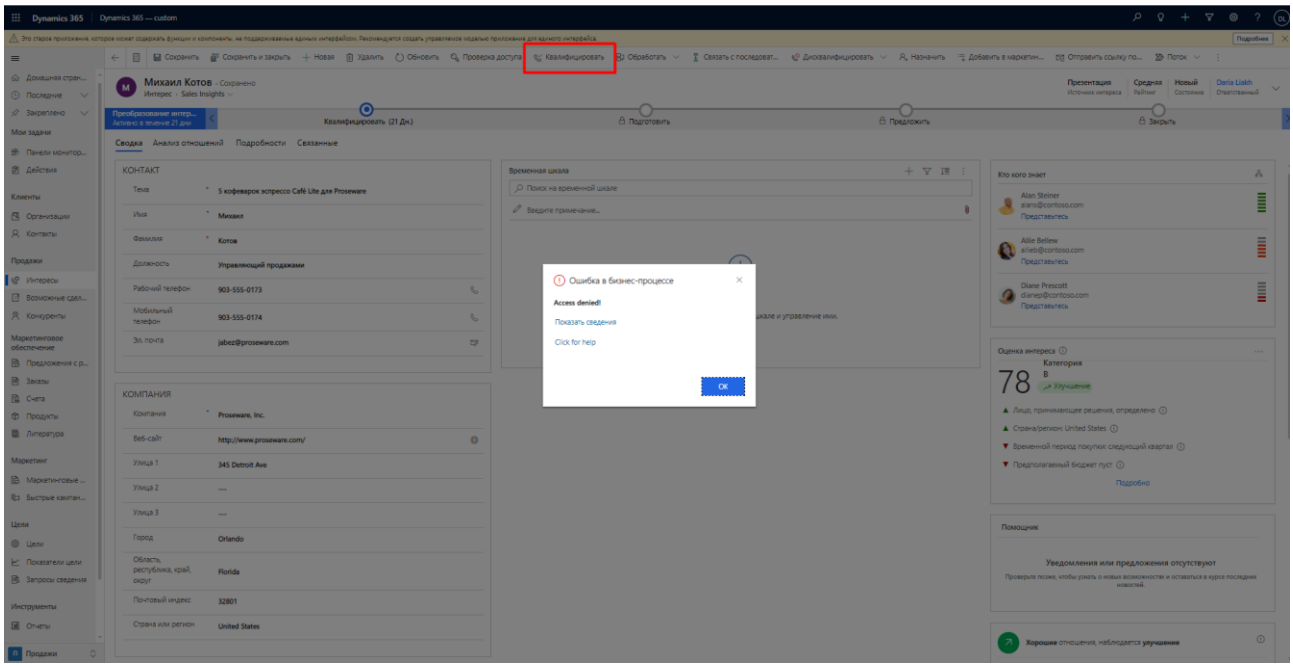

Рисунок 3.23 – Скріншот результату спроби кваліфікації інтересу користувачем на позиціїї «Junior HR»

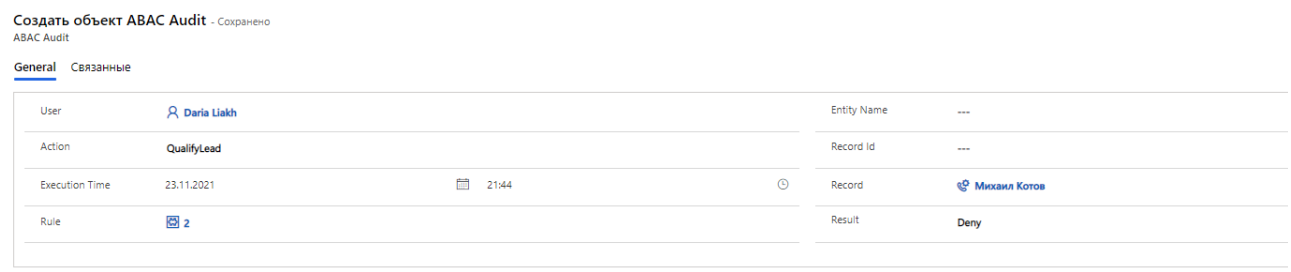

Рисунок 3.24 – Скріншот створеного логу після спроби кваліфікації інтересу

Далі для перевірки наступного правила позиція користувача була змінена на «Оператор контакт-центру» (рисунок 3.25), так як немає наразі правил, що б забороняли кваліфікацію інтересу для користувача на даній позиції, кваліфікація інтересу була виконана успішно (рисунок 3.26).

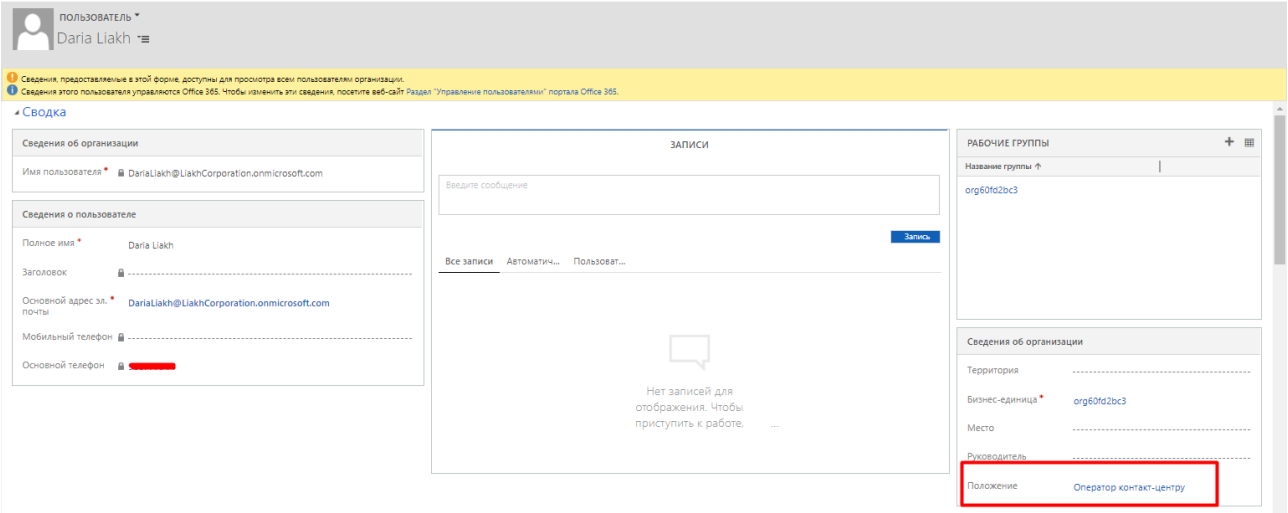

Рисунок 3.25 – Скріншот запису користувача з обраною позицією «Оператор контакт-центру»

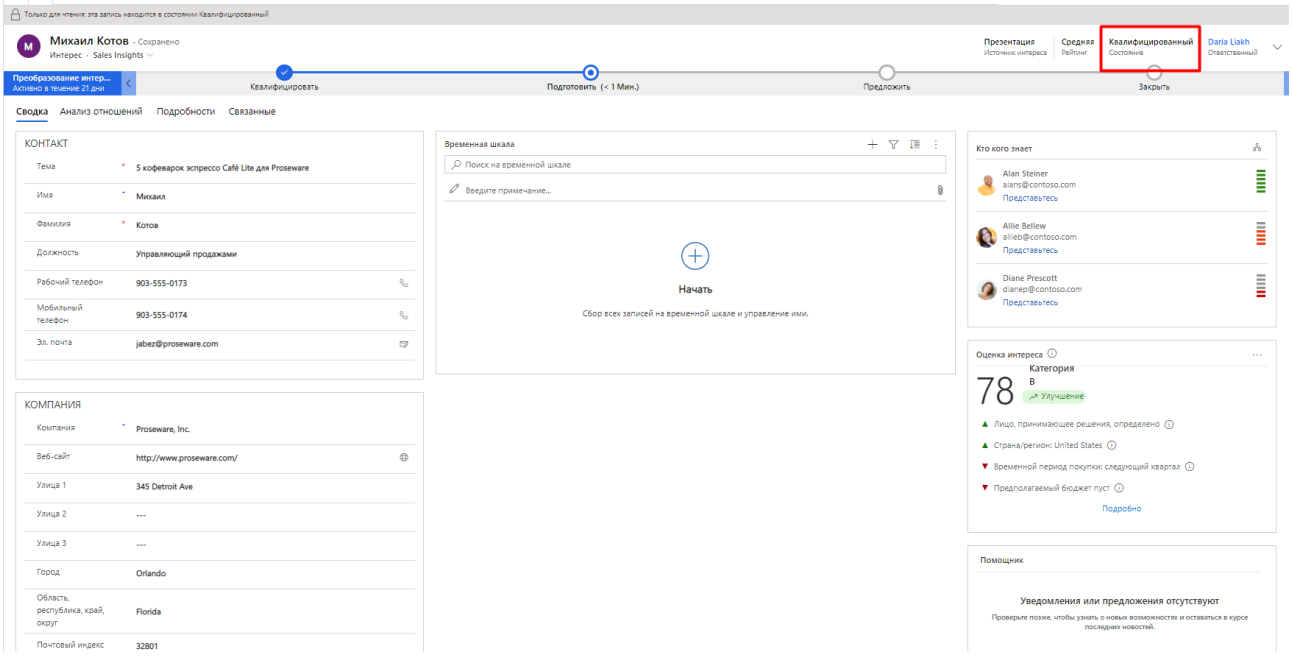

Рисунок 3.26 – Скріншот успішної кваліфікації інтересу після зміни позиції користувача

Користувач наразі знаходиться на позиції «Оператор контакт-центру». Було створено нове замовлення, та додано товар. Сума замовлення перевищує 50 000 грн. Було виконано спроби Відміни замовлення та Видалення запису про замовлення (рисунок 3.27). Обидві операції пройшли невдало та були створені записи логування даних спроб(рисунок 3.28, 3.29).

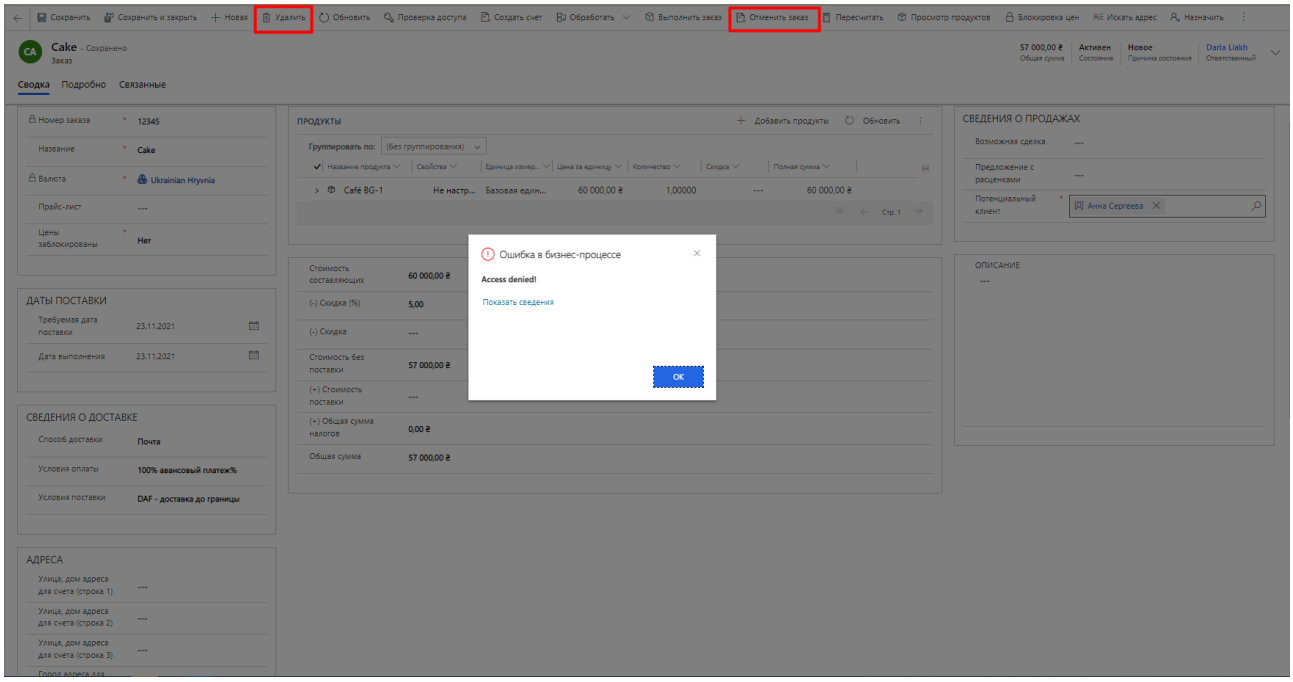

Рисунок 3.27 – Скріншот результату спроби видалення чи відміни замовлення

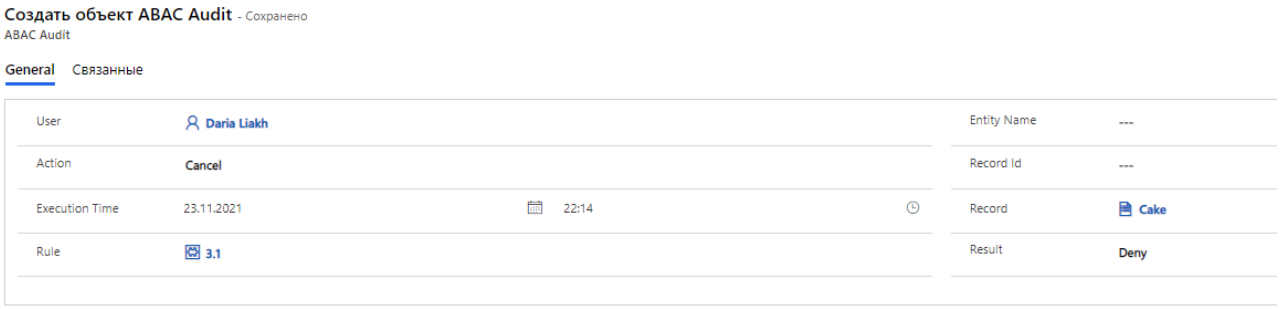

Рисунок 3.28 – Скріншот створеного логу після спроби відміни замовлення на суму вище 50000 грн

Рисунок 3.29 – Скріншот створеного логу після спроби видалення запису замовлення на суму вище 50000 грн

Було також створено інший запис замовлення, сума якого менше 50 000 грн. В даному випадку операція відміни замовлення пройшла успішно (рисунок 2.30).

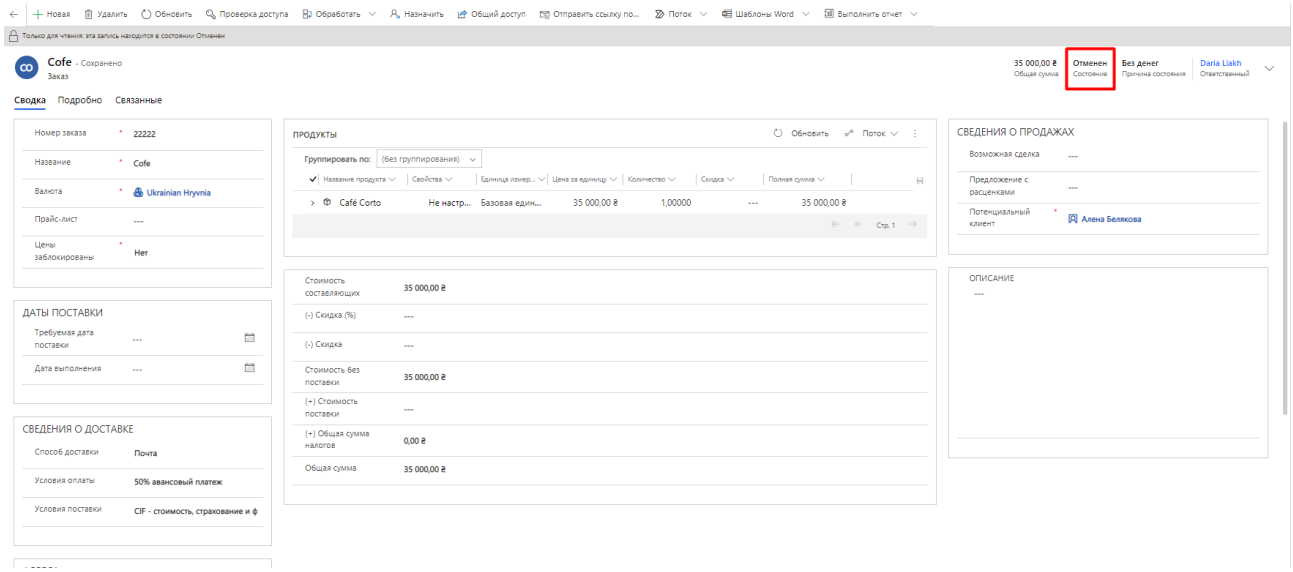

Рисунок 3.30 – Скріншот успішної відміни замовлення на суму менше 50000 грн.

Отже ми переконалися, що розроблене рішення працює коректно.

3.4 Подальший розвиток рішення

Реалізоване рішення є однією з реалізацій, яке вже можливе до використання, проте можливий його розвиток та розширення функціональності. Було продумано деякі кроки подальшого покращення рішення, які можна впровадити на наступних етапах його розвитку:

1. Плагін: реалізація плагіну для автоматичного створення кроків плагіну при створенні правила за їх відсутності , в такому випадку відпаде необхідність створення додаткових кроків вручну при налаштуванні рішення.
2. Скрипти: Додати обов’язковість полів для форми сутності та заблокувати можливість модифікації записів сутності на формі.
3. PCF-контроли: реалізувати PCF-контроли, що будуть реалізовувати пошук, для вибору значень замість заповнення текстових полів для полів Action, Entity, Column для мінімізації ймовірності помилок при адмініструванні через людський фактор.
4. Логіка: реалізувати більш складну логіку для можливості вибору операторів порівняння для кожного з атрибутів, можливості одночасно вказувати декілька конкретних значень, та не вказувати конкретні значення, а тільки відносні, наприклад, «Не має можливості переглядати Контакти, країна яких не співпадає з країною користувача».

Тоді оновлена структура рішення буде виглядати так як зоображено на рисунку 3.31.

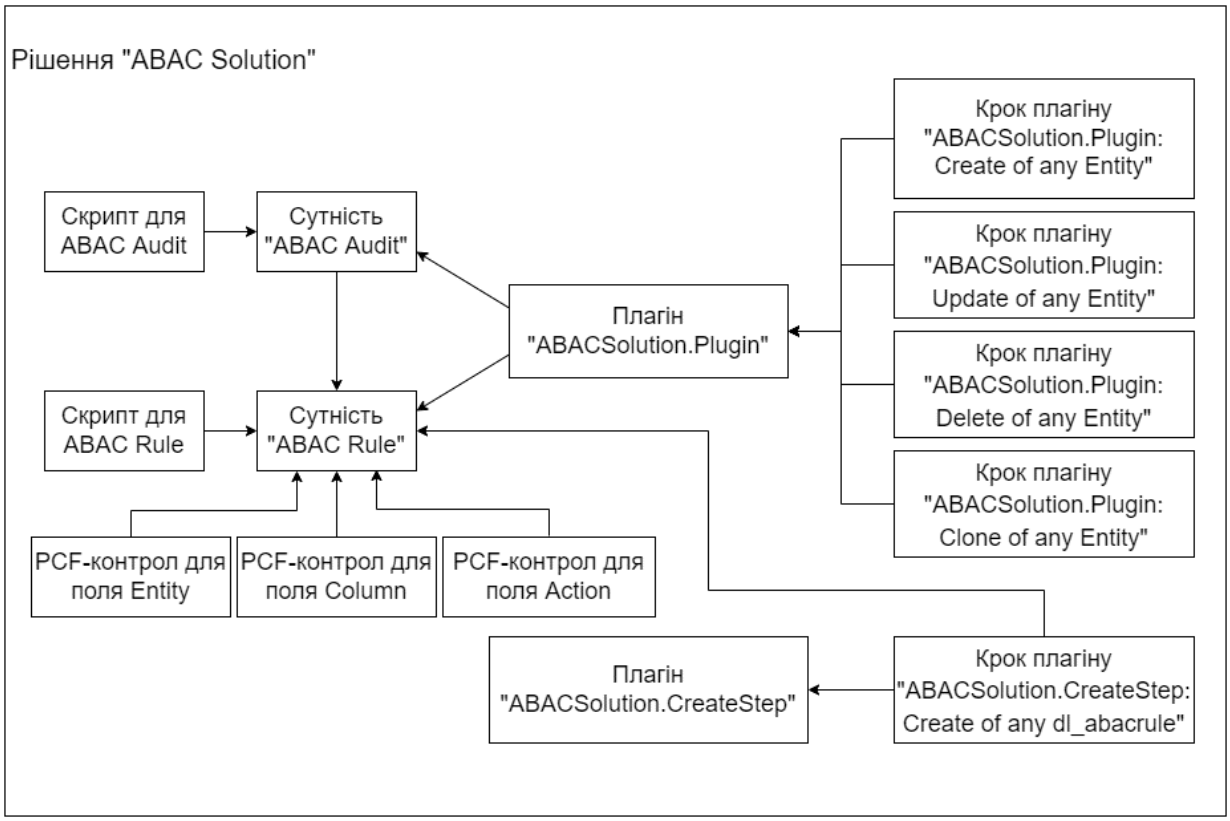

Рисунок 3.31 – Структура розширеного рішення

Висновки до розділу 3

У даному розділі було досліджено проблему, того що керування доступом на основі ролей недостатньо для налаштувань політики керування доступом. На основі аналізу представленому в минулому розділі були сформовані вимоги, які не покриває існуюча рольова система керування доступом у CRM та ERP системах платформи Microsoft Dynamics 365, проте які може задовольнити політика керування доступом на основі атрибутів при її впровадженні до систем.

Було розроблено власну систему атрибутів специфічну для систем класу CRM та ERP, яку можна розширювати додатковими атрибутами об’єкту та суб’єкту.

Було розроблено структуру системи керування доступом на основі атрибутів та на основі неї було розроблено власне рішення, яке забезпечує впровадження політики керування доступом на основі атрибутів для Microsoft Dynamics CRM, яке можна застосовувати як замість існуючої рольової системи керування доступом, так і разом з нею задля підвищення рівня безпеки більш гнучких налаштувань. В представленому варіанті система використовується разом з рольовою політикою. Кроками роботи рішення є збір атрибутів, пошук записів правил на предмет наявності правил, що забороняють доступ при таких значеннях атрибутів та якщо доступ заборонено створення запису в спеціальній таблиці аудиту про спробу виконання операції. Встановлення даного рішення та налаштування даних в таблиці «ABAC Rules» забезпечить дію політики керування доступом на основі атрибутів в системі Microsoft Dynamics CRM.

4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Метою цього розділу є створення стартап-проекту для рішення для впровадження керування доступом на основі атрибутів для Microsoft Dynamics CRM, яке може бути універсальним коробковим рішенням та поставлятися замовникам за їх необхідності. Проведення маркетингового аналізу для можливості його ринкового впровадження. Описати ідею, виконати технологічний аудит ідеї, оцінити ринкові можливості запуску розробити ринкову стратегію і маркетингову програму стартап-проекту «ABAC Solution for CRM». Результатами розділу є опис впровадження та реалізації даного проекту. За результатами аналізу описано доцільність його впровадження та кроки для виходу на ринок.

4.1 Опис ідеї проекту

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Рішення для впровадження керування доступом на основі атрибутів для Microsoft Dynamics CRM	Використання ABAC в якості основної політики керування доступом в Microsoft Dynamics CRM	Підвищення ефективності системи керування доступом, можливість більш гнучких налаштувань політики безпеки
	Використання ABAC в якості допоміжної	Підсилення наявної системи

	системи керування доступом разом з RBAC в Microsoft Dynamics CRM	керування доступом, можливість додаткових налаштувань політики безпеки

Для розуміння того, чи мій проект є конкурентноспроможним визначаються його сильні, нейтральні та слабкі сторони порівняно з проектами конкурентів. В якості конкурента було обрано програму в якій використовується керування доступом на основі атрибутів – Pega Sales Automation. Визначення сильних, слабких та нейтральних характеристик ідеї проекту представлено в Таблиці 4.2

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

а п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Pega Sales Automation			
1	Платформа	Microsoft Dynamics 365	Pega Platform			+
2	Можливість застосування для різних операцій	Застосовується як для операцій над	Застосовується тільки для операцій з даними			+

		даними так і процесів				
3	Підтримка правил	Правила налаштовуються та зберігаються безпосередньо в CRM системі	Централізована підтримка правил безпеки без будь-якої розробки інших правил		+	
4	Політика шифрування	Відсутня	Наявна	+		
5	Ієрархічні атрибути	Відсутня, проте розвиваючи проект можливо додати	Наявні	+		
6	Простота адміністрування	Досить проста в адмініструванні	Складна в адмініструванні			+
7	Підтримка складних комплексних правил	Не підтримує досить складні правила, правила з оператором «або»	Підтримує більш складні правила	+		

4.2 Технологічний аудит ідеї проекту

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Налаштування таблиці правил	Microsoft Dynamics CRM	Наявні	Так, ця технологія є доступною
2	Реалізація плагіну	C# .NET	Наявні	Так, ця технологія є доступною
3	Реєстрація плагіну	SDK для Microsoft Dynamics CRM	Наявні	Так, ця технологія є доступною

Технологічна реалізація проекту – можлива. Для реалізації рішення для впровадження керування доступом на основі атрибутів для Microsoft Dynamics CRM було обрано, для виконання налаштувань – CRM систему Microsoft Dynamics 365 Sales, для написання плагіну – мову C# та для реєстрації плагінів в системі – програму XrmToolBox з утилітою Plugin Registration.

4.3 Опис ринкових можливостей запуску стартап-проекту

Таблиця 4.4 Попередня характеристика потенційного ринку стартап-проекту

п/п	Показники стану ринку (найменування)	Характеристика
	Кількість головних гравців, од	1
	Загальний обсяг продаж, грн/ум.од	
	Динаміка ринку (якісна оцінка)	Зростає
	Наявність обмежень для входу (вказати характер обмежень)	Необхідність достідження та можлива оптимізація рішення при існуванні великої кількості правил
	Специфічні вимоги до стандартизації та сертифікації	Проводиться у відповідності до стандартів ISO: ISO/IEC 27001 «Інформаційні технології - Методи забезпечення безпеки - Системи управління інформаційною безпекою - Вимоги».
	Середня норма рентабельності в галузі (або по ринку), %	Не менше 120%

За результатами аналізу можна зробити висновок, що ринок впровадження CRM систем є досить привабливим для входження.

Таблиця 5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Потреба гнучких динамічних налаштувань керування доступом замість наявних статичних	Компанії, що використовують чи планують використовувати продукти на базі Microsoft Dynamics CRM (банки, компанії з продажу, маркетингові компанії, логістичні компанії)	Компанії можуть мати необхідність більш гнучких налаштувань політик безпеки керування доступом для відповідності їх власним політикам безпеки компанії	- дотримання стандартів - простота налаштування - однозначність

Після визначення потенційних клієнтів був проведений аналіз ринкового середовища. Фактори загроз подані в Таблиці 4.6 та фактори можливостей у Таблиці 4.7.

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії

	Обсяг даних	З часом обсяг даних правил може стати досить великим через, що значно виросте час їх обробки	Оптимізація логіки по визначенню санкціонованості операції задля зменшення часу її виконання
	Оновлення CRM системи	CRM система періодично оновлюється і при значному оновленні дане рішення може стати частково некоректним та буде потребувати допрацювань	Слідкувати за оновленнями системи, та за необхідності здійснювати оновлення існуючого рішення.
	Застарівання технологій	Технології з часом застарівають та потребують оновлення	Слідкувати за підтримкою поточних програмних рішень та оновлювати використовувані технології
	Конкуренція	Компанії модернізують свою продукцію та пропонують все більше можливостей для вибору	Модернізація рішення, збільшення функціональних можливостей

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
	Ріст популярності CRM систем	Ріст популярності онлайн-автоматизації процесів та як наслідок CRM систем через ситуацію в світі	Розширення маркетингової компанії для пропонування рішення більшому числу клієнтів
	Можливість розширення функціоналу рішення	Існуюче рішення можливо покращити розширивши існуючий функціонал для формування більш складних правил та додавши новий	Розробка додаткового функціоналу та покращення існуючого

Після визначення факторів загроз і можливостей було проведено аналіз пропозицій та визначено загальні риси конкуренції на ринку. Ступеневий аналіз конкуренції на ринку представлено в Таблиці 4.8.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)

1. Тип конкуренції - чиста	Відсутня монополізація	Розвиток продукту, вдосконалення рішення для підвищення послуг
2. Рівень конкурентної боротьби - глобальний	Конкурентна боротьба здійснюється між компаніями всього світу	Розширення функціональності, відповідність стандартам
3. За галузевою ознакою - міжгалузева	Продукт може використовуватися компаніями з різних галузей та сфер	Розширення сфер пошуку нових клієнтів
4. Конкуренція за видами товарів - товарно-видова	Конкуренція між товарами з подібним функціоналом	Покращувати та розвивати існуючий функціонал
5. Характер конкурентних переваг - нецінова	Ключовим є не ціна, а якість послуг та отриманий функціонал	Покращувати та розвивати існуючий функціонал, надавати ширший спектр послуг(встановлення, налаштування)
6. Інтенсивність - не марочна	Більш важливим є функціонал, а не компанія що його пропонує	Активна маркетингова діяльність

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники

	Pega Sales Automation (проте для іншої CRM системи)	Існує багато компаній, які розробляють та впроваджують рішення для CRM систем за запитом конкретного клієнта	Запропоноване рішення не потребує постачальників	Компанії, що користуються чи планують використовувати CRM систему	Не знайдено інформації про рішення впровадження ABAC для продуктів на базі платформи Microsoft Dynamics 365
Висновки:	Інтенсивність конкурентної боротьби - невисока	Можливість виходу на ринок потенційних конкурентів - середня	-	Клієнти диктують умови ринку	Наразі обмежень немає

За результатами аналізу таблиці був зроблений висновок, що робота на ринку можлива, проте є ймовірність збільшення конкуренції в майбутньому.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)

1	Легкість адміністрування	Проста та зрозуміла у налаштуванні та адмініструванні система
2	Коробкове рішення	Готове рішення, яке майже не вимагає попередніх налаштувань для початку його використання та яке є універсальним та не потребує його модифікацій для впровадження різним клієнтам
3	Невисока собівартість	Рішення не потребує додаткових ресурсів чи використання інших продуктів окрім CRM системи

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін «назва проекту»

п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Легкість адміністрування	15				+			
2	Коробкове рішення	15				+			
3	Невисока собівартість	17				+			

Далі був проведений SWOT-аналіз (матриця аналізу сильних та слабких сторін, загроз та можливостей на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін).

Таблиця 4.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Платформа	Слабкі сторони: Відсутність ієрархічних атрибутів

Можливість застосування для різних операцій Підтримка правил Простота адміністрування Аудит	Відсутність підтримка складних комплексних правил
Можливості: Ріст популярності CRM систем Можливість розширення функціоналу рішення	Загрози: Обсяг даних Оновлення CRM системи Застарівання технологій Конкуренція

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Реалізація більш складного рішення з можливістю впровадження складних правил	Ймовірна	8 міс.
2	Оптимізація та оновлення рішення	Малоймовірна	3 міс.
3	Активна маркетингова компанія	Ймовірна	5 міс

Після аналізу було обрано альтернативу проведення активної маркетингової компанії для збільшення кількості потенційних клієнтів, так як ймовірність отримання ресурсів є більшою в коротші строки реалізації.

4.4 Розроблення ринкової стратегії проекту

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Банки	Висока	Високий	Низька	Висока
2	Медичні установи	Висока	Високий	Низька	Висока
3	Логістичні компанії	Середня	Середній	Низька	Висока
4	Компанії з продажу	Висока	Середній	Низька	Висока
5	Школи	Середня	Низький	Низька	Висока
6	Салони краси	Середня	Низький	Низька	Висока
7	Маркетингові компанії	Середня	Низький	Низька	Висока
Які цільові групи обрано: В якості цільової групи з компаній, що використовують чи планують використовувати продукти на базі Microsoft					

Dynamics CRM було обрано такі, як банки, медичні установи, логістичні компанії, так як для них більш важлива безпека даних, а тому вони з більшою ймовірністю будуть зацікавлені у даному рішенні. Отже, буде використана стратегія диференційованого маркетингу.

Після обрання сегментів ринку була сформована базову стратегія розвитку, яка наведена в Таблиці 4.15.

Таблиця 4.15 – Визначення базової стратегії розвитку

п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
	Активна маркетингова компанія	Стратегія диференційованого маркетингу	Реклама перстпективи, та якість рішення	Стратегія диференціації

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати	Чи буде компанія копіювати основні характеристики товару	Стратегія конкурентної поведінки

		існуючих у конкурентів?	конкурента, і які?	
	Так, для платформи Microsoft Dynamics 365	Так	Ні	Стратегія розширення первинного попиту

Таблиця 4.17 – Визначення стратегії позиціонування

п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Забезпечення безпеки та відповідність стандартам	Стратегія диференціації	Стабільність системи, простота адміністрування	Гнучкість системи керування доступом на основі атрибутів, Можливість комбінування двух систем керування доступом (RBAC та ABAC) Готовність до модернізації та постійна підтримка клієнтів

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Потреба гнучких динамічних налаштувань керування доступом замість наявних статичних	Впровадження керування доступом на основі атрибутів замість або разом з наявною рольовою політикою керування доступом	На базі Microsoft Dynamics 365 відомі конкуренти відсутні

Таблиця 4.18 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Потреба гнучких динамічних налаштувань керування доступом замість наявних статичних задовольняється впровадженням керування доступом на основі атрибутів замість або разом з наявною рольовою політикою керування доступом		
II. Товар у реальному виконанні	Властивості/характеристики	М/ Нм	Вр/Тх /Тл/Е/Ор
	1. Гнучка політика безпеки 2. Легкість адміністрування		

	3. Аудит неуспішних операцій		
	Якість: необхідно продовжити тестування		
	Постачається в zip-архіві з додатком в якості інструкції з встановлення та налаштування		
	Марка: ABAC Solution for Microsoft Dynamics CE		
III. Товар із підкріпленням	До продажу розробка та оптимізація рішення, навантажувальне тестування		
	Після продажу проведення рекламної компанії, підтримка клієнтів		
За рахунок чого потенційний товар буде захищено від копіювання: право на захист інтелектуальної власності			

Таблиця 4.19 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	-	$140- $170 на місяць	Високий	$60- $150 на місяць

Таблиця 4.20 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту

	Власна система збуту	Усні та письмові консультації, встановлення рішення за потреби	Канал нульового рівня – продаж безпосередньо користувачам	Прямий збут без посередників

Таблиця 4.21 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Дослідження властивостей рішення, тестування продукту, прийняття рішення про необхідність продукту	Соціальні мережі, реклама, конференції за тематикою	Поєднання технологій, гнучкість, легкість адміністрування, постійна підтримка	Показати переваги та необхідність продукту	Розробка маркетингових повідомлень для демонстрації якості та необхідності продукту

Висновки до розділу 4

У розділі розробки стартап-проекту було виконано опис ідеї, проведено технологічний аудит ідеї проекту, проаналізовано ринкові можливості запуску стартап-проекту, визначено фактори конкурентоспроможності, розроблено ринкову та маркетингові стратегії. В результаті проведеного аналізу можна зробити висновки, що рішення по впровадженню керування доступом на основі атрибутів повинне мати попит на ринку та можлива ринкова капіталізація проекту, проте необхідна буде постійна підтримка актуальності продукту. Було обрано таку альтернативу ринкової поведінки: проведення активної маркетингової компанії задля розширення кола потенційних клієнтів. Дана альтернатива забезпечить, що при виході системи на ринок вона вже буде мати попит. Тому вважаю, що подальше розроблення системи є доцільним.

ВИСНОВКИ

В магістерській дисертації було досліджено найбільш популярні політики керування доступом – DAC, MAC, RBAC, ABAC. У представлених в роботі системах – Microsoft Dynamics AX та Microsoft Dynamics CRM використовуються рольова політика керування доступом з деякими особливостями, описаними у роботі. Проаналізувавши системи безпеки даних інформаційних систем класу CRM та ERP було виявлено ряд вимог, які не покривають наявні системи захисту.

Задля покращення існуючої системи керування доступом, а саме надання гнучкості налаштувань політики безпеки та можливості динамічного прийняття рішень, системою керування доступом на основі атрибутів було розроблено власну систему атрибутів специфічну для CRM та ERP систем, яку можна використовувати для розробки рішення для керування доступом на основі атрибутів у системах CRM та ERP на платформі Microsoft Dynamics 365. Також було розроблено структуру рішення для впровадження даної системи атрибутів та політики доступу на основі атрибутів до CRM системи Microsoft Dynamics CRM, та реалізовано рішення на практиці, поточна версія якого вже готова для встановлення та використання на інших екземплярах системи. Було продемонстровано роботу рішення на декількох прикладах та запропоновано кроки до подальшого покращення даного рішення.

За результатами розробки стартап-проекту можна зробити висновки, що розроблене рішення повинне мати попит на ринку, тому що на ринку не існує рішення з аналогічним функціоналом, що базуються на досить популярній платформі Microsoft Dynamics CRM. Обрана альтернатива ринкової поведінки при виході системи на ринок забезпечить стабільність її роботи. Тому слід вважати, що подальша розробка системи є доцільною.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Cleo Team Understanding the Need for CRM and ERP Systems Integration [Електронний ресурс] – Режим доступу до ресурсу: https://www.cleo.com/blog/knowledge-base-erp-and-crm-integration/
2. K.Vijayalakshmi Improving Performance of ABAC Security Policies Validation usinga Novel Clustering Approach [Text] / K.Vijayalakshmi Dr.V.Jayalakshmi – 2021.
3. Буров Є.В ВИКОРИСТАННЯ МОДЕЛЕЙ ДЛЯ КЕРУВАННЯ ДОСТУПОМ ДО РЕСУРСІВ ІНТЕЛЕКТУАЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ [Text] / Буров Є.В, Гульова А.В – 2010
4. Дискреційна політика безпеки [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.com.ua/1_333126_diskretsiyna-politika-bezpeki.html
5. Mandatory Access Control vs Discretionary Access Control: Which to Choose? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ekransystem.com/en/blog/mac-vs-dac
6. David F. Ferraiolo Role-Based Access Controls [Text] / David F. Ferraiolo, D. Richard Kuhn National Institute of Standards and Technology – 1992
7. Mandatory, Discretionary, Role and Rule Based Access Control [Електронний ресурс] – Режим доступу до ресурсу: https://www.techotopia.com/index.php/Mandatory,_Discretionary,_Role_and_Rule_Based_Access_Control
8. Мандатна політика безпеки [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.com.ua/1_333127_mandatna-politika-bezpeki.html
9. Керування доступом на основі ролей [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Керування_доступом_на_основі_ролей

10. Ellen Zhang What is Role-Based Access Control (RBAC)? Examples, Benefits, and More [Електронний ресурс] – Режим доступу до ресурсу: https://digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more – 2020

11. Dnsstuff RBAC vs. ABAC: What's the Difference? [Електронний ресурс] – Режим доступу до ресурсу: https://www.dnsstuff.com/rbac-vs-abac-access-control

12. Neil Madden API Security in Action [Text] / Neil Madden – 2020

13. NIST Guide to Attribute Based Access Control (ABAC) Definition and Considerations [Text] /Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone – 2014 – https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf

14. David Reid CRM vs ERP: What's the difference? [Електронний ресурс] – Режим доступу до ресурсу: https://www.sysco-software.com/crm-vs-erp/

15. Barney Beal CRM vs ERP: What's the Difference? [Електронний ресурс] – Режим доступу до ресурсу: https://www.netsuite.com/portal/resource/articles/erp/erp-vs-crm.shtml

16. 10 Best ERP Software Of 2021 [Електронний ресурс] – Режим доступу до ресурсу: https://peoplemanagingpeople.com/tools/best-erp-software/

17. Microsoft Security model concepts [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dynamics365/customerengagement/on-premises/admin/security-concepts?view=op-9-1/

18. Mahender Pal. Implementing Microsoft Dynamics 365 Customer Engagement [Text] / Mahender Pal. – 2020.

19. Done Microsoft Dynamics AX [Електронний ресурс] – Режим доступу до ресурсу: https://done.ua/rus/produkti/sistemi-upravlinnya-pidpriemstvom-erp/microsoft-dynamics-ax/

20. Microsoft Security architecture [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/sysadmin/security-architecture

21. The Microsoft Dynamics AX Team Inside Microsoft Dynamics AX 2012 R3 [Text] / The Microsoft Dynamics AX Team – 2012

22. Microsoft Role-based security [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/sysadmin/role-based-security/

23. NEXTLABS The Definitive Guide to Attribute-Based Access Control (ABAC) [Електронний ресурс] – Режим доступу до ресурсу: https://www.nextlabs.com/products/technology/abac/

24. Get started with a Dynamics 365 free trial [Електронний ресурс] – Режим доступу до ресурсу: https://dynamics.microsoft.com/en-us/dynamics-365-free-trial/

25. Microsoft.CrmSdk.CoreAssemblies [Електронний ресурс] – Режим доступу до ресурсу: https://www.nuget.org/packages/Microsoft.CrmSdk.CoreAssemblies/

ДОДАТОК А КОД ЗБІРКИ ПЛАГІНУ

ABACSolution.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net462</TargetFramework>
    <SignAssembly>true</SignAssembly>
    <AssemblyOriginatorKeyFile>key.snk</AssemblyOriginatorKeyFile>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.CrmSdk.CoreAssemblies" Version="9.0.2.42" />
  </ItemGroup>

</Project>
```

Plugin.cs

```
using ABACSolution.Enums;
using ABACSolution.Extensions;
using ABACSolution.Models;
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Query;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;

namespace ABACSolution
{
    Public class Plugin : IPlugin
    {
        public void Execute(IServiceProvider serviceProvider)
        {
            var tracer =
(ITracingService)serviceProvider.GetService(typeof(ITracingService));
            var context =
(IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));
            var notificationService =
(IServiceEndpointNotificationService)serviceProvider.GetService(typeof(IServiceEndpointNo
tificationService));
            var serviceFactory =
(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactor
y));
            var service = serviceFactory.CreateOrganizationService(null);

            try
            {
                tracer.Trace($"Message name: {context.MessageName}");
```

```
                if (!MakeDecision(service, context, tracer, out var attributes, out var
rule))
                {
                    CreateLog(service, attributes, rule);
                    throw new Exception("Access denied!");
                }

            }
            catch (Exception exception)
            {
                throw new InvalidPluginExecutionException(exception.Message);
            }
        }

        private void CreateLog(IOrganizationService service, AttributeModel
attributesModel, EntityReference ruleRef)
        {
            var log = new Entity("dl_abacaudit")
            {
                [ABACAuditModel.User] = attributesModel.UserEntity?.ToEntityReference(),
                [ABACAuditModel.Time] = DateTime.UtcNow,
                [ABACAuditModel.Action] = attributesModel.Action,
                [ABACAuditModel.Rule] = ruleRef,
                [ABACAuditModel.Result] = new OptionSetValue((int)ResultEnum.Deny)
            };

            if (attributesModel.Action == "Create")
            {
                log[ABACAuditModel.RecordId] =
attributesModel.ObjectEntity?.Id.ToString();
                log[ABACAuditModel.EntityName] =
attributesModel.ObjectEntity?.LogicalName;
            }
            else
            {
                log[ABACAuditModel.Record] =
attributesModel.ObjectEntity?.ToEntityReference();
            }

            var id = service.Create(log);
        }

        private bool MakeDecision(IOrganizationService service, IPluginExecutionContext
context, ITracingService tracer, out AttributeModel attributes, out EntityReference
ruleRef)
        {
            attributes = new AttributeModel();

            GetAttributes(service, context, attributes);

            tracer.Trace($"Attributes: {JsonSerializer.Serialize(attributes)}");

            var rules = GetRules(service, attributes.Action);

            tracer.Trace($"Found {attributes} active rules for this action.");

            ruleRef = null;

            foreach (var rule in rules)
            {
```

```
                if (rule.GetAttributeValue<EntityReference>(ABACRuleModel.User)?.Id is
var userId && (userId == null || userId != attributes.User))
                    continue;

                if (rule.GetAttributeValue<EntityReference>(ABACRuleModel.UserTeam)?.Name
is var userTeam && (userTeam == null || !attributes.UserTeams.Contains(userTeam)))
                    continue;

                if
(rule.GetAttributeValue<EntityReference>(ABACRuleModel.UserBusinessUnit)?.Id is var
userUnitId && (userUnitId == null || userUnitId != attributes.UserBussinessUnit))
                    continue;

                if (rule.GetAttributeValue<EntityReference>(ABACRuleModel.UserRole)?.Name
is var userRole && (userRole == null || !attributes.UserRoles.Contains(userTeam)))
                    continue;

                if
(rule.GetAttributeValue<EntityReference>(ABACRuleModel.UserPosition)?.Id is var position
&& (position == null || position != attributes.UserPosition))
                    continue;

                if (rule.GetAttributeValue<string>(ABACRuleModel.UserCountry) is var
country && (country == null || country != attributes.UserCountry))
                    continue;

                if (rule.GetAttributeValue<string>(ABACRuleModel.UserCity) is var city &&
(city == null || city != attributes.UserCity))
                    continue;

                if (rule.GetAttributeValue<string>(ABACRuleModel.UserAttributeName) is
var userAttributeName && userAttributeName != null
                    &&
rule.GetAttributeValue<OptionSetValue>(ABACRuleModel.UserAttributeName)?.Value is var
userAttributeOperator && userAttributeOperator != null
                    && rule.GetAttributeValue<string>(ABACRuleModel.UserAttributeName) is
var userAttributeValue && userAttributeValue != null
                    && attributes.UserEntity.Contains(userAttributeName) &&
attributes.UserEntity.GetAttributeValue<object>(userAttributeName) is var value)
                {
                    if (userAttributeOperator == (int)AttributeOperatorEnum.Equals &&
userAttributeValue != value.ToString())
                        continue;

                    if (userAttributeOperator == (int)AttributeOperatorEnum.DoesNotEqual
&& userAttributeValue == value.ToString())
                        continue;

                    if (userAttributeOperator == (int)AttributeOperatorEnum.IsGreaterThan
&& int.Parse(userAttributeValue) >= int.Parse(value.ToString()))
                        continue;

                    if (userAttributeOperator ==
(int)AttributeOperatorEnum.IsGreaterThanorEqualTo && int.Parse(userAttributeValue) >
int.Parse(value.ToString()))
                        continue;

                    if (userAttributeOperator == (int)AttributeOperatorEnum.IsLessThan &&
int.Parse(userAttributeValue) <= int.Parse(value.ToString()))
                        continue;
```

```
                    if (userAttributeOperator ==
(int)AttributeOperatorEnum.IsLessThanorEqualTo && int.Parse(userAttributeValue) <
int.Parse(value.ToString()))
                        continue;

                    if (userAttributeOperator == (int)AttributeOperatorEnum.Contains &&
!value.ToString().Contains(userAttributeValue))
                        continue;

                    if (userAttributeOperator ==
(int)AttributeOperatorEnum.DoesNotContain &&
value.ToString().Contains(userAttributeValue))
                        continue;
                }

                if (rule.GetAttributeValue<string>(ABACRuleModel.Entity) is var entity &&
(entity == null || entity != attributes.
                    Entity))
                    continue;

                if (rule.GetAttributeValue<string>(ABACRuleModel.Column) is var column &&
(column == null || !attributes.Columns.Contains(column)))
                    continue;

                if (rule.GetAttributeValue<int?>(ABACRuleModel.StatusReason) is var
status && (status == null || status != attributes.StatusReason))
                    continue;

                if (rule.GetAttributeValue<EntityReference>(ABACRuleModel.Owner)?.Id is
var ownerId && (ownerId == null || ownerId != attributes.Owner))
                    continue;

                if (rule.GetAttributeValue<string>(ABACRuleModel.ObjectAttributeName) is
var objectAttributeName && userAttributeName != null
                    &&
rule.GetAttributeValue<OptionSetValue>(ABACRuleModel.ObjectAttributeName)?.Value is var
objectAttributeOperator && objectAttributeOperator != null
                    && rule.GetAttributeValue<string>(ABACRuleModel.ObjectAttributeName)
is var objectAttributeValue && objectAttributeValue != null
                    && attributes.ObjectEntity.Contains(objectAttributeName) &&
attributes.ObjectEntity.GetAttributeValue<object>(objectAttributeName) is var val)
                {
                    if (objectAttributeOperator == (int)AttributeOperatorEnum.Equals &&
objectAttributeValue != val.ToString())
                        continue;

                    if (objectAttributeOperator ==
(int)AttributeOperatorEnum.DoesNotEqual && objectAttributeValue == val.ToString())
                        continue;

                    if (objectAttributeOperator ==
(int)AttributeOperatorEnum.IsGreaterThan && int.Parse(objectAttributeValue) >=
int.Parse(val.ToString()))
                        continue;

                    if (objectAttributeOperator ==
(int)AttributeOperatorEnum.IsGreaterThanorEqualTo && int.Parse(objectAttributeValue) >
int.Parse(val.ToString()))
                        continue;
```

```
                    if (objectAttributeOperator == (int)AttributeOperatorEnum.IsLessThan
&& int.Parse(objectAttributeValue) <= int.Parse(val.ToString()))
                        continue;

                    if (objectAttributeOperator ==
(int)AttributeOperatorEnum.IsLessThanorEqualTo && int.Parse(objectAttributeValue) <
int.Parse(val.ToString()))
                        continue;

                    if (objectAttributeOperator == (int)AttributeOperatorEnum.Contains &&
!val.ToString().Contains(objectAttributeValue))
                        continue;

                    if (objectAttributeOperator ==
(int)AttributeOperatorEnum.DoesNotContain &&
val.ToString().Contains(objectAttributeValue))
                        continue;
                }

                if
((rule.GetAttributeValue<OptionSetValue>(ABACRuleModel.TimeOperator)?.Value is var
timeOperator && timeOperator == null)
                    || (rule.GetAttributeValue<string>(ABACRuleModel.Time) is var time &&
time == null)
                    || (timeOperator == (int)TimeOperatorEnum.IsGreaterThan &&
DateTime.ParseExact(time, "HH:mm",
                                    CultureInfo.InvariantCulture).TimeOfDay >
attributes.Time.TimeOfDay)
                    || (timeOperator == (int)TimeOperatorEnum.IsLessThan &&
DateTime.ParseExact(time, "HH:mm",
                                    CultureInfo.InvariantCulture).TimeOfDay <
attributes.Time.TimeOfDay)
                    || (timeOperator == (int)TimeOperatorEnum.InPeriod && time.Split('-
').ToList() is var times &&
                            (DateTime.ParseExact(times.First(), "HH:mm",
CultureInfo.InvariantCulture).TimeOfDay > attributes.Time.TimeOfDay
                            || DateTime.ParseExact(times.Last(), "HH:mm",
CultureInfo.InvariantCulture).TimeOfDay < attributes.Time.TimeOfDay)))
                    continue;

                if
(rule.GetAttributeValue<OptionSetValueCollection>(ABACRuleModel.DayOfWeek) is var
dayOfWeek && (dayOfWeek == null || dayOfWeek.Count == 0 || dayOfWeek.Select(d =>
d.Value).Contains(102900000 + attributes.DayOfWeek)))
                    continue;

                if (rule.GetAttributeValue<OptionSetValue>(ABACRuleModel.Result)?.Value
is var result && result == (int)ResultEnum.Deny)
                {
                    ruleRef = rule.ToEntityReference();

                    return false;
                }
            }

            return true;
        }

        private void GetAttributes(IOrganizationService service, IPluginExecutionContext
context, AttributeModel attributes)
        {
```

```
            GetSubjectAttributes(service, context, attributes);
            GetObjectAttributes(service, context, attributes);
            GetOperationAttributes(context, attributes);
            GetEnvironmentAttributes(attributes);
        }

        private void GetOperationAttributes(IPluginExecutionContext context,
AttributeModel attributesModel)
        {
            attributesModel.Action = context.MessageName;
        }

        private void GetEnvironmentAttributes(AttributeModel attributesModel)
        {
            attributesModel.Time = DateTime.Now;
            attributesModel.DayOfWeek = (int)DateTime.Now.DayOfWeek;
        }

        private void GetSubjectAttributes(IOrganizationService service,
IPluginExecutionContext context, AttributeModel attributesModel)
        {
            var userId = context.UserId;
            var user = service.Retrieve("systemuser", userId, new ColumnSet(true));

            attributesModel.User = userId;
            attributesModel.UserBussinessUnit =
user.GetAttributeValue<EntityReference>("businessunitid")?.Id;
            attributesModel.UserRoles = GetUserSecurityRoles(service, userId);
            attributesModel.UserTeams = GetUserTeams(service, userId);
            attributesModel.UserPosition =
user.GetAttributeValue<EntityReference>("positionid")?.Id;
            attributesModel.UserCountry =
user.GetAttributeValue<string>("address1_country");
            attributesModel.UserCity = user.GetAttributeValue<string>("address1_city");
            attributesModel.UserEntity = user;
        }

        private void GetObjectAttributes(IOrganizationService service,
IPluginExecutionContext context, AttributeModel attributesModel)
        {
            if (!context.InputParameters.Contains("Target") &&
!context.InputParameters.Contains("LeadId")) return;

            Entity target = null;
            EntityReference targetRef = null;

            if (context.InputParameters.Contains("Target") &&
context.InputParameters["Target"].GetType().Equals(typeof(Entity)))
            {
                target = (Entity)context.InputParameters["Target"];
            }
            else if (context.InputParameters.Contains("Target") &&
context.InputParameters["Target"].GetType().Equals(typeof(EntityReference)))
            {
                targetRef = (EntityReference)context.InputParameters["Target"];
                target = attributesModel.Action != "Delete" ?
service.Retrieve(targetRef.LogicalName, targetRef.Id, new ColumnSet(true)) : new
Entity(targetRef.LogicalName, targetRef.Id);
            }
```

```
            attributesModel.Columns = targetRef == null ? target.Attributes.Keys.ToList()
: null;

            if (context.PreEntityImages.Contains("Image") &&
context.PreEntityImages["Image"] is var preImage)
                target = preImage.Merge(target);

            if (context.InputParameters.Contains("LeadId"))
            {
                var leadRef = (EntityReference)context.InputParameters["LeadId"];
                target = service.Retrieve(leadRef.LogicalName, leadRef.Id, new
ColumnSet(true));
            }

            attributesModel.Entity = target.LogicalName;

            attributesModel.StatusReason =
target.GetAttributeValue<OptionSetValue>("statuscode")?.Value;
            attributesModel.Owner =
target.GetAttributeValue<EntityReference>("ownerid").Id;
            attributesModel.ObjectEntity = target;
        }

        public List<string> GetUserSecurityRoles(IOrganizationService service, Guid
userId)
        {
            var query = new QueryExpression("role")
            {
                ColumnSet = new ColumnSet("name")
            };

            var link = query.AddLink("systemuserroles", "roleid", "roleid");

            var linkEntity = link.AddLink("systemuser", "systemuserid", "systemuserid");

            linkEntity.LinkCriteria.AddCondition("systemuserid", ConditionOperator.Equal,
userId);

            var userRoles = service.RetrieveMultiple(query).Entities.Select(e =>
e["name"].ToString()).ToList();

            return userRoles;
        }

        public List<string> GetUserTeams(IOrganizationService service, Guid userId)
        {
            var query = new QueryExpression("team")
            {
                ColumnSet = new ColumnSet("name")
            };

            var link = query.AddLink("teammembership", "teamid", "teamid");

            var linkEntity = link.AddLink("systemuser", "systemuserid", "systemuserid");

            linkEntity.LinkCriteria.AddCondition("systemuserid", ConditionOperator.Equal,
userId);

            var userRoles = service.RetrieveMultiple(query).Entities.Select(e =>
e["name"].ToString()).ToList();
```

```
            return userRoles;
        }

        public List<Entity> GetRules(IOrganizationService service, string action)
        {
            var query = new QueryExpression("dl_abacrule")
            {
                ColumnSet = new ColumnSet(true)
            };

            query.Criteria.AddCondition(ABACRuleModel.Action, ConditionOperator.Equal,
action);
            query.Criteria.AddCondition(ABACRuleModel.Status, ConditionOperator.Equal,
0); // Active

            var rules = service.RetrieveMultiple(query).Entities.ToList();

            return rules;
        }
    }
}
```

ABACRuleModel.cs

```
namespace ABACSolution.Models
{
    public class ABACRuleModel
    {
        // subject attribute
        public static string User = "dl_user";
        public static string UserPosition = "dl_userspositionid";
        public static string UserBusinessUnit = "dl_usersbusinessunitid";
        public static string UserRole = "dl_userroleid";
        public static string UserTeam = "dl_usersteamid";
        public static string UserCountry = "dl_userscountry";
        public static string UserCity = "dl_userscity";
        public static string UserAttributeName = "dl_userattributename";
        public static string UserAttributeOperator = "dl_userattributeoperator";
        public static string UserAttributeValue = "dl_userattributevalue";
        // object attribute
        public static string Entity = "dl_entity";
        public static string Column = "dl_column";
        public static string StatusReason = "dl_statusreason";
        public static string Owner = "dl_owner";
        public static string ObjectAttributeName = "dl_objectattributename";
        public static string ObjectAttributeOperator = "dl_objecctattributeoperator";
        public static string ObjectAttributeValue = "dl_objectattributevalue";
        // operation attribute
        public static string Action = "dl_action";
        // environment attribute
        public static string Time = "dl_time";
        public static string DayOfWeek = "dl_dayofweek";

        public static string Result = "dl_resultcode";

        public static string Status = "statecode";
    }
}
```

ABACAuditModel.cs

```
namespace ABACSolution.Models
{
    public class ABACAuditModel
    {

        public static string User = "dl_userid";

        public static string Record = "dl_recordid";

        public static string Action = "dl_action";

        public static string Time = "dl_executiontime";

        public static string Rule = "dl_ruleid";

        public static string Result = "dl_resultcode";
    }
}
```

AttributeModel.cs

```
using Microsoft.Xrm.Sdk;
using System;
using System.Collections.Generic;

namespace ABACSolution.Models
{
    public class AttributeModel
    {
        // subject attribute
        public Guid User { get; set; }
        public Guid? UserPosition { get; set; }
        public Guid? UserBussinessUnit { get; set; }
        public List<string> UserRoles { get; set; }
        public List<string> UserTeams { get; set; }
        public string UserCountry { get; set; }
        public string UserCity { get; set; }
        public Entity UserEntity { get; set; }
        // object attribute
        public string Entity { get; set; }
        public List<string> Columns { get; set; }
        public int? StatusReason { get; set; }
        public Guid? Owner { get; set; }
        public Entity ObjectEntity { get; set; }
        // operation attribute
        public string Action { get; set; }
        // environment attribute
        public DateTime Time { get; set; }
        public int DayOfWeek { get; set; }
    }
}
```

ResultEnum.cs

```
namespace ABACSolution.Enums
{
    public enum ResultEnum
    {
        Allow = 102900000,
        Deny = 102900001
    }
}
```

AttributeOperatorEnum.cs

```
namespace ABACSolution.Enums
{
    public enum AttributeOperatorEnum
    {
        Equals = 102900000,
        DoesNotEqual = 102900001,
        IsGreaterThan = 102900002,
        IsGreaterThanorEqualTo = 102900003,
        IsLessThan = 102900004,
        IsLessThanorEqualTo = 102900005,
        Contains = 102900006,
        DoesNotContain = 102900007
    }
}
```

TimeOperatorEnum.cs

```
namespace ABACSolution.Enums
{
    public enum TimeOperatorEnum
    {
        IsLessThan = 102900000,
        IsGreaterThan = 102900001,
        InPeriod = 102900002
    }
}
```

JsonSerializer.cs

```
using System.IO;
```

```
using System.Runtime.Serialization;
using System.Runtime.Serialization.Json;
using System.Text;

namespace ABACSolution.Extensions
{
    public static class JsonSerializer
    {
        private static DataContractJsonSerializer GetSerializer<T>() => new
DataContractJsonSerializer(typeof(T), new DataContractJsonSerializerSettings {
DateTimeFormat = new DateTimeFormat("o") });

        public static string Serialize<T>(T value)
        {
            using (var memoryStream = new MemoryStream())
            {
                var serializer = GetSerializer<T>();
                serializer.WriteObject(memoryStream, value);
                return Encoding.UTF8.GetString(memoryStream.ToArray());
            }
        }
    }
}
```

EntityExtension.cs

```
using Microsoft.Xrm.Sdk;

namespace ABACSolution.Extensions
{
    public static class EntityExtension
    {
        public static Entity Merge(this Entity target, Entity source)
        {
            var mergedEntity = new Entity(target.LogicalName, target.Id);

            foreach (var attribute in target.Attributes)
                mergedEntity[attribute.Key] = attribute.Value;

            foreach (var attribute in source.Attributes)
                mergedEntity[attribute.Key] = attribute.Value;

            return mergedEntity;
        }
    }
}
```