Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електроенергетики

Факультет інформаційних технологій

Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеню бакалавра

студента *Василькова Євгена Костянтиновича*

академічної групи *125-17-1*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему ***Способи реалізації послуг безпеки для WEB-додатків при використанні фреймворків Django та Flask***

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	професор Кагадій Т.С.			
розділів:				
спеціальний	ст. викл. Кручінін О.В.			
економічний	к. е. н., доц. Пілова Д.П.			
Рецензент				
Нормоконтролер				

Дніпро
2021

ЗАТВЕРДЖЕНО:

завідувач кафедри

безпеки інформації та телекомунікацій

_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ

на кваліфікаційну роботу ступеня бакалавра

студенту ***Василькову Євгену Костянтиновичу*** **академічної групи** ***125-17-1***

(прізвище ім'я по-батькові) (шифр)

спеціальності ***125 Кібербезпека***

спеціалізації _____

на тему ***Способи реалізації послуг безпеки для WEB-додатків при використанні фреймворків Django та Flask***

Затверджену наказом ректора НТУ «Дніпровська політехніка» від 07.06.2021 № 317-с

Розділ	Зміст	Термін виконання
Розділ 1	*Визначено актуальність забезпечення захисту WEB-додатків. Проаналізовано технології розробки WEB-додатків. Наведені вимоги щодо послуг безпеки.*	09.02.2021-10.03.2021
Розділ 2	*Виконано аналіз рівнів послуг безпеки, які реалізуються при застосуванні фреймворків. Виконано порівняльний аналіз фреймворків.*	12.03.2021-29.04.2021
Розділ 3	*Економічне обґрунтування доцільності впровадження запропонованих у проекті рішень*	04.05.2021-27.05.2021

Завдання видано _____ *Кагадій Т.С.*

(підпис керівника) (прізвище, ініціали)

Дата видачі: *05.01.2021р.*

Дата подання до екзаменаційної комісії: *11.06.2021р.*

Прийнято до виконання _____ *Васильков Є. К.*

(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 101 ст., містить 31 рис., 1 табл., 8 додатків, 16 джерел.

Предмет дослідження: способи реалізації послуг безпеки для WEB-додатків при використанні фреймворків Django та Flask.

Мета проекту: оцінити рівні послуги безпеки які можливо реалізувати із застосуванням фреймворків Django та Flask.

У першому розділі проаналізовано актуальність забезпечення безпеки WEB-додатків, вимоги до безпеки WEB-додатків, завдання забезпечення безпеки WEB-додатків на різних етапах створення проектів, основні ризики безпеки WEB-додатків. Також виконано: оцінка переваг та недоліків фреймворків Django та Flask та показано їх актуальність для забезпечення безпеки WEB-додатків.

У другому розділі описано послуги безпеки інформації WEB-сторінок, механізми захисту інформації в Django, послуги безпеки, що реалізовує Django. З'ясовано, що фреймворк Django спрощує завдання забезпечення WEB-додатку безпекою відмінно від Flask, який надає теж високий рівень захисту але він більш гибкий та потребує більшого контролю при написанні коду.

В третьому розділі були розраховані витрати на впровадження та річну підтримку політики безпеки інформації. Також, було з'ясовано, що запропоновані способи реалізації безпеки є економічно вигідними для кінцевого програмного продукту.

Результати дослідження можуть бути застосовані в розробці WEB-додатків.

WEB-ДОДАТОК, ФРЕЙМВОРК, РЕАЛІЗАЦІЯ ПОСЛУГ БЕЗПЕКИ, DJANGO, FLASK

РЕФЕРАТ

Пояснительная записка: 101 стр., содержит 31 рис., 1 табл., 8 приложений, 16 источников.

Предмет исследования: оценить уровни услуг безопасности, которые возможно реализовать с применением фреймворков Django и Flask.

Цель проекта: оценить услуги безопасности фреймворков Django и Flask, показать их реализацию при создании WEB-приложений.

В первой главе проанализировано актуальность обеспечения безопасности WEB-приложений, требования к безопасности WEB-приложений, задача обеспечения безопасности WEB-приложений на различных этапах создания проектов, основные риски безопасности WEB-приложений. Также выполнено: оценка преимуществ и недостатков фреймворков Django и Flask и показано их актуальность для обеспечения безопасности WEB-приложений.

Во втором разделе описаны услуги безопасности информации WEB-страниц, механизмы защиты информации в Django, услуги безопасности Django. Выяснено, что фреймворк Django, упрощает задачу обеспечения WEB-приложения безопасностью в отличии от Flask, который предоставляет тоже высокий уровень защиты но он более гибкий и требует большего контроля в написании кода.

В третьем разделе были рассчитаны затраты на внедрение и годовую поддержку политики безопасности информации. Также, было выяснено, что предложенные способы реализации безопасности являются экономически выгодными для конечного программного продукта.

Результаты исследования могут быть применены в разработке WEB-приложений.

WEB-ПРИЛОЖЕНИЕ, ФРЕЙМВОРКИ, DJANGO, FLASK, РЕАЛИЗАЦИЯ УСЛУГ БЕЗОПАСНОСТИ

ABSTRACT

Explanatory note: 90 pages, contains 31 figures, 1 tables, 8 appendices, 16 sources.

Subject of research: ways to implement security services for WEB applications using the Django and Flask frameworks.

The goal of the project: assess the levels of security services that can be implemented using the Django and Flask frameworks.

The first section analyzes the relevance of security of WEB-applications, requirements for security of WEB-applications, the task of ensuring the security of WEB-applications at different stages of project creation, the main security risks of WEB-applications. Also performed: assessment of the advantages and disadvantages of Django and Flask frameworks and shows their relevance for the security of WEB-applications.

The second section describes information security services for WEB pages, information security mechanisms in Django, and Django security services. It was found that the Django framework simplifies the task of providing a WEB application with security, unlike Flask, which also provides a high level of protection, but it is more flexible and requires more control in writing code.

In the third section, the costs of implementing and maintaining the security policy annually were calculated. Also, it was found that the proposed methods of implementing security are cost-effective for the final software product.

The research results can be applied in the development of WEB-applications.

WEB-APP, FRAMEWORKS, DJANGO, FLASK, SECURITY SERVICES REALIZATION

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface;

HTTP – HyperText Transfer Protocol;

OWASP – Open Web Application Security Project;

SQL – Structured query language;

WEB – World Wide Web;

XML – eXtensible Markup Language;

АС – автоматизована система;

КСЗІ – комплексна система захисту інформації;

НД ТЗІ – нормативний документ в галузі технічного захисту інформації;

НСД – несанкціонований доступ;

ПЗ – програмне забезпечення;

СЗІ – служба захисту інформації.

ЗМІСТ

ВСТУП

В сучасному світі людство завдяки інтернету щоденно користується різноманітними інформаційними послугами, які надають компанії, установи, організації тощо. Майже жодна людина не уявляє своє життя без швидкого отримання та передачі конфіденційної інформації.

З кінця 90-х початку 2000-х років для обміну інформацією в мережі інтернету почали широко використовуватися WEB-додатки, в яких клієнт взаємодіє з WEB-сервером, саме там зберігається конфіденційна інформація. Для забезпечення безпеки цієї інформації необхідно захищати не тільки сервера, а й WEB-додатки. За деякими оцінками кожного дня зламують від 30000 до 50000 WEB-додатків. Кількість проникнень зростає з кожним днем.

Саме тому безпека WEB-додатків стала першочерговим завданням для підприємств всіх форм і розмірів. Незалежно від того, чи це транснаціональна медіа-компанія, яка займається потоковою передачею контенту, або невеликий стартап з віддаленими співробітниками, все рівно бізнес знаходиться в мережі, що піддає ризику клієнтів і власні конфіденційні дані. Хоча розробка WEB-додатків поліпшила методи ведення бізнесу компаніями, вони також підвищили ризик зловмисних атак.

Таким чином, метою цієї роботи є проведення аналізу способів реалізації послуг безпеки для WEB-додатків, а саме що створюються за допомогою фреймворків Django та Flask.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття WEB-додатку як програмного продукту

WEB-додаток - це будь-яка комп'ютерна програма, яка виконує певну функцію, використовуючи WEB-браузер у якості свого клієнта. Додаток може бути таким же простим, як дошка оголошень або контактна форма на WEB-сайті, або настільки ж складним, як текстовий редактор або багатокористувацький мобільний ігровий додаток, який завантажується на телефон.

WEB-додатки існують з тих пір, коли всесвітня павутина набула популярності. Наприклад, Ларрі Уолл ще у 1987 році розробив Perl - популярну мову сценаріїв на стороні сервера. Це було за сім років до того, як Інтернет дійсно почав набирати популярність поза академічними та технологічними колами.

Перші основні WEB-додатки були відносно простими, але наприкінці 90-х років відбувся поштовх до більш складних WEB-додатків. У наш час мільйони людей використовують WEB-додатки для здійснення платежів та он-лайн покупок в Інтернеті, виконання завдань в Інтернеті, підтримуючи зв’язок із друзями та багато іншого. Саме тому протягом останніх декількох років спостерігається великий поштовх для розробки WEB-додатків.

Більшість WEB-додатків базуються на архітектурі клієнт-сервер, куди клієнт надає інформацію, а сервер отримує та зберігає інформацію. Прикладом цього є Інтернет-пошта, такі компанії, як Gmail та Microsoft Outlook, пропонують WEB-клієнт електронної пошти. Інший приклад популярних WEB-додатків останнього покоління - G Suite (раніше Google Apps), Microsoft Office 365. В свою чергу мобільні додатки, які підключаються до Інтернету (наприклад програма Facebook, додаток Dropbox або додаток для онлайн-банкінгу), також є прикладами того, як WEB-додатки були розроблені для все більш популярного використання мобільної мережі.

1.2 Актуальність забезпечення безпеки WEB-додатків

1.2.1 Безпека WEB-додатків

Безпека WEB-додатків - це стан WEB-сайтів, WEB-програм в яких забезпечується збереження інформації. Безпека WEB-додатків спирається на принципи захисту коду програм, застосовуючи їх адаптованість до Інтернету та WEB-систем, з метою захисту від зловмисних загроз або атак.

Атаки на WEB-додатки варіюються від цілеспрямованих маніпуляцій з базами даних, які можуть відкрити конфіденційні дані, такі як номери облікових записів та паролі, до масштабних збоїв у мережі.

Хоча WEB-атаки не є новиною, вони є дорогою і зростаючою проблемою. За оцінками групи Herjavec, щорічні витрати на кіберзлочинність становитимуть 6 трлн дол. США до 2021 р. Спільне дослідження Інституту Понемона та IBM визначає середньосвітову вартість окремого порушення даних у 3,92 млн дол. Для США ця цифра ще вища - 8,19 мільйона доларів [12].

Шанси у будь-якого бізнесу, який зазнав порушення даних протягом двох років, зросли майже до 30%. Згідно зі звітом про розслідування порушень даних Verizon за 2020 рік , майже 25% усіх порушень безпеки безпосередньо стосуються вразливостей WEB-додатків, а 62% випадків злому спрямовані на WEB-програми [12].

Таким чином безпека WEB-додатків набуває все більшого значення. На цей час існує ряд ресурсів та інструментів, які допомагають захистити WEB-програми. Ключовим є активність щодо безпеки на кожному етапі життєвого циклу створення програмного забезпечення.

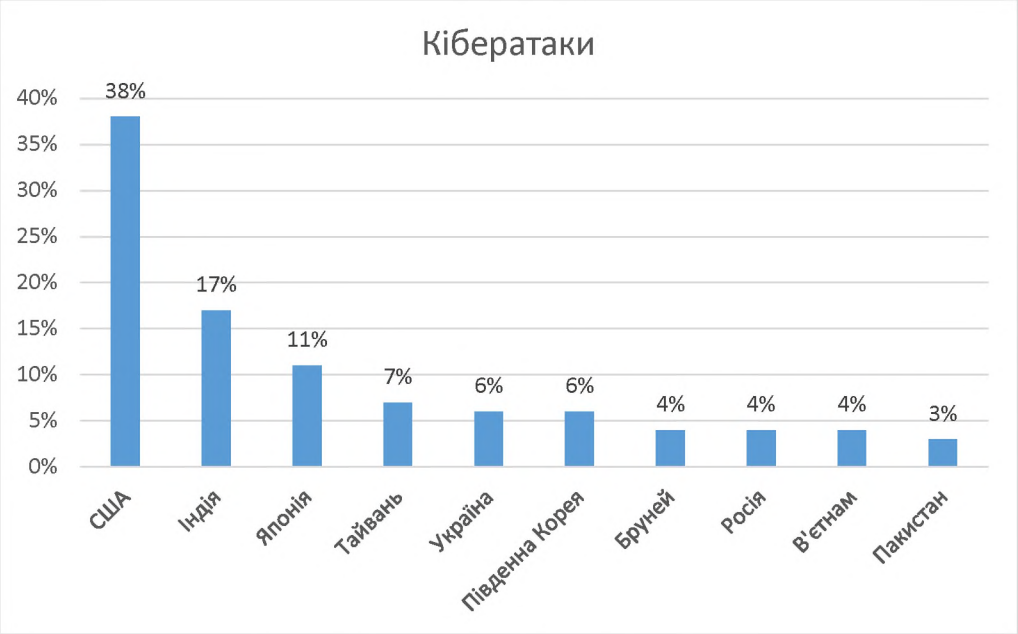


Рисунок 1.1 – Статистика кібератак

1.2.2 Способи забезпечення безпеки WEB-додатків на різних етапах створення

Перш ніж WEB-додаток почне свою роботу за призначенням він проходить великий шлях по закладанню безпеки, а саме:

– етап технічного завдання;
– етап розробки структури;
– етап адміністрування;
– етап експлуатації.

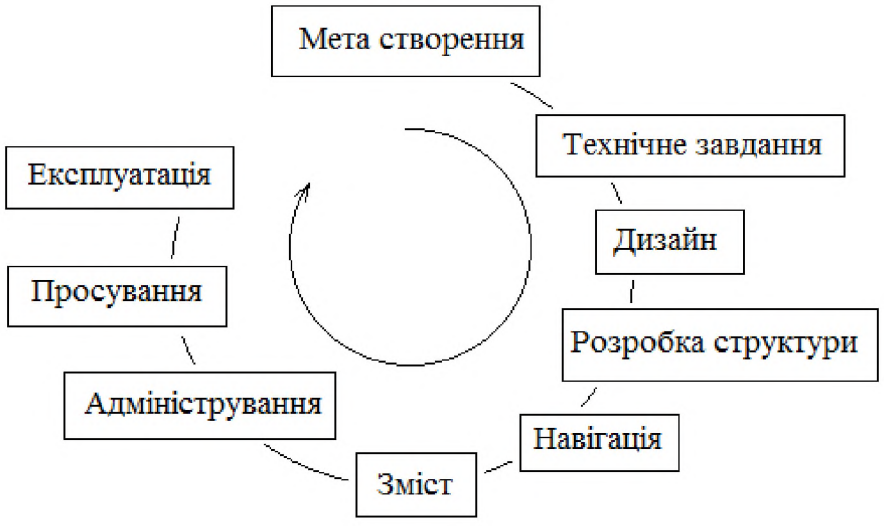


Рисунок 1.2 - Етапи створення WEB-додатку

На етапі проектування закладається структура WEB-додатку, що безумовно впливає на безпеку. Якщо під час реалізації наступних етапів зробити зміну структури це може призвести до ануляції всього проекту.

На етапі розробки реалізується структура проекту, а саме розробка дизайну, кодування та тестування. Саме тут закладаються механізми безпеки.

На етапі адміністрування підтримується працездатність WEB-додатку, відбувається розмежування доступу, реалізації механізмів безпеки та інше.

Кожен з цих етапів дуже важливий, так як допущену помилку не можливо виправити у наступних етапах. Саме тому етап розробки є одним з основних. Коли на пізньому етапі тестування виявляються недоліки безпеки, зміни коду стають важкими.

1.2.3 Основні ризики безпеки WEB-додатків

OWASP розшифровується як Open Web Application Security Project, Інтернет-спільнота, яка робить статті, методології, документацію, інструменти та технології в галузі безпеки WEB-додатків.

OWASP є лідером у галузі безпеки WEB-додатків і підтримує 10 найкращих списків ризиків безпеки WEB-додатків, які вважаються галузевим стандартом захисту WEB-додатків.

Ризики безпеки за останніми даними OWASP для WEB-додатків [9]:

– ін’єкція: вади ін'єкції, такі як SQL, NoSQL, OS та LDAP, виникають, коли ненадійні дані надсилаються інтерпретатору як частина команди або запиту. Ворожі дані зловмисника можуть змусити перекладача виконати ненавмисні команди або отримати доступ до даних без належного дозволу;

– порушення автентифікації: функції додатків, пов’язані з автентифікацією та управлінням сеансами, часто реалізуються неправильно, дозволяючи зловмисникам компрометувати паролі, ключі або маркери сеансів або використовувати інші недоліки реалізації, щоб тимчасово або назавжди прийняти ідентифікаційні дані інших користувачів;

– розкриття конфіденційних даних: багато WEB-додатків та API не

захищають належним чином конфіденційні дані, такі як фінансові, охоронні та ідентифікаційні дані. Зловмисники можуть викрасти або модифікувати такі слабо захищені дані для здійснення шахрайства з кредитними картками, викрадення особистих даних або інших злочинів. Конфіденційні дані можуть бути скомпрометовані без додаткового захисту, наприклад, шифрування в спокої або під час передачі, і вимагає особливих заходів обережності при обміні ними з браузером;

– зовнішні об'єкти XML (XXE): багато старих або погано налаштованих процесорів XML оцінюють посилання на зовнішні сутності в документах XML. Зовнішні сутності можна використовувати для розкриття внутрішніх файлів за допомогою обробника URI файлу, внутрішніх спільних файлів, внутрішнього сканування портів, віддаленого виконання коду та атак відмови в обслуговуванні;

– порушений контроль доступу: обмеження щодо того, що дозволяють робити автентифіковані користувачі, часто не застосовуються належним чином. Зловмисники можуть використовувати ці недоліки для доступу до несанкціонованих функціональних можливостей та / або даних, таких як доступ до облікових записів інших користувачів, перегляд конфіденційних файлів, зміна даних інших користувачів, зміна прав доступу тощо;

– неправильна конфігурація: неправильна конфігурація безпеки є найбільш часто зустрічається проблемою. Зазвичай це результат небезпечних конфігурацій за замовчуванням, неповних або спеціальних конфігурацій, відкритого хмарного сховища, неправильно налаштованих заголовків HTTP та детальних повідомлень про помилки, що містять конфіденційну інформацію. Не тільки всі операційні системи, фреймворки, бібліотеки та програми повинні бути надійно налаштовані, але вони повинні бути вчасно виправлені або оновлені;

– Cross-Site Scripting (XSS): недоліки XSS трапляються, коли програма включає ненадійні дані на новій WEB-сторінці без належної перевірки чи екранування, або оновлює наявну WEB-сторінку даними, наданими користувачем, за допомогою API браузера, який може створювати HTML або JavaScript. XSS дозволяє зловмисникам виконувати сценарії в браузері жертви, які можуть

викрадати сеанси користувачів, псувати WEB-сайти або перенаправляти користувача на шкідливі сайти;

– небезпечна десеріалізація: небезпечна десеріалізація часто призводить до віддаленого виконання коду. Навіть якщо недоліки десеріалізації не призводять до віддаленого виконання коду, їх можна використовувати для виконання атак, включаючи атаки відтворення, атаки інєкції та атаки з підвищенням привілеїв;

– компоненти, що використовують з відомими вразливостями: компоненти, такі як бібліотеки, фреймворки та інші програмні модулі, працюють з тими самими привілеями, що і програма. Якщо експлуатується вразливий компонент, така атака може сприяти серйозній втраті даних або захопленню сервера. Програми та API, що використовують компоненти з відомими вразливими місцями, можуть підірвати захист додатків та забезпечити різні атаки та впливи;

– недостатня реєстрація та моніторинг: недостатня реєстрація та моніторинг у поєднанні з відсутністю або неефективною інтеграцією з реагуванням на інциденти дозволяє зловмисникам продовжувати атакувати системи, підтримувати стійкість, переходити до більшої кількості систем і підробляти, витягувати або знищувати дані. Більшість досліджень порушень показують, що час на виявлення порушення становить понад 200 днів, як правило, виявляються зовнішніми сторонами, а не внутрішніми процесами чи моніторингом.

1.3 Технології розробки WEB-додатків

Раніше WEB-сайти створювалися за допомогою мов HTML та CSS при цьому сайти виходили статичними. В сучасному світі завдяки використанню фреймворків сайти стали динамічними.

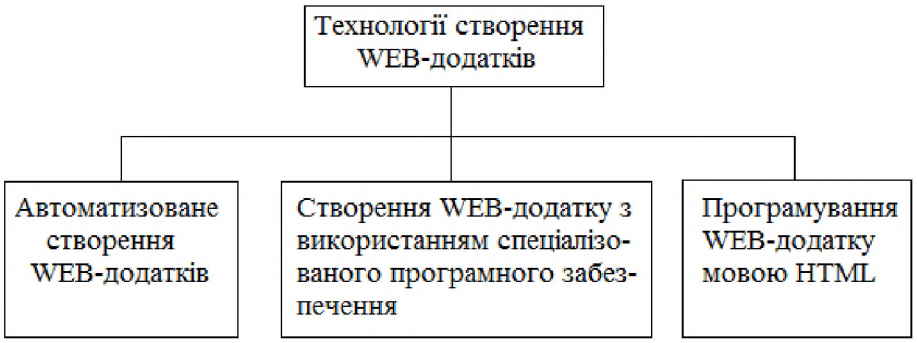


Рисунок 1.3 – Класифікація технологій створення WEB-додатків

1.3.1 Поняття фреймворку та його призначення

Фреймворки - це програмні продукти, які спрощують створення і підтримку технічно складних або навантажених проектів. Фреймворк, як правило, містить тільки базові програмні модулі, а все специфічні для проекту компоненти реалізуються розробником на їх основі. Тим самим досягається не тільки висока швидкість розробки, а й велика продуктивність і надійність рішень.

WEB-фреймворк - це платформа для створення сайтів і WEB-додатків, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. За рахунок широких можливостей в реалізації бізнес-логіки і високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків і WEB-сервісів.

WEB-платформа підтримує створення, розробку і публікацію WEB-додатків і WEB-сайтів. Сюди можуть входити WEB-служби, API та інші ресурси. WEB - фреймворки - це програмні фреймворки, які пропонують стандартний і доступний спосіб створення і розробки WEB-додатків.

WEB-додатки використовуються кожен день більшістю підключених до Інтернету людей, вони все ще знаходяться в зародковому стані, коли справа доходить до розробки тактики і інструментів. Ось чому так важливо мати WEB-фреймворк, який виступав би в якості системи підтримки для розробки і

розгортання таких додатків. Платформа WEB-додатків є одним з таких інструментів.

1.3.2 Загальні функції WEB-фреймворків

WEB-фреймворки надають функціональні можливості в своєму коді або через розширення для виконання повсякденних операцій, необхідних для запуску WEB-додатків. Ці операції включають:

- маршрутизація URL;
- управління і перевірка форми введення;
- HTML, XML, JSON і інші настройки продукту з механізмом створення шаблонів;
- конфігурація підключення до бази даних і обробка даних за допомогою об'єктно-реляційного відображення (ORM);
- WEB-безпека від підробки міжсайтових запитів (CSRF), SQL-ін'єкцій, міжсайтових сценаріїв (XSS) та інших атак;
- репозиторий сеансів і пошук.

Проаналізувавши вакансії спеціалістів по Backend розробці Flask та Django користуються великою популярністю. Саме тому актуально розглянути які послуги вони надають у сфері безпеки.

1.4 Загальні характеристики Flask та Django

Flask та Django завдяки швидкої адаптації до технологічних змін є дуже корисною WEB-розробкою для мови програмування Python. Обидва фреймворки привернули увагу розробників і використовуються для створення WEB-додатків . Flask та Django мають свої переваги та недоліки, тому не завжди легко вибрати між ними [11].

Flask - це мікро-фреймворк, що пропонує основні функції WEB-додатку. Він пропонує розширення для перевірки форми, об'єктно-реляційного відображення,

має відкриті системи аутентифікації та механізм завантаження і декілька інших інструментів.

Важливі особливості Flask:

- інтегрована підтримка модульного тестування;
- відправлення RESTful-запитів;
- використовує шаблонизатор Ninja2;
- підтримка безпечних файлів cookie (клієнтські сеанси);
- велика документація;
- сумісність з двигуном додатків Google;
- API-інтерфейси мають красиву форму і узгодженість;
- легко розгортається у виробничому середовищі.

Django - це середовище WEB-розробки для Python. Цей фреймворк пропонує стандартний метод швидкої та ефективної розробки WEB-сайтів. Це допомагає створювати і підтримувати якісні WEB-додатки і дозволяє спростити процес розробки заощаджуючи час.

Важливі особливості Django:

- пропонує архітектуру Модель - Представлення - Контролер (MVC);
- має визначені бібліотеки для зображень, графіки, наукових розрахунків та інше;
- підтримує декілька баз даних;
- кроссплатформенна операційна система;
- оптимізує сайти на спеціалізованих серверах;
- підтримує інтерфейсні інструменти, такі як jQuery, Pyjamas та інші.

1.4.1 Переваги та недоліки Django

Переваги Django:

- Django легко налаштувати і запустити;
- надає простий у використанні інтерфейс для різних адміністративних дій;

– пропонує багатомовні WEB-сайти, використовуючи вбудовану систему інтернаціоналізації;
– дозволяє проводити тестування додатків;
– дозволяє документувати ваш API за допомогою виведення HTML;
– REST Framework має багату підтримку декількох протоколів аутентифікації;
– використовується для обмеження швидкості запитів API від одного користувача;
– допомагає визначати шаблони для URL-адрес у вибраному додатку;
– пропонує вбудовану систему аутентифікації;
– фреймворк кешування має кілька механізмів кешування;
– фреймворк високого рівня для швидкої WEB-розробки;
– повний набір інструментів;
– дані, змодельовані за допомогою класів Python.

Недолік Django:

– це монолітній майданчик;
– висока залежність від Django ORM. Потрібні великі знання;
– менше проектних рішень і компонентів;
– сумісність з новітніми технологіями;
– більш висока точка входу для простих рішень;
– більший розмір коду;
– занадто великий для милих проектів;
– недостатньо потужні шаблони і ORM;
– шаблони зазнали невдачі без попередження;
– автоматична перезавантаження перезавантажує весь сервер;
– документації не охоплюють реальних сценаріїв;
– дозволяє обробляти тільки один запит за раз;
– маршрутизація вимагає деякого знання регулярних виразів;
– може розгорнути компоненти разом, що може створити плутанину..

1.4.2 Переваги та недоліки Flask

Переваги Flask:

- підвищена сумісність з новітніми технологіями;
- технічні експерименти;
- легше використовувати для простих випадків;
- розмір кодової бази менше;
- висока масштабованість для простих додатків;
- легко побудувати швидкий прототип;
- URL-адресу маршрутизації;
- легко розробляти і підтримувати додатки;
- проста інтеграція з базою даних;
- менше ядро і легко розширюване;
- мінімальна, але потужна платформа;
- безліч ресурсів є в Інтернеті, особливо на GitHub.

Недоліки Flask:

- більш повільна розробка MVP в більшості випадків;
- більш високі витрати на обслуговування більш складних систем;
- асинхронність може бути невеликою проблемою;
- відсутність бази даних і ORM;
- для створення великого проекту потрібні деякі попередні знання фреймворка;
- пропонує обмежену підтримку і меншу співтовариство в порівняння з Django.

1.4.3 Бази даних що використовуються

ORM - технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних» [13].

Django поставляється в комплекті з Django ORM (Object Relational Mapping) що має дуже великий функціонал. Django ORM офіційно підтримує такі бази даних:

- PostgreSQL;
- MariaDB;
- MySQL;
- Oracle;
- SQLite.

Модель - це єдине, остаточне джерело інформації про дані. Вона містить основні поля та поведінку даних, які зберігаються. Як правило, кожна модель відображається в єдиній таблиці бази даних [8].

Основи:

- кожна модель є класом Python, який є підкласами django.db.models.Model;
- кожен атрибут моделі представляє поле бази даних;
- з усім цим, Django надає автоматично згенерований API доступ до бази даних.

Приклад моделі:

```
from django.db import models
class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

Де first_name і last_name є полями моделі. Кожне поле вказується як атрибут класу, і кожен атрибут відображається у стовпці бази даних.

Наведена вище Person модель створить таблицю бази даних, як це:

```
CREATE TABLE myapp_person (
"id" serial NOT NULL PRIMARY KEY,
"first_name" varchar(30) NOT NULL,
"last_name" varchar(30) NOT NULL
);
```

Основна база даних WEB-додатку за замовченням зберігає такі таблиці, що зберігають технологічні данні:

- django_migrations (рисунок А.1, додаток А);
- sqlite_sequence (рисунок А.2, додаток А);
- auth_group_permissions (рисунок А.3, додаток А);
- auth_users_user_permissions (рисунок А.4, додаток А);
- django_admin_log (рисунок А.5, додаток А);
- django_conrent_type (рисунок А.6, додаток А);
- auth_permission (рисунок А.7, додаток А);
- auth_group (рисунок А.8, додаток А);
- auth_user (рисунок А.9, додаток А);
- django_session (рисунок А.11, додаток А).
- auth_user_groups (рисунок А.11, додаток А);

auth_permission - це таблиця, що представляє auth_permission модель.

auth_group_permissions - це таблиця, яка представляє зв'язок ManyToMany між auth_group і auth_permission.

auth_users_user_permissions - це таблиця, яка представляє ManyToManyField між users_user і auth_permission.

django_migrations – зберігає зміни структури бази даних.

auth_user - зберігає данні про користувачів.

auth_group - зберігає данні про те, що який користувач якій групі належить.

django_session - зберігає сеанси.

1.4.4 Порівняння Flask і Django

Якщо необхідно створити невеликий WEB-додаток то краще використовувати Flask, бо він надає можливість використовувати тільки ті розширення які необхідні. Django навпаки є повнофункціональним, тому вимагає меншої кількості рішення, що необхідно прийняти для розробки. Одна з головних задач Django є безпека, саме тому більшість систем захисту використовуються за

замовчуванням. Тому в цій роботі більшу увагу наділено саме Django. Однак якщо необхідно відмовитись від деяких послуг, що надає Django можуть з'явитися обмеження щодо реалізації проекту.

1.5 Аналіз вимог до захисту інформації WEB-додатків

При створенні WEB-додатків однією з задач є встановлення умов щодо їх захисту. В нашій країні у відповідності нормативним документам, вимоги по захисту від НСД це все профілі захищеності. Нормативний документ НД ТЗІ 2.5-004-99 встановлює критерії оцінки захищеності інформації, яка оброблюється в комп'ютерних системах, від несанкціонованого доступу [3].

Критерії є методологічною базою для визначення вимог:

- з захисту інформації в комп'ютерних системах від несанкціонованого доступу;
- створення захищених комп'ютерних систем і засобів захисту від несанкціонованого доступу;
- оцінки захищеності інформації в комп'ютерних системах і їх придатності для обробки критичної інформації (інформації, що вимагає захисту).

Нормативний документ НД ТЗІ 2.5-010-03 встановлює вимоги до технічних та організаційних заходів захисту інформації WEB-сторінки в мережі Інтернет.

Згідно з визначеними в НД ТЗІ 2.5-004-99 специфікаціями встановлюється мінімально необхідний перелік послуг безпеки інформації та рівнів їх реалізації у комплексах засобів захисту інформації WEB-сторінки від несанкціонованого доступу. Мета цього НД ТЗІ – надання нормативно-методологічної бази для розроблення комплексу засобів захисту від несанкціонованого доступу до інформації WEB-сторінки під час створення комплексної системи захисту інформації.

Для того щоб розглянути вимоги до захисту інформації WEB-додатків необхідно розглянути типові умови функціонування WEB-сайту в автоматизованій системі.

Автоматизована система; АС (automated system) — організаційно-технічна система, що реалізує інформаційну технологію і об'єднує ОС, фізичне середовище, персонал і інформацію, яка обробляється [2].

Слід розуміти, що до складу АС, яка забезпечує функціонування WEB-сторінки, входять: обчислювальна система, фізичне середовище, в якому вона знаходиться і функціонує, середовище користувачів, оброблювана інформація, у тому числі й технологія її оброблення. Під час забезпечення захисту інформації мають бути враховані всі характеристики зазначених складових частин, які впливають на реалізацію політики безпеки WEB-сторінки.

У випадку, якщо WEB-сторінка містить посилання на інформаційні ресурси іншої WEB-сторінки, умови функціонування останньої не повинні порушувати встановлену для даної WEB-сторінки політику безпеки.

В свою чергу інформація WEB-сторінки поділяється на дві категорії:

- загальнодоступна інформація;
- технологічна інформація.

До загальнодоступної інформації відноситься публічно оголошувана інформація, користуватися якою можуть будь-які фізичні або юридичні особи (користувачі інформаційних ресурсів), що мають доступ до мережі Інтернет.

До технологічної інформації WEB-сторінки відноситься технологічна інформація КСЗІ та технологічна інформація щодо адміністрування та управління обчислювальною системою АС і засобами обробки інформації – дані про мережеві адреси, імена, персональні ідентифікатори та паролі користувачів, їхні повноваження та права доступу до об'єктів, інформація журналів реєстрації дій користувачів, інша інформація баз даних захисту, встановлені робочі параметри окремих механізмів або засобів захисту, інформація про профілі обладнання та режими його функціонування, робочі параметри функціонального ПЗ тощо.

Технологічна інформація призначена для використання тільки уповноваженими користувачами з числа співробітників СЗІ та персоналу, що забезпечує функціонування АС.

Способи і методи обробки інформації WEB-сторінки (зберігання, супроводження, передачі, введення, актуалізації та використання інформації) визначають технології оброблення інформації.

Технологічні особливості роботи користувачів із загальнодоступною інформацією WEB-сторінки визначаються особливостями системного та функціонального ПЗ, зокрема броузерів, які ними використовуються.

Технологічні особливості роботи користувачів інших категорій визначаються, крім того, архітектурою АС, способами оброблення та передавання інформації між компонентами АС і способами здійснення доступу до неї.

Можливі наступні способи здійснення доступу до технологічної інформації та передавання даних для актуалізації загальнодоступної інформації:

- з робочої станції, розміщеної на тій самій території, що і WEB-сервер (установи-власника WEB-сторінки або оператора) або з терміналу WEB-сервера;
- з робочої станції, яка розміщена на території установи-власника WEB-сторінки, до WEB-сервера, що розміщений на території оператора, з використанням мереж передачі даних.

Вимоги до захисту інформації WEB-додатків [1]:

- КСЗІ повинна забезпечувати реалізацію вимог із захисту цілісності та доступності розміщеної на WEB-сторінці загальнодоступної інформації, а також конфіденційності та цілісності технологічної інформації WEB-сторінки;
- технологія оброблення інформації повинна відповідати вимогам політики безпеки інформації, визначеної для АС, що забезпечує функціонування WEB-сторінки;
- вимоги щодо забезпечення цілісності загальнодоступної інформації WEB-сторінки та конфіденційності й цілісності технологічної інформації вимагають застосування технологій, що забезпечують реалізацію контрольованого і санкціонованого доступу до інформації та заборону неконтрольованої й несанкціонованої її модифікації;
- технологія оброблення інформації повинна бути здатною реалізовувати

можливість виявлення спроб несанкціонованого доступу до інформації WEB-сторінки та процесів, які з цією інформацією пов'язані, а також забезпечити реєстрацію в системному журналі визначених політикою відповідної послуги безпеки подій (як НСД, так і авторизованих звернень);

– для користувачів, які порушили встановлені правила розмежування доступу до WEB-сторінки, засоби КСЗІ на період сеансу роботи повинні забезпечити блокування доступу до WEB-сторінки;

– технологічними процесами повинна бути реалізована можливість створення резервних копій інформації WEB-сторінки та процедури їх відновлення з використанням резервних копій;

– технологія оброблення інформації повинна передбачати можливість аналізу використання користувачами і процесами обчислювальних ресурсів АС і забезпечувати керування ресурсами.

Виходячи з вище зазначеного визначаються наступні мінімально необхідні рівні послуг безпеки WEB-додатків для забезпечення захисту інформації від загроз:

- за умови, коли WEB-сервер і робочі станції розміщуються на території установи-власника WEB-сторінки або на території оператора (технологія Т1), мінімально необхідний функціональний профіль визначається [1]:

КА-2, ЦА-1, ЦО-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1;

- за умови, коли WEB-сервер розміщується у оператора, а робочі станції – на території власника WEB-сторінки, взаємодія яких з WEB-сервером здійснюється з використанням мереж передачі даних (технологія Т2), мінімально необхідний функціональний профіль визначається [1]:

КА-2, КВ-1, ЦА-1, ЦО-1, ЦВ-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1, НВ-1.

Таким чином НД ТЗІ 2.5-010-03 пред'являє вимоги до WEB-ресурсів на основі двох технологій Т1 та Т2. А саме коли WEB-додаток розміщен на хостингу схема наведена на рисунок 1.4 (технологія Т2) та на серверах компанії схема наведена на рисунок 1.5 (технологія Т1). Так як розміщення на серверах компанії дуже затратно, актуально розмістити WEB-додаток на хостингу.

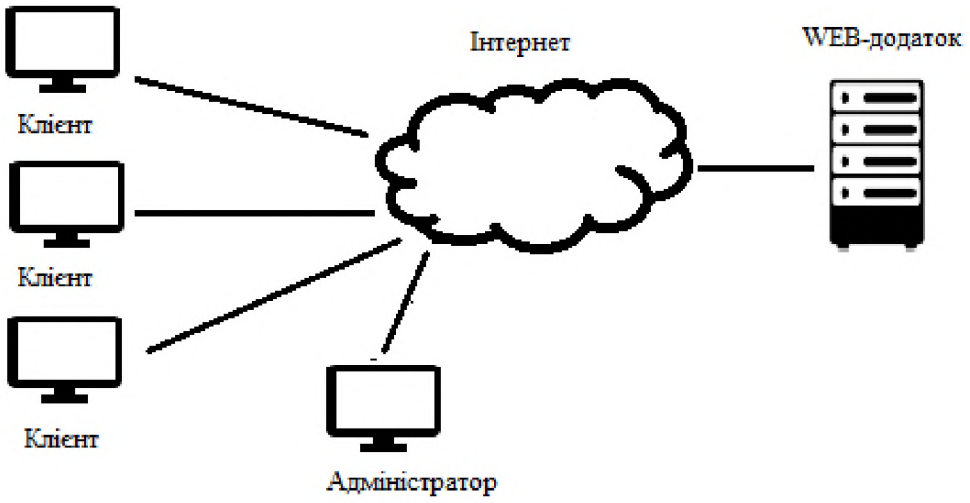


Рисунок 1. 4 - Узагальнена структура обчислювальної системи з використанням хостингу

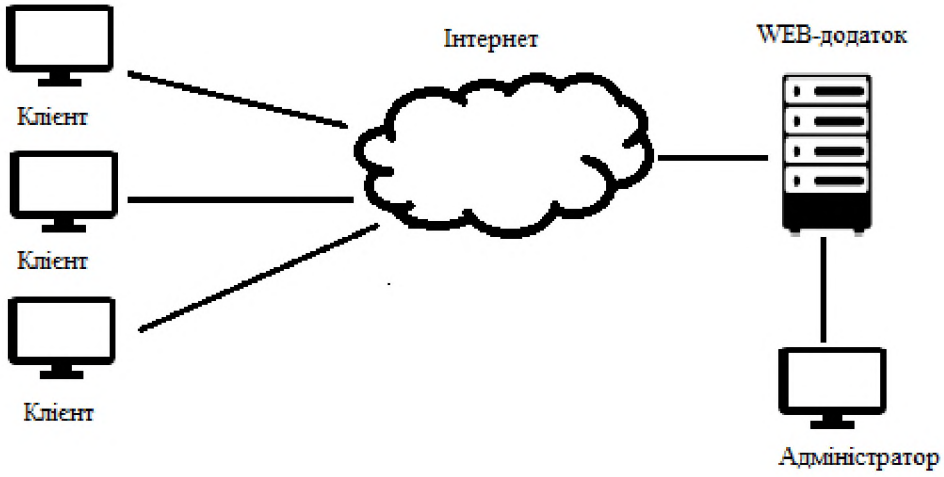


Рисунок 1.5 - Узагальнена структура схема системи при розміщення WEB-додатку на приватному сервері

1.6 Висновок

Задача яка розв’язується є актуальною. Безпека закладається на різних етапах створення WEB-додатку, один з найактуальнішим є етап розробки. При реалізації

технології створення WEB-додатку з використанням спеціалізованого програмного забезпечення широко використовується фреймворки Django та Flask. Саме тому необхідно сформулювати механізми безпеки які створюються за допомогою фреймворків Django та Flask у виді послуг безпеки.

2 СПЕЦІАЛЬНА ЧАСТИНА

2.1 Послуги безпеки інформації WEB-сторінок

WEB-сайти може містити багато конфіденційної інформації, таку як адреси електронної пошти, імена, дати народження та номери кредитних карток. Сьогодні захист конфіденційності інформації дуже важливий, для забезпечення цього використовуються наступні механізми безпеки:

- сесійна аутентифікація;
- управління ролями і дозволами;
- хешування паролів;
- базова HTTP-аутентифікація;
- аутентифікація на базі токенів(атрибут доступу користувача);
- активація облікової запису на базі токенів;
- відновлення / скидання пароля на базі токенів;
- двухфакторна аутентифікація;
- єдиний вхід;
- реєстрація користувачів;
- відстеження входу в систему.

Аутентифікація - процедура, в ході якої користувач повинен пред'явити системі секретну інформацію, відому тільки йому одному.

Сесійна аутентифікація це коли при аутентифікації користувача виділяється Json Web Tokens (JWT), який зберігається у куках користувача. Сервер зберігає у технологічній базі а саме у Django_session рис 1.10 та відстежує активні сесії.

JSON Web Token (JWT) – це відкритий стандарт (RFC 7519) який визначає компактний та автономний спосіб безпечної передачі інформації між сторонами. Цю інформацію можна перевірити та довіряти їй, оскільки вона має цифровий підпис. JWT можна підписати, використовуючи секрет (з алгоритмом HMAC) або пару відкритих/приватних ключів, використовуючи RSA або ECDSA. [15]

Незважаючи на те, що JWT можуть бути зашифровані, щоб також забезпечити таємницю між сторонами, зупинимось на підписаних токенах.

Підписані токени можуть перевірити цілісність вимог, що містяться в ньому, тоді як зашифровані токени приховують ці вимоги від інших сторін. Коли токени підписуються за допомогою пар відкритого / приватного ключів, підпис також засвідчує, що лише сторона, яка володіє закритим ключем, є та, яка його підписала [15].

JWT складаються з трьох частин, розділених крапками (.), а саме:

- заголовок (header);
- корисне навантаження (payload);
- підпис.

JWT зазвичай виглядає наступним чином.

xxxxx.yyyyy.zzzzz

Перша частина - це заголовок, як правило, складається в свою чергу з двох частин: типу токена, яким є JWT, та використовуваного алгоритму підписання, наприклад HMAC SHA256 або RSA.

Друга частина - це корисне навантаження як правила це дані про користувача, а саме registered, public, and private

Третя частина – це підпис що містить закодований заголовок, корисне навантаження, секрет, та алгоритм підписання, що зазначений у заголовку. Наприклад:

HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

Підпис використовується для перевірки того, що повідомлення не було змінено під час передачі. [https://jwt.io/introduction]

Хешування паролів – це зберігання паролів у неявному виді. За замовчуванням, Django використовує алгоритм PBKDF2 з хешем SHA256, механізм захисту паролів рекомендований NIST(Національний інститут стандартів і технологій). PBKDF2 - стандарт формування ключа на основі пароля.

Базова HTTP-аутентифікація це протокол, що виконує аутентифікацію користувача, що вводить логін та пароль.

Аутентифікація на базі токенів - це один з багатьох методів веб-аутентифікації. Аутентифікація токенів вимагає, щоб користувачі отримали згенерований код (або токен), перш ніж їм буде надано доступ в мережу. Аутентифікація токенів зазвичай використовується в поєднанні з аутентифікацією паролів для додаткового рівня безпеки (двухфакторна аутентифікація (2FA)).
Двухфакторна аутентифікація - це один з видів аутентифікації. Сенс двухфакторної аутентифікації полягає в тому, що для того, щоб кудись потрапити, користувач повинен двічі підтвердити той факт, що він - це він, причому, різними способами. Наприклад, ввести логін / пароль (перший фактор), а потім ввести код, присланий на його мобільний телефон (другий фактор).

Активація облікової запису на базі токенів відбувається коли користувач створює акаунт, генерується URL з JWT та відправляється на почну користувача. Перейшовши по якому користувач пройде аутентифікацію та активує акаунт.
Відстеження входу користувача відбувається завдяки сесійній аутентифікації.

Згідно пункту 7.1.2 документа НД ТЗІ 2.5-010-03 є дві технології Т1 і Т2. Технологія Т2 це коли WEB-додаток розміщується у оператора, а робочі станції – на території власника WEB-сторінки. Для цієї технології згідно цього документу рекомендується використовувати такий мінімальний профіль [1]:

КА-2, КВ-1, ЦА-1, ЦО-1, ЦВ-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1, НВ-1.

2.2 Механізми захисту інформації в Django

Для перевірки послуг безпеки на практиці було розроблено WEB-додаток, код якого наведено в додатку Б, та результат в додатку В. Для візуалізації використовувалися стилі від Bootstrap.

2.2.1 Захист від міжсайтових сценаріїв (XSS)

Атаки XSS дозволяють користувачеві вводити сценарії на стороні клієнта в браузери інших користувачів. Зазвичай це досягається шляхом зберігання шкідливих сценаріїв у базі даних, де вони будуть отримані та показані іншим користувачам, або шляхом натискання користувачами на посилання, що спричинить виконання JavaScript зловмисника браузером користувача. Однак атаки XSS можуть походити з будь-якого ненадійного джерела даних, таких як файли cookie або веб-служби, коли дані недостатньо дезінфіковані перед включенням на сторінку. Загальна схем зображена на рисунок 2.1

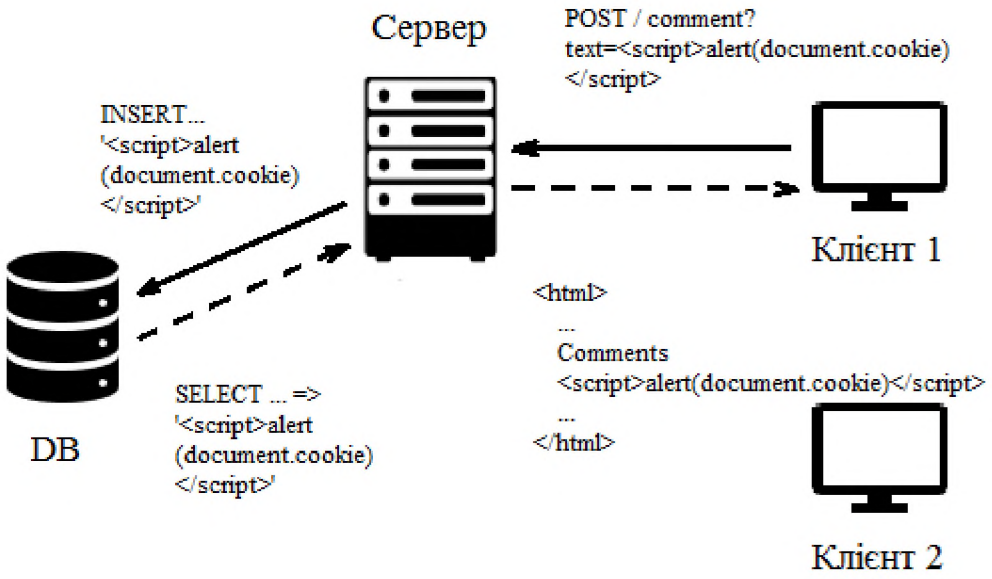


Рисунок 2.1 - Схема атаки XSS

Використання шаблонів Django захищає від більшості атак XSS. Однак важливо розуміти, який захист він забезпечує та його обмеження. Шаблони Django уникають певних символів, які особливо небезпечні для HTML. За замовченням Django кожний шаблон екранує кожний тег змінної.

Наприклад [7]:

- < конвертує у <
- > конвертує у >

– ′ конвертує у "

– " конвертує у &lt.

За необхідності можливо вимкнути XSS у певному місці якщо необхідно відобразити HTML з бази даних.

Це робиться за допомогою фільтру safe як наведено на рисунок 2.2

```
<table width="100%">
    <tr>
        <th width="1%"></th>
        <th width="1%">№</th>
        <th width="25%">Угода</th>
        <th width="25%">Парламентер</th>
        <th width="25%">Дата початку дії договору</th>
        <th width="25%">Дата закінчення дії договору</th>
    </tr>

    {% for el in pages %}
    <tr>
        <td><input type="checkbox" value={{ el.id }} id={{ el.id }}
                   name={{ el.id }}></td>
        <td>{{ forloop.counter }}</td>
        <td><a href="/admin/agreement/{{ el.id }}">{{ el.agreement|safe }}</a></td>
        <td>{{ el.negotiator }}</td>
        <td>{{ el.start_date|date:'d.m.Y' }}</td>
        <td>{{ el.stop_date|date:'d.m.Y' }}</td>
    </tr>
    {% endfor %}

</table>
```

Рисунок 2.2 - Приклад реалізації safe у HTML коді

JavaScript скрипт може попасти у базу даних через звичайну форму. Форма призначена для обміну даними між користувачем і сервером. Наприклад створимо запис у форму та запишемо скрипт: <script>alert(document.cookie)</script> у поля як показано на рисунок 2.3.

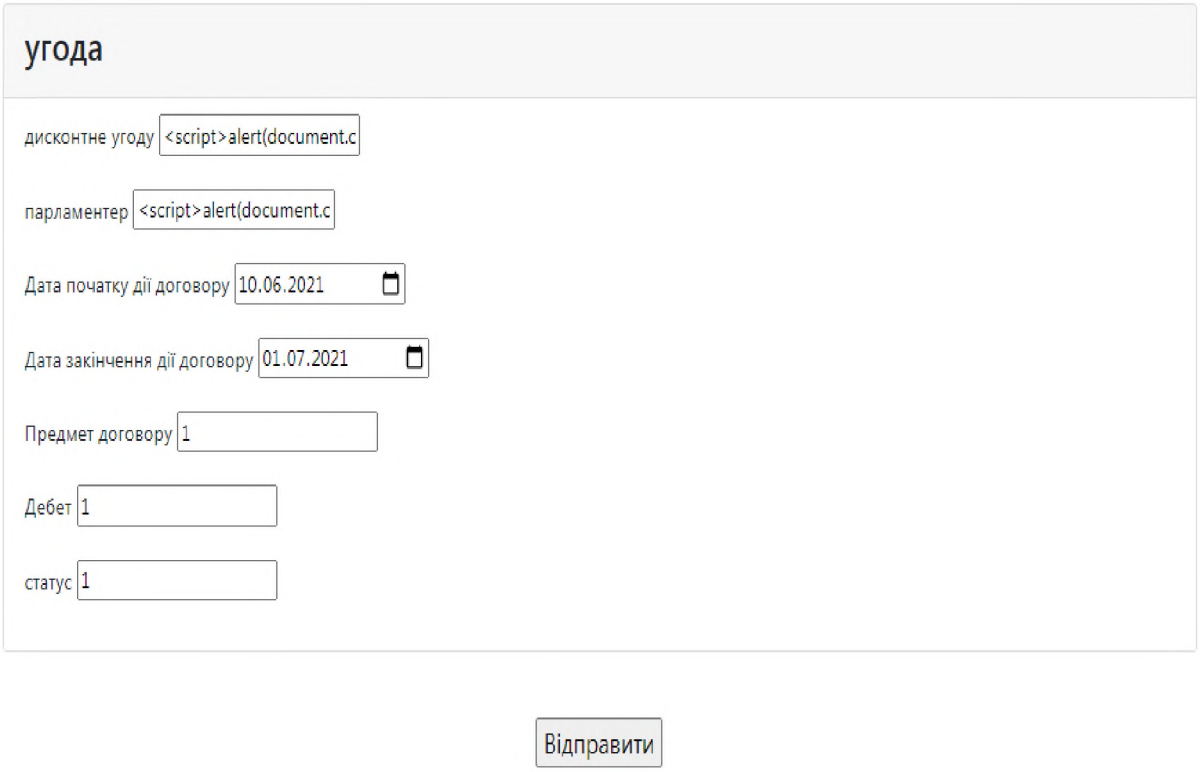


Рисунок 2.3 - Схема полів за допомогою яких JavaScript скрипт заносився у базу даних

Після збереження форми при відображенні таблиці рисунок 2.4, спрацює JavaScript скрипт, результат якого наведено на рисунок 2.5.

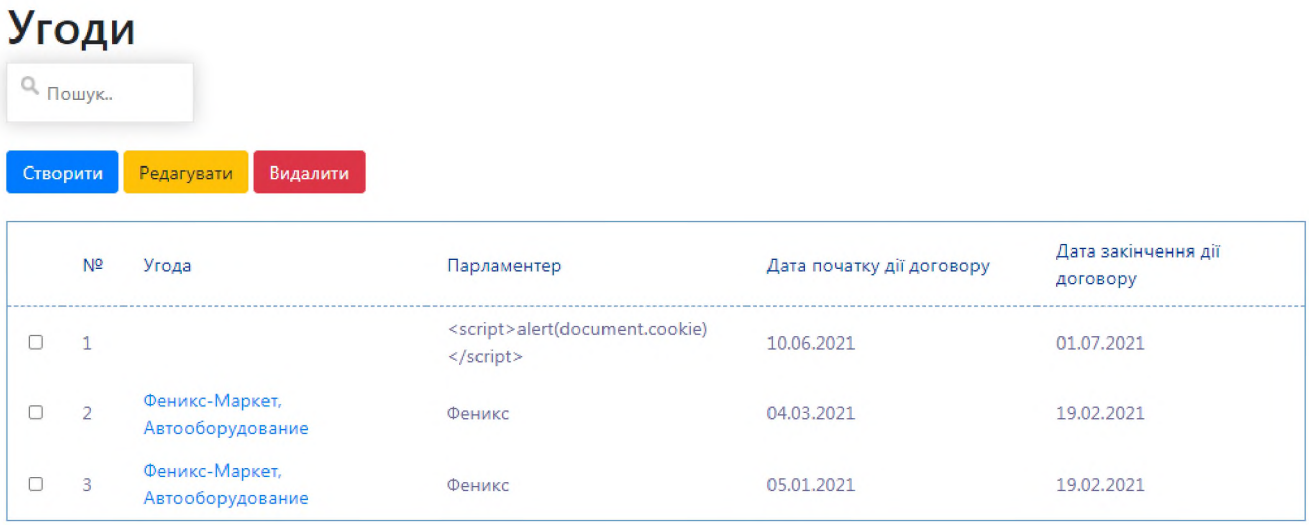


Рисунок 2.4 - Схема таблиці відображення даних

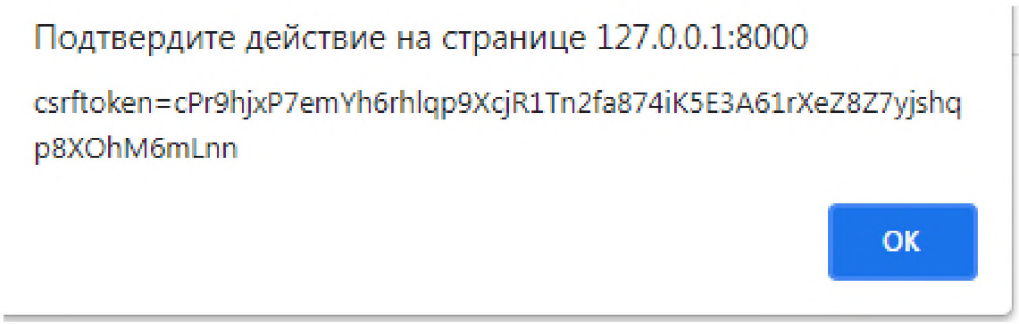


Рисунок 2.5 - Повідомлення, що є результатом роботи JavaScript скрипта

2.2.2 Захист від підробки міжсайтових запитів (CSRF)

Атаки CSRF дозволяють зловмисному користувачу виконувати дії, використовуючи облікові дані іншого користувача, без його відома або згоди [4].

Django має вбудований захист від більшості типів атак CSRF. Захист CSRF працює, перевіряючи секрет у кожному запиті POST. Це гарантує, що зловмисник не може повторно відправити форму POST на ваш веб-сайт. Зловмисний користувач повинен був би знати секрет, який відповідає конкретному користувачеві (за допомогою файлів cookie).

Для того щоб розмістити CSRF у HTML необхідно написати {% crsf_token %} у форму, як показано на рисунок 2.6 також необхідно підключити декоратор csrf_protect до функції що обслуговує цей HTML файл як показано на рисунок 2.7

```
{% extends "TemplatesForAdmin/base.html" %}
{% block content %}
    <div class="container">
        <form action="{% url 'add_agreement' %}" method="post">
            {% csrf_token %}
            <div class="card" style="width: 100%;">
                <div class="card-header">
                    <h3>Угода</h3>
                </div>
                <ul class="list-group list-group-flush">
                    <li class="list-group-item">
                        {{ form.agreement.label }}  {{ form.agreement }}<br><br>
                        {{ form.negotiator.label }}  {{ form.negotiator }}<br><br>
                        {{ form.start_date.label }}  {{ form.start_date }}<br><br>
                        {{ form.stop_date.label }}  {{ form.stop_date }}<br><br>
                        {{ form.subject_of_agreement.label }}  {{ form.subject_of_agreement }}<br><br>
                        {{ form.debit.label }}  {{ form.debit }}<br><br>
                        {{ form.credit_turnover.label }}  {{ form.credit_turnover }}<br><br>
                    </li>

                </ul>
            </div><br>
            <br>
            <blockquote class="blockquote text-center">
                <input type="submit" value="Відправити">
            </blockquote>
            <br>
        </form>
    </div>
{% endblock %}
```

Рисунок 2.6 - Фрагмент коду написаний на HTML для застосування crsf_token

```
165
166     from django.views.decorators.csrf import csrf_protect
167
168
169     @csrf_protect
170     def add_agreement(request):
171         if request.method == 'POST':
172             form = Profile(request.POST)
173             if form.is_valid():
174                 add(form.cleaned_data)
175         else:
176             form = Profile()
177         return render(request, 'TemplatesForAdmin/AddAgreement.html', {'form': form})
178
```

Рисунок 2.7 - Фрагмент коду який забезпечує підключення CSRF токену до функції add_agreement

2.2.3 Захист від ін'єкції SQL

SQL ін'єкція - це тип атаки, коли зловмисний користувач може виконувати довільний код SQL у базі даних [4]. Це може призвести до видалення записів або витоку даних. Схематично зображено на рисунок 2.8

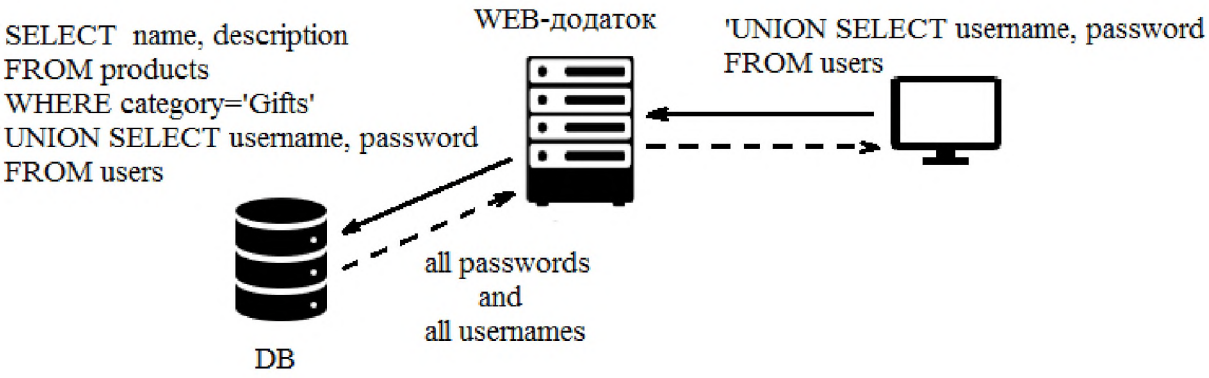


Рисунок 2.8 - Схема реалізації SQL ін'єкції

Django містить певні набори запитів, що захищені від введення SQL, оскільки будуються за допомогою параметризації. SQL-код в Django визначається окремо від параметрів запиту. Оскільки параметри можуть бути надані користувачем або зловмисником Django ORM використовує їх як змінні, а не як запит.

2.2.4 Атака Clickjacking

Атаки Clickjacking засновані на візуальних прийомах, що дозволяють відвідувачам веб-сайту натискати на елементи призначеного для користувача інтерфейсу, які будуть виконувати дії на іншому веб-сайті. Django містить захист від натискання у формі, який у підтримуваному браузері може запобігти отриманню сайту всередині frame [4]. Схематичне зображення зображене на рисунок 2.9

Сучасні браузери дотримуються HTTP - заголовоку X-Frame-Options, який вказує, чи дозволено завантажувати ресурс у frame чи iframe. Якщо відповідь містить заголовок зі значенням, SAMEORIGIN тоді браузер завантажить ресурс у frame, лише якщо запит походить з того самого сайту. Якщо для заголовка встановлено значення, DENY тоді браузер заблокує завантаження ресурсу у кадрі незалежно від того, який сайт зробив запит.

Django пропонує кілька способів включення цього заголовку у відповіді з сайту:

– Проміжне програмне забезпечення, яке встановлює заголовок у всіх відповідях.

– Набір декораторів, який можна використовувати для заміни проміжного програмного забезпечення або для встановлення лише заголовка для певних видів.

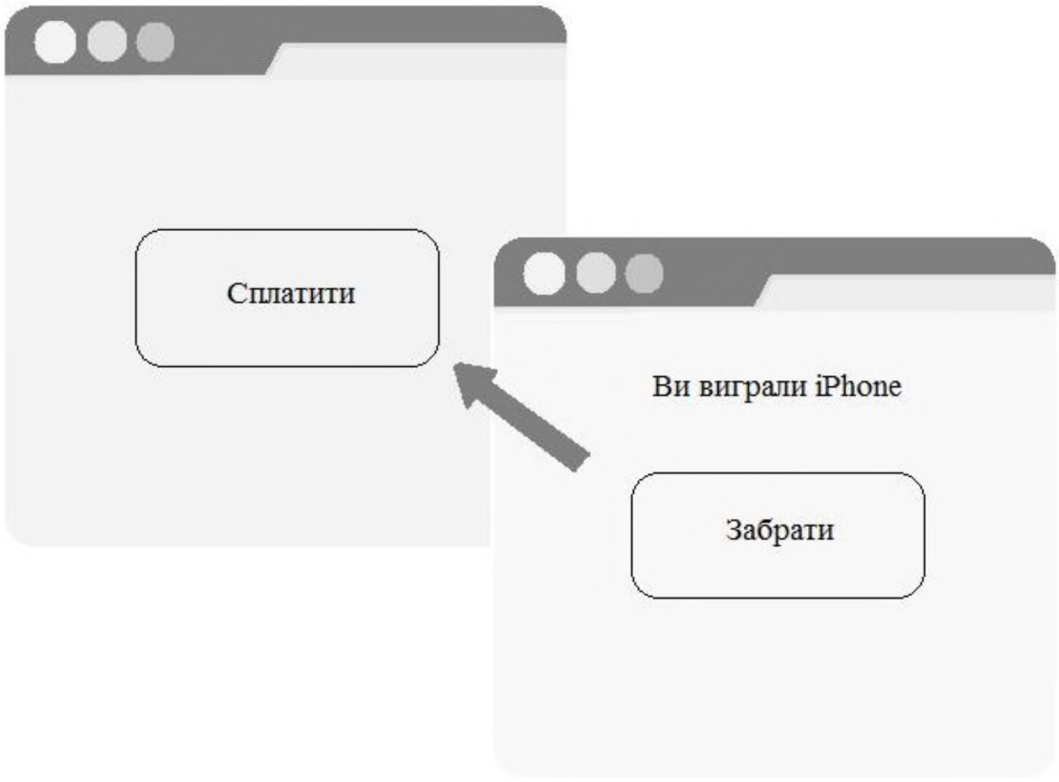


Рисунок 2.9 - Схематичне зображення реалізації атаки Clickjacking

Для того, щоб встановити однакові X-Frame-Options значення для всіх відповідей на вашому сайті, покласти 'django.middleware.clickjacking.XFrameOptionsMiddleware' в список MIDDLEWARE:

```
MIDDLEWARE = [
    django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Це проміжне програмне забезпечення ввімкнено у файлі налаштувань, згенерованому startproject.

За замовчуванням програмне забезпечення встановлює X-Frame-Options заголовок на DENY кожен вихідний HttpResponse. Якщо замість цього потрібно будь-яке інше значення для цього заголовка, встановіть X_FRAME_OPTIONS параметр:

```
X_FRAME_OPTIONS = 'SAMEORIGIN'
```

2.2.5 SSL/HTTPS

HTTPS має ідентичний синтаксис використання схемі HTTP. Однак HTTPS подає сигнал браузеру використовувати додатковий рівень шифрування SSL / TLS для захисту трафіку. SSL / TLS особливо підходить для HTTP, оскільки він може забезпечити певний захист, навіть якщо автентифікована лише одна сторона зв'язку

SSL - це сертифікат для захисту Інтернет-з'єднання та захисту будь-яких конфіденційних даних, які надсилаються між двома системами, не даючи злочинцям читати та змінювати будь-яку передану інформацію, включаючи потенційні особисті дані. Ці дві системи можуть бути сервером і клієнтом (наприклад, веб-сайт і браузер для покупок) або сервер на сервері (наприклад, програма з персональною ідентифікаційною інформацією або з інформацією про заробітну плату).

Коли користувач заходить на веб-сайт, браузер запитує інформацію про сертифікат у сервера, який надсилає копію SSL-сертифіката з відкритим ключем. Далі, браузер перевіряє сертифікат, назва якого має збігатися з ім'ям веб-сайту.

Крім того, перевіряється дата дії сертифіката і наявність кореневого сертифіката, виданого надійним центром сертифікації. Якщо браузер довіряє сертифікату, то він генерує попередній секрет (pre-master secret) сесії на основі відкритого ключа, використовуючи максимально високий рівень шифрування, який підтримують обидві сторони.

Сервер розшифровує попередній секрет за допомогою свого закритого ключа, погоджується продовжити комунікацію і створити загальний секрет (master secret), використовуючи певний вид шифрування. Тепер обидві сторони використовують симетричний ключ, який дійсний тільки для даної сесії. Після її завершення ключ знищується, а під час наступного відвідування процес рукостискання запускається спочатку.

2.2.6 Безпека сесії

Django повністю підтримує анонімні сеанси. Інфраструктура сеансу дозволяє зберігати та видавати довільні дані за принципом «один сайт - один відвідувач». Дані зберігаються на стороні сервера; відправка та отримання файлів cookie прозорі. Файли cookie містять ідентифікатор сеансу, а не самі дані.

За замовчуванням Django зберігає сеанси в базі даних (з шаблоном django.contrib.sessions.models.Session). Хоча це може бути зручно, в деяких конфігураціях швидше зберігати дані сеансу в іншому місці; тому можна налаштувати Django для зберігання даних сеансу в файлової системі або в кеші.

Для того щоб помістити вміст сеансів в базу даних, необхідно додати 'django.contrib.sessions' настройку INSTALLED_APPS.

Після настройки установки запустіть для установки єдиної таблиці бази даних, в якій зберігаються дані сеансу.manage.py migrate

2.3 Аналіз реалізації послуг безпеки фреймворка Django

В даному пункті проаналізовані які послуги безпеки, що задані у НД ТЗІ 2.5-010-03, можна реалізувати за допомогою Django.

Базова адміністративна конфіденційність

{КА-2}. { Базова адміністративна конфіденційність }. {Частково реалізована}. { Вбудована базова адміністративна панель для управлінням користувачами та групами користувачів, а також для відображення таблиць бази даних з загальнодоступною інформацією. Для більшого контролю необхідно її доопрацьовувати або реалізувати свою панель адміністратора з необхідним можливостями }.

Конфіденційність при обміні

{КВ-1}. { Мінімальна конфіденційність при обміні }. {Реалізована}. { Механізми HTTP(за замовчуванням), SSL/HTTPS та використання CSRF токенів дозволяють це реалізувати }.

Базова адміністративна цілісність

{ЦА-2}. { Мінімальна адміністративна цілісність }. { Реалізована }. { За замовчуванням зміну технологічної та інформації з обмеженим доступом надається авторизованим користувачам. Базове розмежування доступом реалізовано за замовчуванням. В ході розробки можливо збільшити рівень адміністративної цілісності за необхідністю }.

Цілісність при обміні

{ЦВ-2}. {Базова цілісність при обміні}. {Реалізована}. { Даний фреймворк дозволяє при обміні інформації використовувати CSRF (Захист від підробки міжсайтових запитів), HTTP, HTTPS для забезпечення цілісність при обміні }.

Відкат

{ЦО-1}. { Обмежений відкат }. {Не реалізовано}. { Django надає можливість відкатувати зміну у структуру таблиць та зв'язки між ними але не надає можливості відкотити зміст таблиць. Для реалізації цього необхідно на етапі проектуванні розробити систему відстеження змін та реалізувати її на етапі розробки. Також це можливо зробити на етапі адміністрування робивши копії бази даних}.

Використання ресурсів

{ДР-1}. { Квоти }. {Не реалізовано}. { Не можливо виділити ресурси під певний процес. Можна реалізувати на етапі адмініструванні }.

Відновлення після збоїв

{ДВ-2}. {Автоматизоване відновлення }. {Не реалізована}. { Реалізовується на етапі адмініструванні та на етапі експлуатації }.

Реєстрація

{ НР-2}. { Аналіз в реальному часі}. { Частково реалізована}. { Фреймворк реєструє результати авторизації та зберігає її у базі даних інших можливостей реєстрацій не надає але можна розробити це додатково}.

Ідентифікація і автентифікація

{НИ-2}. { Одиночна ідентифікація і автентифікація }. {Реалізована}. { В базі даних зберігається ідентифікатори та паролі у захищеному вигляді }.

Ідентифікація і автентифікація при обміні

{НВ-1}. { Автентифікація вузла }. {Реалізована}. { Реалізовується завдяки SSL/HTTPS }.

Достовірний канал

{НК-1}. { Однонаправлений достовірний канал}. {Реалізована}. { Реалізовується за допомогою SSL/HTTPS }.

Розподіл обов'язків

{НО-1}. {Виділення адміністратора}. {Реалізована}. { При необхідності можна створити супер користувача, що матиме повний доступ. Права користувачів надаються в ході розробки }.

Цілісність комплексу засобів захисту

{НЦ-1}. { КЗЗ з контролем цілісності }. {Не реалізована}. { Реалізовується на етапі адмініструванні }.

Самотестування

{НТ-1}. {Самотестування за запитом}. {Реалізована}. { В мові програмування Python є Try Except за допомогою якого можна виявити помилку або збій у реальному часі}.

Try Except в Python

Try - Блок дозволяє перевірити блок коду на наявність помилок.

Except - Блок дозволяє обробити помилку.

Finally - Блок дозволяє виконати код, незалежно від результату, і, крім намагаючись вгадати блоки.

```
try:
    print("Привіт")
except:
    print("Помилка")
except:
    print("Виконано")
```

2.4 Аналіз реалізації послуг безпеки фреймворка Flask

В даному пункті проаналізовані які послуги безпеки, що задані у НД ТЗІ 2.5-010-03, можна реалізувати за допомогою Flask.

Базова адміністративна конфіденційність

{КА-2}. { Однонаправлений достовірний канал}. {Не реалізована}. { Flask не має адміністративної панелі та груп користувачів. Для реалізації цієї послуги необхідно встановити додаткове розширення }.

Конфіденційність при обміні

{КВ-1}. {Мінімальна конфіденційність при обміні}. {Реалізована}. Механізми HTTP(за замовчуванням), SSL/HTTPS дозволяють це реалізувати. Також необхідно встановити flask_wtf для використання CSRF токенів [6]}.

Базова адміністративна цілісність

{ЦА-2}. { Базова адміністративна цілісність }. { Не реалізована }. { Для реалізації необхідно підключити до проекту Flask_Security, а також створити необхідну базу даних, яку він потребує [5]}.

Цілісність при обміні

{ЦВ-2}. {Базова цілісність при обміні}. {Реалізована}. { Даний фреймворк дозволяє при обміні інформації використовувати HTTP, HTTPS для забезпечення цілісність при обміні. Також необхідно встановити flask_wtf для CSRF (Захист від підробки міжсайтових запитів) та виконати відповідні налаштування [6]}.

Відкат

{ЦО-1}. { Обмежений відкат }. {Не реалізовано}. { Для можливості відкату змін у структуру таблиць та зав'язків між ними необхідно встановити Flask-Migrate. Для реалізації можливості відкотити зміст бази даних необхідно на етапі проектуванні розробити систему відстеження змін та реалізувати її на етапі розробки. Також це можливо зробити на етапі адміністрування робивши копії бази даних }.

Використання ресурсів

{ДР-1}. { Квоти }. {Не реалізовано}. { Не можливо виділити ресурси під певний процес }.

Відновлення після збоїв

{ДВ-2}. {Автоматизоване відновлення }. {Не реалізована}. { Реалізовується на етапі адмініструванні та на етапі експлуатації }.

Реєстрація

{НР-2}. { Аналіз в реальному часі}. {Не реалізована}. { Для реалізації необхідно підключити до проекту Flask_Security та виконати відповідні налаштування [5]}.

Ідентифікація і автентифікація

{НИ-2}. {Одиночна ідентифікація і автентифікація}. {Не реалізована}. { Для реалізації необхідно підключити до проекту Flask_Security, за допомогою якої можливо зберігати ідентифікатори та паролі у захищеному вигляді [5]}.

Ідентифікація і автентифікація при обміні

{НВ-1}. {Автентифікація вузла }. {Реалізована}. { Реалізовується завдяки SSL/HTTPS }.

Достовірний канал

{НК-1}. { Однонаправлений достовірний канал}. {Реалізована}. { Реалізовується за допомогою SSL/HTTPS }.

Розподіл обов'язків

{НО-1}. {Виділення адміністратора}. {Не реалізована}. { Для реалізації необхідно підключити до проекту Flask_Security, створити необхідну базу даних, а також створити необхідні ролі [5]}.

Цілісність комплексу засобів захисту

{НЦ-1}. { КЗЗ з контролем цілісності }. {Не реалізована}. { Реалізовується на етапі адмініструванні }.

Самотестування

{НТ-1}. {Самотестування за запитом}. {Реалізована}. { В мові програмування Python є Try Except за допомогою якого можна виявити помилку або збій у реальному часі }.

2.5 Висновок

В спеціальній частині було показано послуги безпеки інформації WEB-сторінок, механізми захисту інформації в Django, послуги безпеки Django та Flask.

Таким чином Django забезпечує повне виконання наступних послуг: мінімальна конфіденційність при обміні, мінімальна адміністративна цілісність, базова цілісність при обміні, одиночна ідентифікація і автентифікація, автентифікація вузла, однонаправлений достовірний канал, виділення адміністратора, самотестування за запитом. Та частково реалізовано: базова адміністративна конфіденційність, аналіз в реальному часі.

Flask забезпечує повне виконання наступних послуг: мінімальна конфіденційність при обміні, базова цілісність при обміні, автентифікація вузла, однонаправлений достовірний канал, самотестування за запитом.

Таким чином фреймворк Django спрощує завдання по забезпеченню WEB-додатку безпекою, відмінно від Flask, який надає теж високий рівень захисту але він більш гибкий та потребує більшого контролю. Встановлено зв'язок між послугами безпеки що реалізують фреймворки та вимогами що необхідні для безпеки WEB-сторінок.

3 ЕКОНОМІЧНА ЧАСТИНА

3.1 Мета техніко-економічного обґрунтування дипломного проекту

Метою виконання економічного розділу є визначення економічної ефективності використання запропонованих у технічній частині засобів та заходів для покращення інформаційної безпеки WEB-додатків розроблених за допомогою фреймворків Django та Flask.

Для визначення економічної доцільності необхідно визначити капітальні витрати на розробку та налагодження, розрахунок річних експлуатаційних витрат на утримання і обслуговування, визначити річний економічний ефект від впровадження засобів і заходів. На основі цих показників можна визначити, чи будуть прибутковими запропоновані рішення.

3.2 Визначення витрат на розробку політики безпеки інформації

3.2.1 Розрахунок капітальних (фіксованих) витрат

Капітальні інвестиції – це кошти, призначені для створення і придбання основних фондів і нематеріальних активів, що підлягають амортизації.

За методикою Gartner Group до фіксованих (капітальних) варто відносити наступні витрати:

- вартість розробки проекту інформаційної безпеки;
- витрати на залучення зовнішніх консультантів;
- вартість первісних закупівель ліцензійного основного й додаткового програмного забезпечення (ПЗ);
- вартість створення основного й додаткового програмного забезпечення;
- витрати на первісні закупівлі апаратного забезпечення;
- витрати на інтеграцію системи інформативної безпеки у вже існуючу корпоративну систему (встановлення обладнання, програмного забезпечення та налагодження системи інформаційної безпеки);

– витрати на навчання технічних фахівців і обслуговуючого персоналу.

Так як для компаній вигідно найняти спеціалізовану компанію для розробки WEB-додатку. Розрахунок витрат на розробку проекту політики інформаційної безпеки включає в себе визначення трудомісткості розробки ПБ і розрахунок витрат на розробку ПБ. Ціна проекту залежить від необхідних годин для його розробки в середньому це 1080 грн/год. Нехай на проект затрачено 10 годин. Тоді його ціна 10800 грн.

3.3.2 Розрахунок експлуатаційних (поточних) витрат

Експлуатаційні витрати – це поточні витрати на експлуатацію та обслуговування об'єкта проектування за визначений період (наприклад, рік), що виражені у грошовій формі.

Для коледжу актуальними будуть наступні витрати:

– заробітна плата обслуговуючого персоналу;

– кваліфікаційні заходи та перевірка знань персоналу стосовно правил;

– організаційне адміністрування.

Оскільки методи захисту, передбачені політикою безпеки, мають більш організаційний характер, поточними витратами можна вважати заробітну платню системного адміністратора і двох співробітників служби безпеки, витрати на електроенергію, що буде витрачено за рік роботи системи охорони та витрати пов’язані з діяльністю користувачів, тож поточні витрати розраховуються за формулою 3.1:

$$С = Сзп + Сел + Сх, \quad (3.1)$$

де С – сума поточних витрат, грн.;

Сел – витрати на електроенергію, що буде витрачено за рік роботи системного адміністратора, грн.;

Сзп – сума заробітної платні персоналу, грн.;

Сх –витрати на аренду хостингу, що буде витрачено за рік, грн.;

Для того щоб підтримувати працездатність WEB-додатку для компаній актуально розміщувати його на хостингу. Це рішення вигідне тим що немає

необхідності в його обслуговуванні та підтримки наймаючи додатковий персонал. Одним з найактуальніших хостингів є heroku, що обійдеться для невеликих проектів у 1350 грн/міс. або 16200 грн/рік.

Для обслуговування WEB-додатку необхідно найняти системного адміністратора.

У свою чергу, витрати на заробітну платню системного адміністратора розраховуються за формулою 3.2:

$$С_{зпад} = (З_{осн} + З_{дод}) + 0,22 \cdot (З_{осн} + З_{дод}) \quad (3.2)$$

$З_{дод}$ – додаткова заробітна плата системного адміністратора за проведення кваліфікаційних заходів

Додаткова заробітна платня складає 500 грн. за проведення одного кваліфікаційного заходу. Такі заходи планується проводити раз на 3 місяці, тож фактично за місяць системний адміністратор отримуватиме 167 грн. додаткової заробітної платні. Основна заробітна платня системного адміністратора на рік становить 162000 грн.

За формулою 3.2 витрати на заробітну платню системного адміністратора становлять:

$$С_{зпад} = (162000 + 2004) + 0,22 \cdot (162000 + 2004)$$

$$С_{зпад} = 200084,00 \text{ грн.}$$

Вартість електроенергії, що споживається апаратурою системи безпеки протягом року визначається за формулою 3.3:

$$С_{ел} = Р \cdot F_р \cdot Ц_е, \quad (3.3)$$

де Р – потужність апаратури, 0,024 кВт;

$F_р$ – річний фонд робочого часу системи безпеки, становить 2304 год;

$Ц_е$ – тариф на електроенергію, 1,68 грн/кВт годин;

За формулою 3.3 слідує:

$С_{ел} = 0,024 \cdot 2304 \cdot 1,68$

$С_{ел} = 93,00$ грн.

Таким чином, за формулою 3.1 розраховуємо експлуатаційні витрати:

$$С = 16200,00 + 93,00 + 200084,00 = 216377,00 \text{ грн.}$$

3.3.3 Оцінка величини збитку у разі реалізації загроз

Метою оцінки є відображення втрат прибутку в разі реалізації загрози інформаційної безпеки.

До загроз інформаційній системі WEB-додатку з можливими економічними втратами можна віднести:

– доступ зловмисниками до вразливостей в програмному коді ПЗ через несвоєчасне оновлення WEB-додатку системним адміністратором, що призведе до простою системи, втрати доступності, конфіденційності інформації і подальшим економічним збиткам;

– несанкціоноване читання, модифікація або видалення інформації через некоректне розмежування прав доступу приводить до втрати конфіденційності, цілісності і доступності інформації, що в свою чергу призводить до подальших грошових втрат;

– помилки користувачів, що призводять до втрати інформації чи надання доступу зловмисникам призведе до простою системи, втрати доступності, конфіденційності інформації і подальшим економічним збиткам;

– злам слабких паролів чи їх крадіжка з метою проникнення у систему, що призведе до втрати доступності, цілісності, конфіденційності інформації і подальшим економічним збиткам;

– збої у функціонуванні системи, що призводять до втрати чи пошкодження інформації, що в ній циркулює призведе до простою у функціонуванні системи і втраті доступності і цілісності.

Таблиця 3.1 – Розрахунок річних обсягів збитків від реалізації загроз

Загроза	Збиток від одиничної реалізації загрози, грн.	Передбачувана кількість реалізацій загрози на рік, шт.	Вірогідність реалізації загрози	Річні збитки від реалізації загрози, грн.
1	2	3	4	5
Доступ зловмисниками до вразливостей через несвоєчасне оновлення ПЗ	41053,00	1	0,8	32842,00
Несанкціоноване читання, модифікація або видалення інформації	80718,00	1	1	80718,00
Помилки користувачів, що призводять до втрати інформації	47923,00	1	1	47923,00
Злам слабких паролів чи їх крадіжка з метою проникнення у систему	60326,00	1	0,8	48260,00

Продовження таблиці 3.1

Загроза	Збиток від одиничної реалізації загрози, грн.	Передбачувана кількість реалізацій загрози на рік, шт.	Вірогідність реалізації загрози	Річні збитки від реалізації загрози, грн.
Збої у функціонуванні системи	37912,00	1	1	37912,00
Загалом				247655,00

3.3.4 Загальний ефект від впровадження системи інформаційної безпеки.

Загальний ефект від впровадження системи інформаційної безпеки визначається за формулою:

$$E = B - C \quad (3.4)$$

де B – загальний збиток від атаки на вузол або сегмент корпоративної мережі, грн.;

C – щорічні витрати на експлуатацію системи інформаційної безпеки, грн.

За формулою 3.4 визначимо:

$$E = 247655,00 – 216377,00$$

$$E = 31278,00 \text{ грн.}$$

3.4 Загальна оцінка економічної ефективності системи захисту інформації

Загальна оцінка економічної ефективності системи захисту інформації здійснюється на основі таких показників, як: – коефіцієнт повернення інвестицій ROSI (Return on Investment for Security); – термін окупності капітальних інвестицій To.

$$ROSI = E / K, \quad (3.5)$$

де Е – загальний ефект від впровадження системи інформаційної безпеки, грн.;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн.

Отже, за формулою 3.5:

$$ROSI = 31278,00/10800,00$$

$$ROSI = 2,89$$

Проект вважається економічно доцільним, якщо розрахунковий коефіцієнт ефективності перевищує річний рівень прибутковості альтернативного варіанта, розраховується за формулою:

$$ROSI > (N_{деп} – N_{інф}) / 100, \quad (3.6)$$

де $N_{деп} = 9$ – річна депозитна ставка, %;

$N_{інф} = 5$ – річний рівень інфляції, %.

Оскільки 2,89 > 0,04, то проект можна вважати економічно доцільним. Термін окупності капітальних інвестицій T_o показує, за скільки років капітальні інвестиції окупаються за рахунок загального ефекту від впровадження системи інформаційної безпеки і розраховується за формулою:

$$T_o = К / Е = 1 / ROSI \quad (3.7)$$

$$T_o = 0,34 \text{ року}$$

3.5 Висновок економічного розділу

В розділі були проведені розрахунки капітальних і річних експлуатаційних витрат. Було визначено збережені збитки від реалізації можливих загроз. З'ясовано, що запропоновані способи реалізації безпеки є економічно вигідними для кінцевого програмного продукту. Термін окупності капітальних інвестицій є досить малим (0,34 року), а коефіцієнт ефективності перевищує річний рівень прибутковості альтернативного варіанта (2,89 > 0,04). Отже, впровадження обраних рішень є доцільним.

ВИСНОВОК

Під час виконання дипломного проекту було виконано дослідження способів реалізації послуг безпеки WEB-додатків при використанні фреймворків Django та Flask. Був проведен аналіз з метою доведення актуальності забезпечення безпеки WEB-додатків в сучасному світі, доказано що саме на етапі розробці WEB-додатків закладається їх безпека.

Розкрито поняття фреймворків та їх призначення. Визначені характеристики та функції фреймворків Django та Flask, їх переваги та недоліки

Було здійснено оцінку послуг безпеки фреймворків Django та Flask згідно «Вимог до захисту інформації WEB-сторінки від несанкціонованого доступу» НД ТЗІ 2.5-010-03, та показано їх реалізацію при створенні WEB-додатків. З'ясовано, що фреймворк Django спрощує завдання забезпечення WEB-додатку безпекою відмінно від Flask.

В економічному розділі були проведені розрахунки капітальних і річних експлуатаційних витрат. Визначено збереження збитків від реалізації можливих загроз. Також доказано, що запропоновані способи реалізації безпеки є економічно вигідними для кінцевого програмного продукту.

Таким чином на прикладі фреймворків Django та Flask доказано актуальність безпеки WEB-додатків для зниження ризику зловмисних атак в середовищі їх використання, тим самим створюючи захист підприємств всіх форм та бізнесу від кібератак.

ПЕРЕЛІК ПОСИЛАНЬ

1. НД ТЗІ 2.5-010-03 "Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу" [Електронний ресурс]. – 2003. – Режим доступу до ресурсу: http://www.dut.edu.ua/uploads/l_1050_27264349.pdf

2. НД ТЗІ 1.1-003-99 "Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу" [Електронний ресурс]. – 2003. – Режим доступу до ресурсу: http://www.dut.edu.ua/uploads/l_1050_27264349.pdf

3. НД ТЗІ 2.5-004-99 "Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу" [Електронний ресурс]. – 1999. – Режим доступу до ресурсу: https://tzi.com.ua/downloads/2.5-004-99.pdf

4. Безпека в Django [Електронний ресурс]. – 2021.– Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/security/

5. Безпека в Flask [Електронний ресурс]. – 2021.– Режим доступу до ресурсу: https://flask-security-too.readthedocs.io/en/stable/

6. Документація Flask [Електронний ресурс]. – 2021.– Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/tutorial/index.html

7. Документація Django [Електронний ресурс]. – 2021.– Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/

8. Django Models [Електронний ресурс]. – 2021.– Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/db/models/

9. OWASP [Електронний ресурс]. – 2021.– Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/

10. WEB-додаток [Електронний ресурс]. – Режим доступу до ресурсу: https://ukr.4meahc.com/what-exactly-is-web-application-50384

11. Порівняння Django та Flask [Електронний ресурс].– Режим доступу до ресурсу: https://www.guru99.com/flask-vs-django.html

12. Статистика [Електронний ресурс] – Режим доступу до ресурсу: https://www.whitesourcesoftware.com/resources/blog/web-application-security

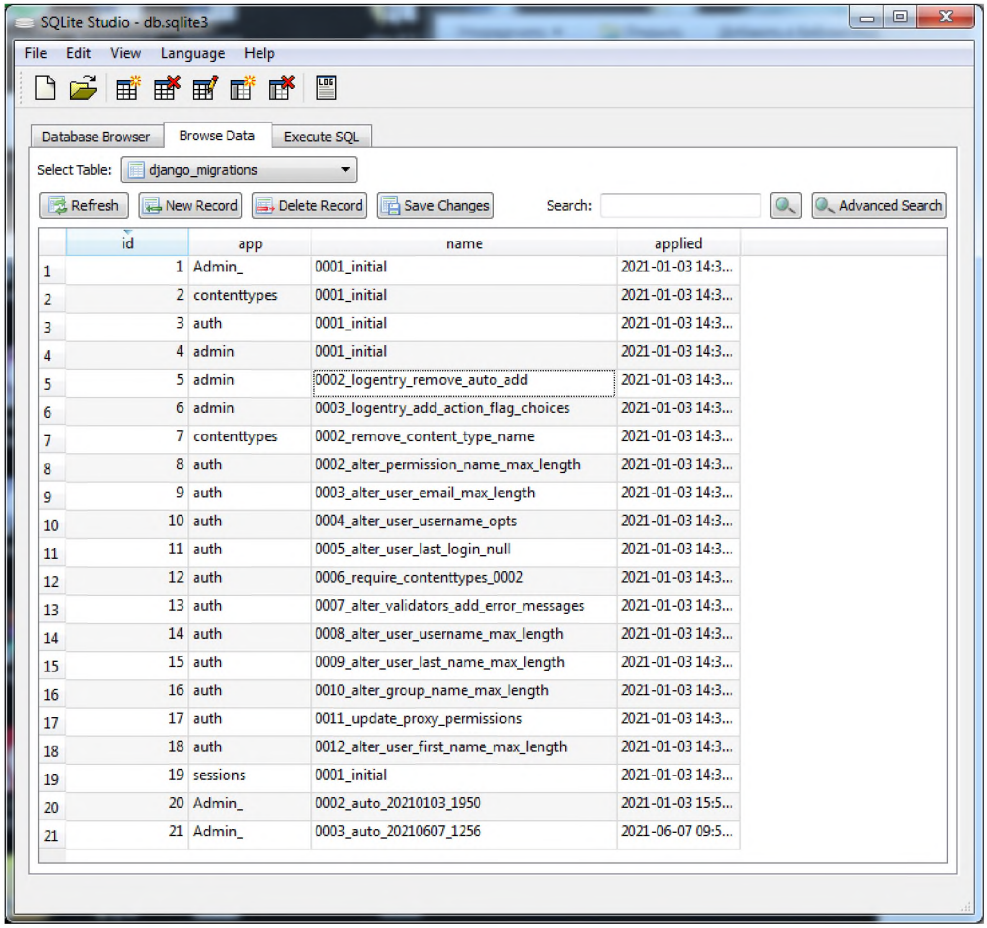
13. ORM [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B5_%D0%B2%D1%96%D0%B4%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F

14. Сесія в Flask [Електронний ресурс] – Режим доступу до ресурсу: https://pythonhosted.org/Flask-Security/quickstart.html#id2

15. JSON WEB Token [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/introduction

16. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів спеціальності [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://www.dut.edu.ua/uploads/l_1050_27264349.pdf

ДОДАТОК А. Таблиці що використовує Django



	id	app	name	applied
1	1	Admin_	0001_initial	2021-01-03 14:3...
2	2	contenttypes	0001_initial	2021-01-03 14:3...
3	3	auth	0001_initial	2021-01-03 14:3...
4	4	admin	0001_initial	2021-01-03 14:3...
5	5	admin	0002_logentry_remove_auto_add	2021-01-03 14:3...
6	6	admin	0003_logentry_add_action_flag_choices	2021-01-03 14:3...
7	7	contenttypes	0002_remove_content_type_name	2021-01-03 14:3...
8	8	auth	0002_alter_permission_name_max_length	2021-01-03 14:3...
9	9	auth	0003_alter_user_email_max_length	2021-01-03 14:3...
10	10	auth	0004_alter_user_username_opts	2021-01-03 14:3...
11	11	auth	0005_alter_user_last_login_null	2021-01-03 14:3...
12	12	auth	0006_require_contenttypes_0002	2021-01-03 14:3...
13	13	auth	0007_alter_validators_add_error_messages	2021-01-03 14:3...
14	14	auth	0008_alter_user_username_max_length	2021-01-03 14:3...
15	15	auth	0009_alter_user_last_name_max_length	2021-01-03 14:3...
16	16	auth	0010_alter_group_name_max_length	2021-01-03 14:3...
17	17	auth	0011_update_proxy_permissions	2021-01-03 14:3...
18	18	auth	0012_alter_user_first_name_max_length	2021-01-03 14:3...
19	19	sessions	0001_initial	2021-01-03 14:3...
20	20	Admin_	0002_auto_20210103_1950	2021-01-03 15:5...
21	21	Admin_	0003_auto_20210607_1256	2021-06-07 09:5...

Рисунок А.1 - Вигляд django_migrations

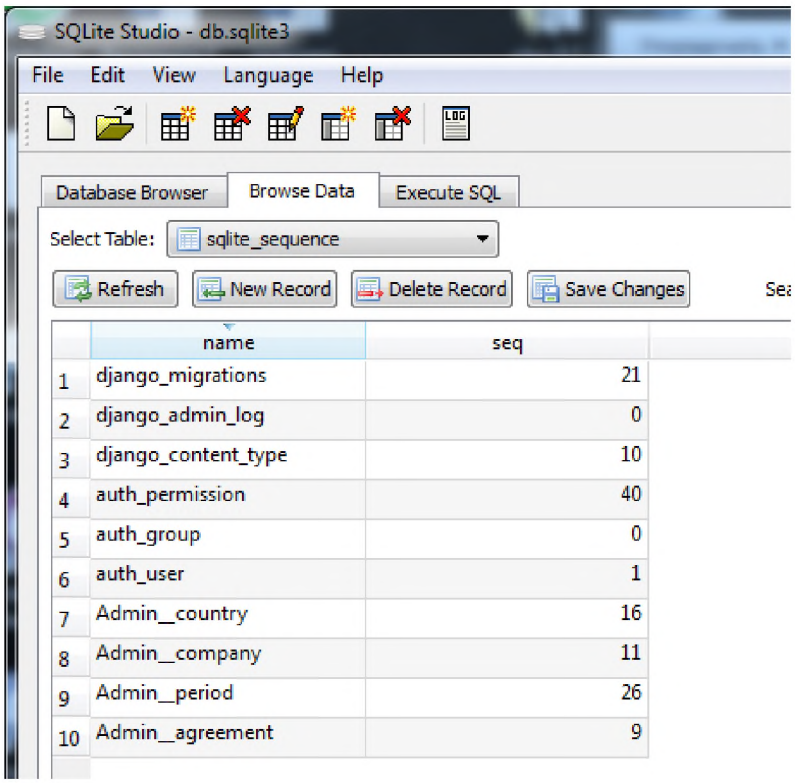


Рисунок А.2 - Вигляд sqlite_sequence

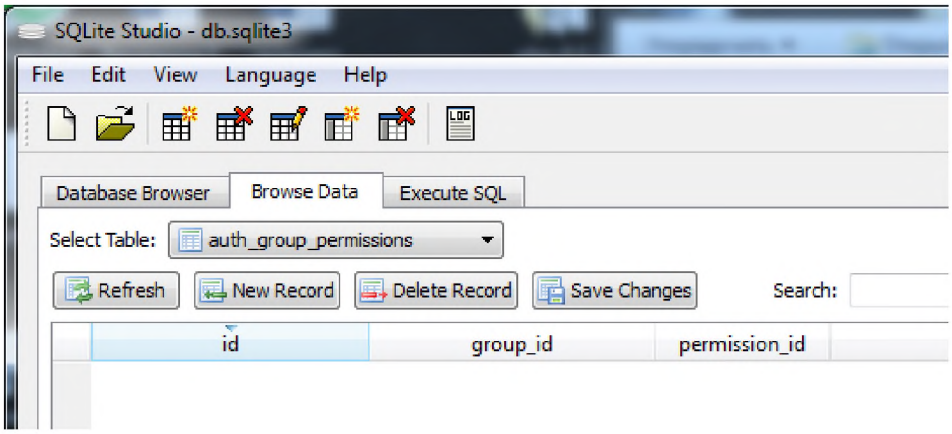


Рисунок А.3 - Вигляд auth_group_permissions

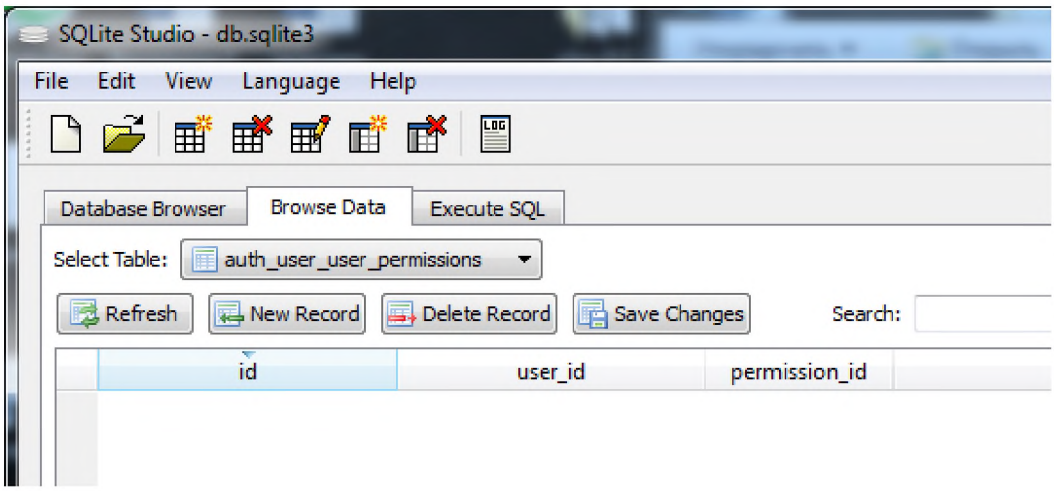


Рисунок А.4 - Вигляд auth_users_user_permissions

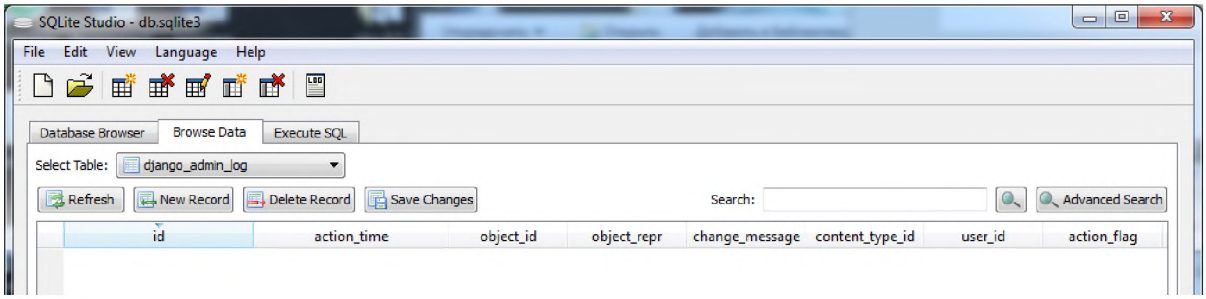
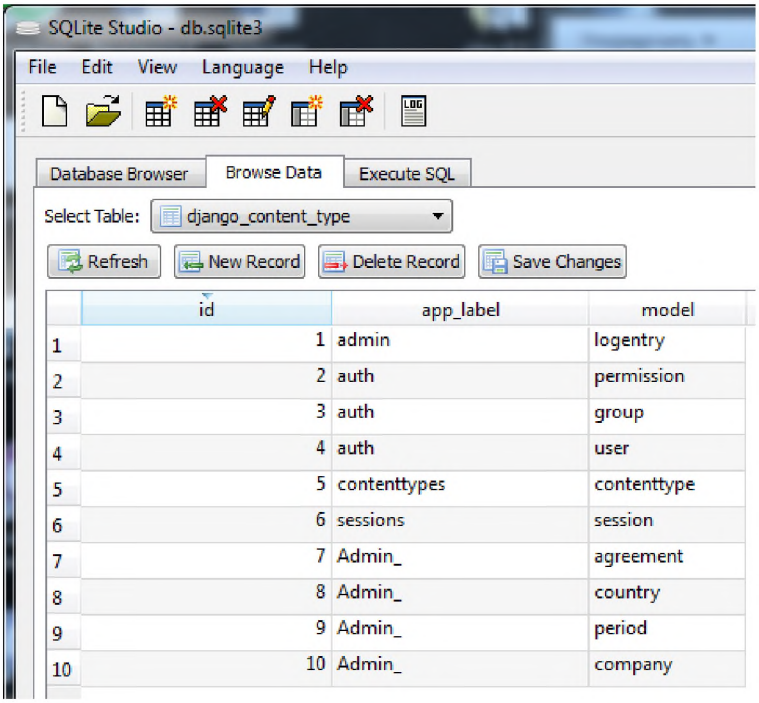


Рисунок А.5 - Вигляд django_admin_log



	id	app_label	model
1	1	admin	logentry
2	2	auth	permission
3	3	auth	group
4	4	auth	user
5	5	contenttypes	contenttype
6	6	sessions	session
7	7	Admin_	agreement
8	8	Admin_	country
9	9	Admin_	period
10	10	Admin_	company

Рисунок А.6 - Вигляд django_conrent_type

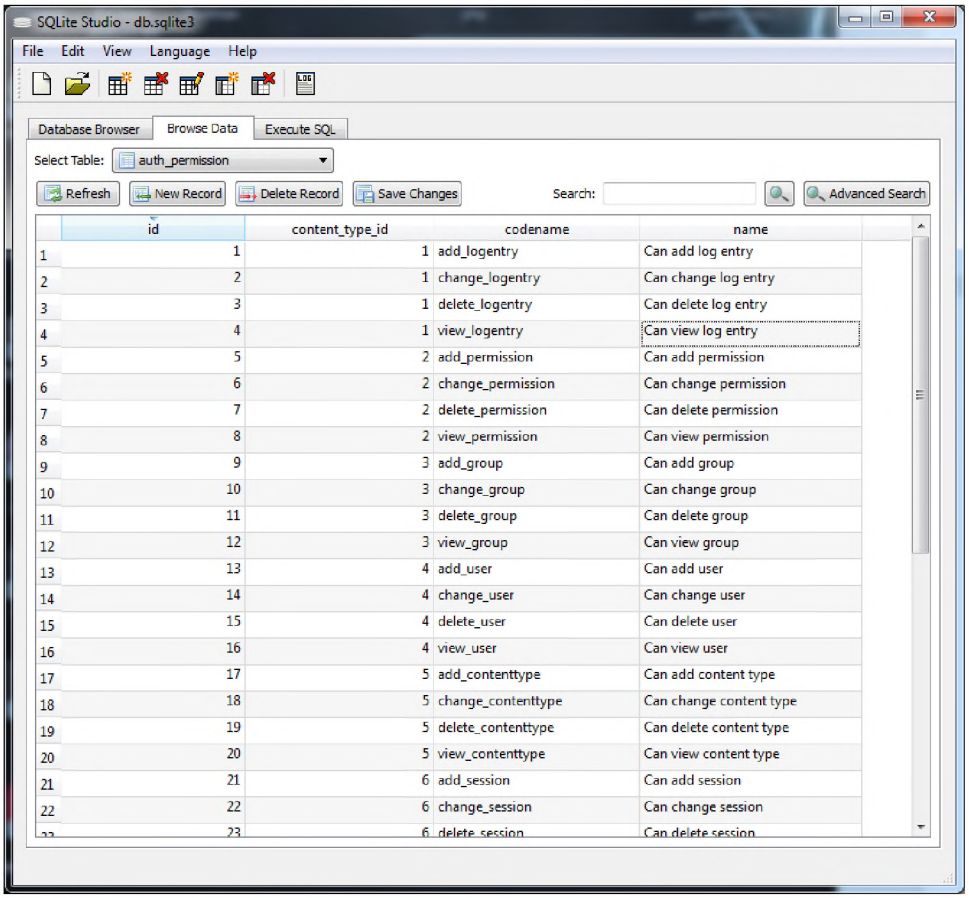


Рисунок А.7 - Вигляд auth_permission

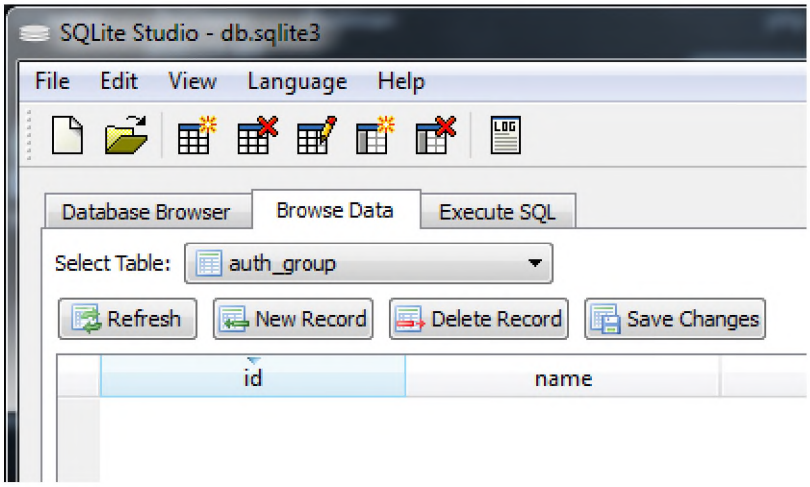


Рисунок А.8 - Вигляд auth_group

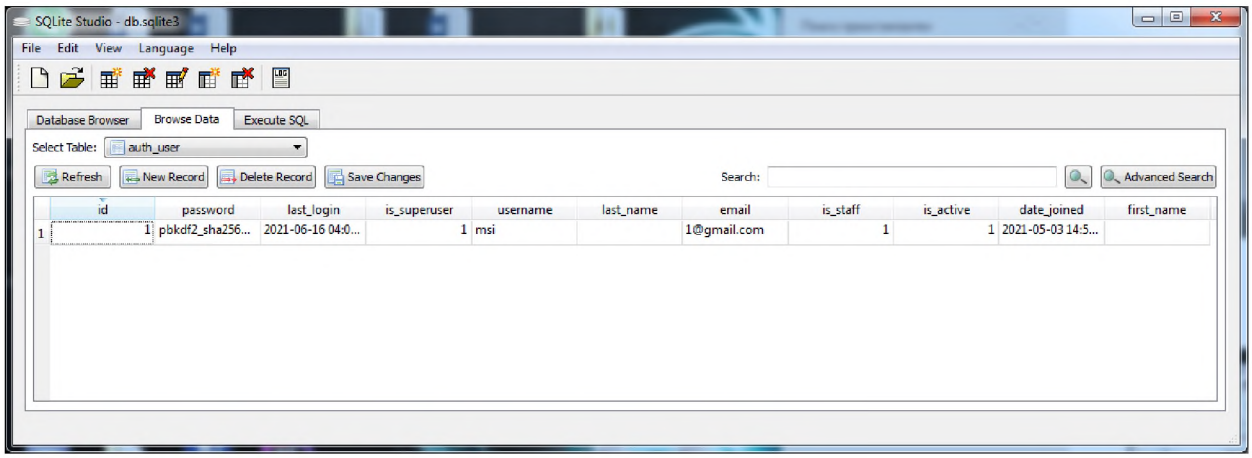


Рисунок А.9 - Вигляд auth_user

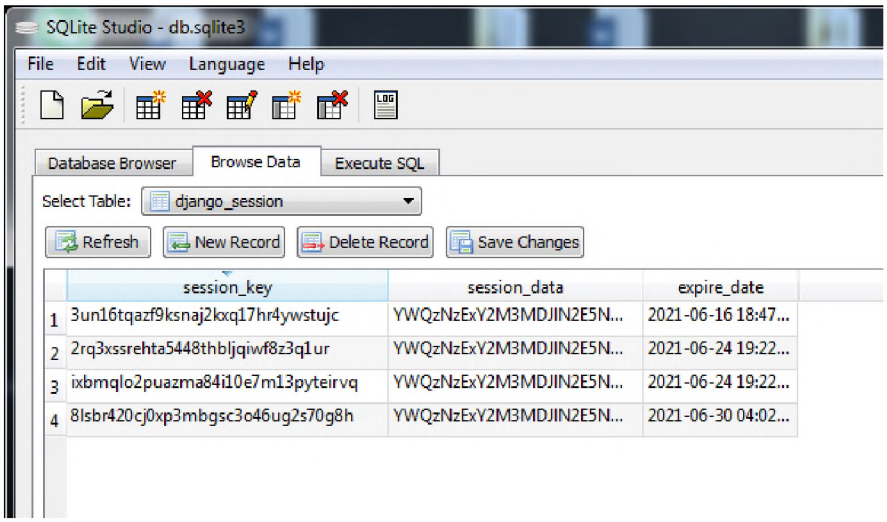


Рисунок А.10 - Вигляд django_session

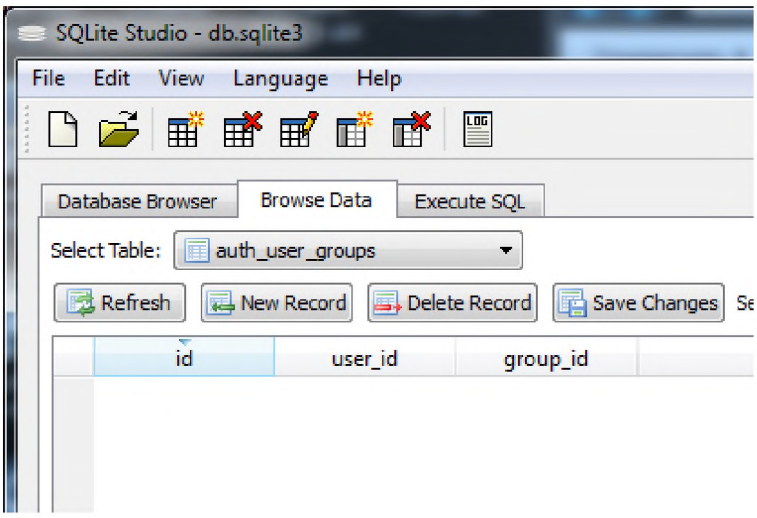


Рисунок А.11 - Вигляд auth_user_groups

ДОДАТОК Б Код розробленого WEB-додатку

urls.py

```
from django.urls import path

from . import views
from . import api

urlpatterns = [
    path('all/', api.api_all_tasks, name='api_all_tasks'),
    path('agreement/', views.agreement, name='agreement'),
    path('', views.home, name='home'),
    path('agreement/add/', views.add_agreement, name='add_agreement'),
    path('agreement/<int:number>/', views.agreement_page, name='agreement_page'),
    path('logout/', views.log_out, name='logout'),
    path('login/', views.loginPage, name="login"),
]
```

models.py

```
from django.db import models

class Country(models.Model):
    __tablename__ = 'country'
    alpha_iso_code = models.CharField('3 альфа изо код', max_length=3)
    country = models.CharField('Страна', max_length=100)

    def __repr__(self):
        return '<Company (id=%s ' \
```

```
            'country=%s ' \
            'alpha_iso_code=%s)>' % (self.id,
                                     self.country,
                                     self.alpha_iso_code,
                                     )


class Company(models.Model):
    __tablename__ = 'company'
    name_company = models.CharField('Компания', max_length=300)
    country = models.ForeignKey(Country, blank=True, on_delete=models.CASCADE)

    def __repr__(self):
        return '<Company (name_company=%s ' \
               'country=%s)>' % (self.name_company,
                                 self.country,
                                 )


class Agreement(models.Model):
    __tablename__ = 'agreement'
    # id_agreement = models.AutoField(primary_key=True)
    agreement = models.TextField('Дисконтна угода')
    negotiator = models.TextField('Парламентер')
    start_date = models.DateTimeField('Дата початку дії договору')
    stop_date = models.DateTimeField('Дата закінчення дії договору')

    subject_of_agreement = models.TextField('Предмет договору')
    debit = models.PositiveIntegerField('Дебет')
    credit_turnover = models.TextField('Статус')
```

```
    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    def __repr__(self):
        return '<Company (agreement=%s ' \
               'negotiator=%s ' \
               'start_date=%s ' \
               'stop_date=%s ' \
               'subject_of_agreement=%s ' \
               'debit=%s ' \
               'company=%s ' \
               'credit_turnover=%s)>' % (self.agreement,
                                         self.negotiator,
                                         self.start_date,
                                         self.stop_date,
                                         self.subject_of_agreement,
                                         self.debit,
                                         self.company,
                                         self.credit_turnover,
                                         )


class Period(models.Model):
    start_date_period = models.DateTimeField('Дата начала конкретной сделки')
    stop_date_period = models.DateTimeField('Дата окончания сделки')
    status = models.TextField('Статус')
    agreement = models.ForeignKey(Agreement, blank=True, null=True,
on_delete=models.CASCADE)

    def __repr__(self):
        return '<Company (id=%s' \
```

```
            'start_date_period=%s ' \
            'stop_date_period=%s ' \
            'status=%s)>' % (self.id,
                             self.start_date_period,
                             self.stop_date_period,
                             self.status,
                             )
```

views.py

```
import datetime

from django.contrib import messages
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.core.paginator import Paginator
from django.http import HttpResponseRedirect
from django.shortcuts import render, redirect
from django.views.decorators.csrf import csrf_protect

from .form import Profile, CreatePeriod
from .models import Agreement, Period
from .models import Company, Country


# superuser
# 1@gmail.com
# msi
# 12345678
```

```
@login_required(login_url='login')
@csrf_protect
def home(request):
    return render(request, 'TemplatesForAdmin/Homepage.html')



def delete_from_db(result):
    for i in result.items():
        if i[0] != 'csrfmiddlewaretoken':
            if i[0] != 'del':
                Agreement.objects.filter(id=i[1]).delete()
                Period.objects.filter(agreement_id=i[1]).all().delete()



def get_our_home_page(request, all_channels):
    paginator = Paginator(all_channels, 15)
    page_number = request.GET.get('page', 1)
    pages = paginator.get_page(page_number)

    is_paginated = pages.has_other_pages()
    if pages.has_previous():
        prev_url = f'?page={pages.previous_page_number()}'
    else:
        prev_url = ''
    if pages.has_next():
        next_url = f'?page={pages.next_page_number()}'
    else:
        next_url = ''
    context = {
```

```
        'pages': pages,
        'is_paginated': is_paginated,
        'prev_url': prev_url,
        'next_url': next_url,
    }
    return context


def set_cookie(response, key, value, days_expire=7):
    if days_expire is None:
        max_age = 365 * 24 * 60 * 60  # one year
    else:
        max_age = days_expire * 24 * 60 * 60
    expires = datetime.datetime.strftime(
        datetime.datetime.utcnow() + datetime.timedelta(seconds=max_age),
        "%a, %d-%b-%Y %H:%M:%S GMT",
    )
    response.set_cookie(
        key,
        value,
        max_age=max_age,
        expires=expires,

    )


@login_required(login_url='login')
def agreement(request):
    print(request.GET)
    try:
```

```
        if request.method == 'GET':
            all_channels = Agreement.objects.filter(agreement__contains=request.GET['q'])
            return render(request, 'TemplatesForAdmin/agreement.html',
context=get_our_home_page(request, all_channels))
    except:
        pass
    if request.method == 'POST':
        print(request.POST)

        for i in request.POST.items():
            if i[0] == 'del':
                delete_from_db(request.POST)
                break
        all_channels = Agreement.objects.all().order_by('-id')
    else:
        all_channels = Agreement.objects.all().order_by('-id')
    return render(request, "TemplatesForAdmin/agreement.html",
context=get_our_home_page(request, all_channels))



def delete_from_period(items):
    for i in items:
        if i[0] != 'csrfmiddlewaretoken':
            if i[0] != 'del':
                Period.objects.filter(id=i[1]).delete()



@login_required(login_url='login')
@csrf_protect
def agreement_page(request, number):
```

```
    if request.POST:
        form = CreatePeriod(request.POST)
        # print(request.POST.items())
        for i in request.POST.items():
            if i[0] == 'del':
                delete_from_period(request.POST.items())
        if form.is_valid():
            Period.objects.create(start_date_period=form.cleaned_data['start_date_period'],
                                  stop_date_period=form.cleaned_data['stop_date_period'],
                                  status=form.cleaned_data['status'],
                                  agreement=Agreement.objects.filter(id=number).first(),
                                  )
    agreement_ = Agreement.objects.get(id__iexact=number)
    all_period = Period.objects.filter(agreement_id=number).all().order_by('-id')
    dict = get_our_home_page(request, all_period)
    dict['agreement'] = agreement_
    dict['FormCreatePeriod'] = CreatePeriod()
    return render(request, 'TemplatesForAdmin/AgreementPage.html', context=dict)


def loginPage(request):
    if request.user.is_authenticated:
        return redirect('home')
    else:
        if request.method == 'POST':
            username = request.POST.get('username')
            password = request.POST.get('password')
            user = authenticate(request, username=username, password=password)
            if user is not None and user.is_staff:
                login(request, user)
```

```
            return redirect('home')
        else:
            messages.info(request, 'Username OR password is incorrect')
    context = {}
    return render(request, 'TemplatesForAdmin/login.html', context)


def log_out(request):
    logout(request)
    return redirect('home')


def add(data):
    Agreement.objects.create(agreement=data['agreement'],
                             negotiator=data['negotiator'],
                             start_date=data['start_date'],
                             stop_date=data['stop_date'],
                             subject_of_agreement=data['subject_of_agreement'],
                             debit=data['debit'],
                             credit_turnover=data['credit_turnover'],
                             company=Company.objects.create(
                                 name_company=data['name_company'],

country=Country.objects.filter(alpha_iso_code=data['country']).first()
                             )
                             )
```

```
from django.views.decorators.csrf import csrf_protect


@csrf_protect
def add_agreement(request):
    if request.method == 'POST':
        form = Profile(request.POST)
        if form.is_valid():
            add(form.cleaned_data)
    else:
        form = Profile()
    return render(request, 'TemplatesForAdmin/AddAgreement.html', {'form': form})
```

form.py

```
from django import forms

from .models import Country, Agreement


class DateInput(forms.DateInput):
    input_type = 'date'


class Profile(forms.Form):
    CHOICES = [(i[1], i[2]) for i in Country.objects.all().values().values_list()]

    agreement = forms.CharField(label='Дисконтное соглашение')
```

```
    negotiator = forms.CharField(label='Переговорщик')
    start_date = forms.DateField(label='Дата начала действия договора',
widget=DateInput)
    stop_date =forms.DateField(label='Дата окончания действия договора',
widget=DateInput)

    subject_of_agreement = forms.CharField(label='Предмет договора')
    debit = forms.IntegerField(label='Дебет')
    credit_turnover =forms.CharField(label='Статус')

    name_company = forms.CharField(label='Компания')

    country = forms.ChoiceField(label='Страна', choices=CHOICES)


class CreatePeriod(forms.Form):
    start_date_period = forms.DateField(label='Дата начала конкретной сделки',
widget=DateInput)
    stop_date_period = forms.DateField(label='Дата окончания сделки',
widget=DateInput)
    status = forms.CharField(label='Статус')
```

api.py

```
from django.db.models import Count
from rest_framework.authentication import SessionAuthentication,
BasicAuthentication, TokenAuthentication
from rest_framework.decorators import api_view, permission_classes,
authentication_classes
from rest_framework.permissions import IsAuthenticated
```

```
from rest_framework.response import Response

from .decorators import define_usage
from .models import Agreement


def counting_agreements():
    val = {}
    all = Agreement.objects.all()
    for el in all:
        if f"{el.stop_date.year}" not in val:
            val[f"{el.stop_date.year}"] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        val[f"{el.stop_date.year}"][el.stop_date.month - 1] = \
            (all.filter(stop_date__month=el.stop_date.month).aggregate(
                Count('stop_date')))['stop_date__count']
    return val


@define_usage(returns={'val': 'Dict'})
@api_view(['GET'])
@authentication_classes((SessionAuthentication, BasicAuthentication,
TokenAuthentication))
@permission_classes((IsAuthenticated,))
def api_all_tasks(request):
    return Response(counting_agreements())
```

decorators.py

```
from functools import wraps
from rest_framework.permissions import IsAuthenticated


#Gives function-based API views a lot of the same capabilities as class-based views
def define_usage(params=None, returns=None):
    def decorator(function):
        cls = function.view_class
        header = None
        # Is authentication required to access this view?
        if IsAuthenticated in cls.permission_classes:
            header = {'Authorization': 'Token String'}
        # Build a list of the valid methods, but take out 'OPTIONS'
        methods = [method.upper() for method in cls.http_method_names if method != 'options']
        # Build response dictionary
        usage = {'Request Types': methods, 'Headers': header, 'Body': params, 'Returns': returns}

        # Prevent side effects
        @wraps(function)
        def _wrapper(*args, **kwargs):
            return function(*args, **kwargs)
        _wrapper.usage = usage
        return _wrapper
    return decorator
```

base.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
      integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">

  <title>Угоди</title>
</head>
<body>

<div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">
  <h2 class="my-0 mr-md-auto font-weight-normal">Admin</h2>
  <nav class="my-2 my-md-0 mr-md-3">
    {#        <a class="p-2 text-dark" href="{% url 'add_channel' %}">Створити</a>#}
    {#        <a class="p-2 text-dark" href="{% url 'home' %}">Додому</a>#}
    {#        <a class="p-2 text-dark" href="{% url 'table' %}">Таблиця</a>#}
  </nav>
      <a class="btn btn-outline-primary" href="{% url 'logout' %}">Выход</a>
</div>
```

```
{% block content %}{% endblock %}
{% block table %}{% endblock %}
{% block channel %}{% endblock %}
<div class="container-sm">
   <footer class="pt-4 my-md-5 pt-md-5 border-top">
      <div class="row">
         <div class="col-12 col-md">
            <small class="d-block mb-3 text-muted">© 2020</small>
         </div>
      </div>
   </footer></div></body></html>
```

agreement.html

```
{% extends "TemplatesForAdmin/base.html" %}
{% block table %}
<script
      src="https://code.jquery.com/jquery-3.5.1.min.js"
      integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
      crossorigin="anonymous"></script>
<style>

      table {
         border: 1px solid #69c;
      }

      th {
```

```
    font-weight: normal;
    color: #039;
    border-bottom: 1px dashed #69c;
    padding: 12px 17px;
}

td {
    color: #669;
    padding: 7px 17px;
}

tr:hover td {
    background: #ccddff;
}

.animated-search-form[type=search] {
    width: 10rem;
    border: 0.125rem solid #e6e6e6;
    box-shadow: 0 0 1rem rgba(0, 0, 0, 0.18);
    border-radius: 0;
    background-image: url("https://image.ibb.co/i7NbrQ/search_icon_15.png");
    background-position: 0.625rem 0.625rem;
    background-repeat: no-repeat;
    padding: 0.75rem 1.25rem 0.75rem 2rem;
    transition: width 0.4s ease-in-out;
}

.animated-search-form[type=search]:focus {
    width: 70%;
```

```
    }
</style>

<div class="row">
  <div class="col-2"></div>
  <div class="col-7">
    <div class="container-xl">
      <h1>Угоди</h1>

      <form id="form" name="form" method="GET">
        <input type="search" placeholder="Пошук.." class="animated-search-form"
name="q">
      </form>
      <br>

      <form id="myform" name="myform" method="post">
        {% csrf_token %}
        <a type="submit" href="{% url 'add_agreement' %}" class="btn btn-
primary">Створити</a>
        <button type="submit" form="myform" name="edit" class="btn btn-
warning">Редагувати</button>
        <button type="submit" form="myform" name="del" class="btn btn-
danger">Видалити</button>

        <br>
        <br>
        <table width="100%">
          <tr>
            <th width="1%"></th>
            <th width="1%">№</th>
```

```
                <th width="25%">Угода</th>
                <th width="25%">Парламентер</th>
                <th width="25%">Дата початку дії договору</th>
                <th width="25%">Дата закінчення дії договору</th>
            </tr>

            {% for el in pages %}
            <tr>
                <td><input type="checkbox" value={{ el.id }} id={{ el.id }}
                        name={{ el.id }}></td>
                <td>{{ forloop.counter }}</td>
                <td><a href="/admin/agreement/{{ el.id }}">{{ el.agreement|safe
}}</a></td>
                <td>{{ el.negotiator }}</td>
                <td>{{ el.start_date|date:'d.m.Y' }}</td>
                <td>{{ el.stop_date|date:'d.m.Y' }}</td>
            </tr>
            {% endfor %}

        </table>
    </form>
    <br>

    {% if is_paginated %}
    <ul class="pagination">
        <li class="page-item {% if not prev_url %}disabled{% endif %}">
            <a class="page-link" href="{{ prev_url }}" aria-label="Previous">
                <span aria-hidden="true">&laquo;</span>
            </a>
        </li>
```

```
            {% for page in pages.paginator.page_range %}
            {% if pages.number == page %}
            <li class="page-item active">
              <a class="page-link" href="?page={{ page }}">
                {{ page }}<span class="sr-only">(current)</span>
              </a>
            </li>
            {% elif page > pages.number|add:-3 and page < pages.number|add:3 %}
            <li class="page-item">
              <a class="page-link" href="?page={{ page }}">{{ page }}
              </a>
              {% endif %}
              {% endfor %}
            <li class="page-item {% if not next_url %}disabled{% endif %}">
              <a class="page-link" href="{{ next_url }}" aria-label="Next">
                <span aria-hidden="true">&raquo;</span>
              </a>
            </li>

          </ul>
          {% endif %}
        </div>
      </div>

      <div class="col-3">
        <script>
            {#src = "https://code.jquery.com/jquery-3.5.1.min.js"#}
            {#integrity = "sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="#}

            crossorigin = "anonymous" >
```

```
$(document).ready(function () {


    var requestURL = 'http://127.0.0.1:8000/admin/all/';
    var request = new XMLHttpRequest();
    request.open('GET', requestURL);
    request.responseType = 'json';
    request.send();
    request.onload = function () {
      var superHeroes = request.response;
      let nowYear = new Date().getFullYear()
      document.getElementById("year").innerHTML = ('<text>' +
nowYear + '</text>')

      function writeInHTML(nowYear, obj) {
        const arr = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN",
"JUL", "AUG", "SEP", "OCT", "NOV", "DEC"]
        let i = 0
        if (obj[nowYear] !== undefined) {
          while (arr) {
            if (i >= 12) break
            if (obj[nowYear][i] !== 0) {
              document.getElementById(arr[i]).innerHTML = ('<text>'
+ arr[i] +
                ' <div class="flex-item1"><p class="col-md-0 offset-
md-0" >'
                + obj[nowYear][i] + '</p></div>' + '</text>')
            } else {
              document.getElementById(arr[i]).innerHTML = ('<text>'
+ arr[i] + ' </text>')
```

```
                    }
                    i++

                }
            } else {
                while (arr) {
                    if (i >= 12) break
                    document.getElementById(arr[i]).innerHTML = ('<text>' +
arr[i] + '</text>')
                    i++
                }
            }
        }

        writeInHTML(nowYear, superHeroes)
        $('button.prewbutton').on('click', function () {
            const a = Number($('text.flex-item2').text()) - 1;
            document.getElementById("year").innerHTML = ('<text>' + a +
'</text>');
            writeInHTML(a, superHeroes)
        })
        $('button.nextbutton').on('click', function () {
            let a = Number($('text.flex-item2').text()) + 1;
            document.getElementById("year").innerHTML = ('<text>' + a +
'</text>')
            writeInHTML(a, superHeroes)
        })
    }
}
);
```

```
    </script>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
    <style>
        .flex-container {
            padding: 0;
            margin: 0;
            list-style: none;
            display: -webkit-box;
            display: -moz-box;
            display: -ms-flexbox;
            display: -webkit-flex;
            display: flex;
            -webkit-flex-flow: row wrap;
            justify-content: space-around;
            max-width: 350px;
        }

        .flex-item {
            background: #efefef;
            padding: 4px;
            width: 80px;
            height: 66px;
            margin-top: 10px;
            line-height: 47px;
            color: #424242;
            font-size: 1.2em;
            text-align: center;
            border-radius: 5px;
```

```
    position: relative
}

.flex-item1 {
    background: #d45f4c;
    padding: 3px;
    width: 20px;
    height: 20px;
    line-height: 15px;
    color: #fcf2f1;
    font-size: 0.8em;
    text-align: center;
    border-radius: 4px;
    z-index: 1;
    position: relative;
    left: 52px;
    bottom: 48px;
}

.flex-item2 {
    background: #ffffff;
    padding: 4px;
    width: 200px;
    height: 20px;
    margin-top: 5px;
    line-height: 25px;
    color: #424242;
    font-weight: bold;
    font-size: 1.6em;
    text-align: center;
```

```
        }


    </style>
    <div class="container">
      <div class="flex-container">
        <button name="prew" class="prewbutton">
          <svg width="2em" height="2em" viewBox="0 0 16 16" class="bi bi-
chevron-double-left"
            fill="currentColor"
            xmlns="http://www.w3.org/2000/svg">
            <path fill-rule="evenodd"
              d="M8.354 1.646a.5.5 0 0 1 0 .708L2.707 8l5.647 5.646a.5.5 0 0 1-
.708.708l-6-6a.5.5 0 0 1 0-.708l6-6a.5.5 0 0 1 .708 0z"/>
            <path fill-rule="evenodd"
              d="M12.354 1.646a.5.5 0 0 1 0 .708L6.707 8l5.647 5.646a.5.5 0 0 1-
.708.708l-6-6a.5.5 0 0 1 0-.708l6-6a.5.5 0 0 1 .708 0z"/>
          </svg>
        </button>
        <text class="flex-item2" id="year"></text>
        <button name="next" class="nextbutton">
          <svg width="2em" height="2em" viewBox="0 0 16 16" class="bi bi-
chevron-double-right"
            fill="currentColor"
            xmlns="http://www.w3.org/2000/svg">
            <path fill-rule="evenodd"
              d="M3.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 0 .708l-6 6a.5.5 0 0
1-.708-.708L9.293 8 3.646 2.354a.5.5 0 0 1 0-.708z"/>
            <path fill-rule="evenodd"
```

```
                    d="M7.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 0 .708l-6 6a.5.5 0 0
1-.708-.708L13.293 8 7.646 2.354a.5.5 0 0 1 0-.708z"/>
                </svg>
            </button>
            <div class="flex-item" id="JAN"></div>
            <div class="flex-item" id="FEB"></div>
            <div class="flex-item" id="MAR"></div>
            <div class="flex-item" id="APR"></div>
            <div class="flex-item" id="MAY"></div>
            <div class="flex-item" id="JUN"></div>
            <div class="flex-item" id="JUL"></div>
            <div class="flex-item" id="AUG"></div>
            <div class="flex-item" id="SEP"></div>
            <div class="flex-item" id="OCT"></div>
            <div class="flex-item" id="NOV"></div>
            <div class="flex-item" id="DEC"></div>
        </div>
    </div>
  </div>

</div>
{% endblock %}
```

AddAgreement.html

```
{% extends "TemplatesForAdmin/base.html" %}
{% block content %}
  <div class="container">
    <form action="{% url 'add_agreement' %}" method="post">
      {% csrf_token %}
      <div class="card" style="width: 100%;">
        <div class="card-header">
          <h3>Угода</h3>
        </div>
        <ul class="list-group list-group-flush">
          <li class="list-group-item">
            {{ form.agreement.label }} {{ form.agreement }}<br><br>
            {{ form.negotiator.label }} {{ form.negotiator }}<br><br>
            {{ form.start_date.label }} {{ form.start_date }}<br><br>
            {{ form.stop_date.label }} {{ form.stop_date }}<br><br>
            {{ form.subject_of_agreement.label }} {{ form.subject_of_agreement
}}<br><br>
            {{ form.debit.label }} {{ form.debit }}<br><br>
            {{ form.credit_turnover.label }} {{ form.credit_turnover }}<br><br>
          </li>

        </ul>
      </div><br>
      <br>
      <blockquote class="blockquote text-center">
        <input type="submit" value="Відправити">
      </blockquote>
      <br>
```

```
      </form>
   </div>
{% endblock %}
```

Login.py

```
<!DOCTYPE html>
<html>

<head>
  <title>Login</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLP
MO" crossorigin="anonymous">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.1/css/all.css"
integrity="sha384-
gfdkjb5BdAXd+lj+gudLWI+BXq4IuLW5IT+brZEZsLFm++aCMlF1V92rMkPaX4PP"
crossorigin="anonymous">


  <style>
    body,
    html {
      margin: 0;
      padding: 0;
      height: 100%;
```

```
    background: #ffffff !important;
  }
  .user_card {
    width: 350px;
    margin-top: auto;
    margin-bottom: auto;
    background: #efebeb;
    position: relative;
    display: flex;
    justify-content: center;
    flex-direction: column;
    padding: 10px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    -webkit-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0,
0.19);
    -moz-box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    border-radius: 5px;

  }

  .form_container {
    margin-top: 20px;
  }

  #form-title{
    color: #000000;

  }

  .login_btn {
```

```
    width: 100%;
    background: #33ccff !important;
    color: white !important;
  }
  .login_btn:focus {
    box-shadow: none !important;
    outline: 0px !important;
  }
  .login_container {
    padding: 0 2rem;
  }
  .input-group-text {
    background: #f7ba5b !important;
    color: white !important;
    border: 0 !important;
    border-radius: 0.25rem 0 0 0.25rem !important;
  }
  .input_user,
  .input_pass:focus {
    box-shadow: none !important;
    outline: 0px !important;
  }

  #messages{
    background-color: grey;
    color: #fff;
    padding: 10px;
    margin-top: 10px;
  }
</style>
```

```
</head>
<body>
  <div class="container h-100">
    <div class="d-flex justify-content-center h-100">
      <div class="user_card">
        <div class="d-flex justify-content-center">


          <h3 id="form-title">LOGIN</h3>
        </div>
        <div class="d-flex justify-content-center form_container">
          <form method="POST" action="">
            {% csrf_token %}
            <div class="input-group mb-3">
              <div class="input-group-append">
                <span class="input-group-text"><i class="fas fa-user"></i></span>
              </div>

              <input type="text" name="username" placeholder="Username..."
class="form-control">
            </div>

            <div class="input-group mb-2">
              <div class="input-group-append">
                <span class="input-group-text"><i class="fas fa-key"></i></span>
              </div>

              <input type="password" name="password" placeholder="Password..."
class="form-control" >
```

```
            </div>

              <div class="d-flex justify-content-center mt-3 login_container">
                <input class="btn login_btn" type="submit" value="Login">
                 </div>
            </form>

          </div>

          {% for message in messages %}
            <p id="messages">{{message}}</p>
          {% endfor %}

        </div>
      </div>
    </div>
</body>
</html>
```

Homepage.html

```
{% extends "TemplatesForAdmin/base.html" %}
{% block content %}
   <div class="container">
   <h2>Таблици</h2>
   <ul class="list-group list-group-flush">
      <li class="list-group-item"><a href="{% url 'agreement'
%}">Соглашения</a></li>
      <li class="list-group-item"><a href="URL">Пользователи</a></li>
   </ul>
   </div>
{% endblock %}
```

ДОДАТОК В Інтерфейс WEB-додатку

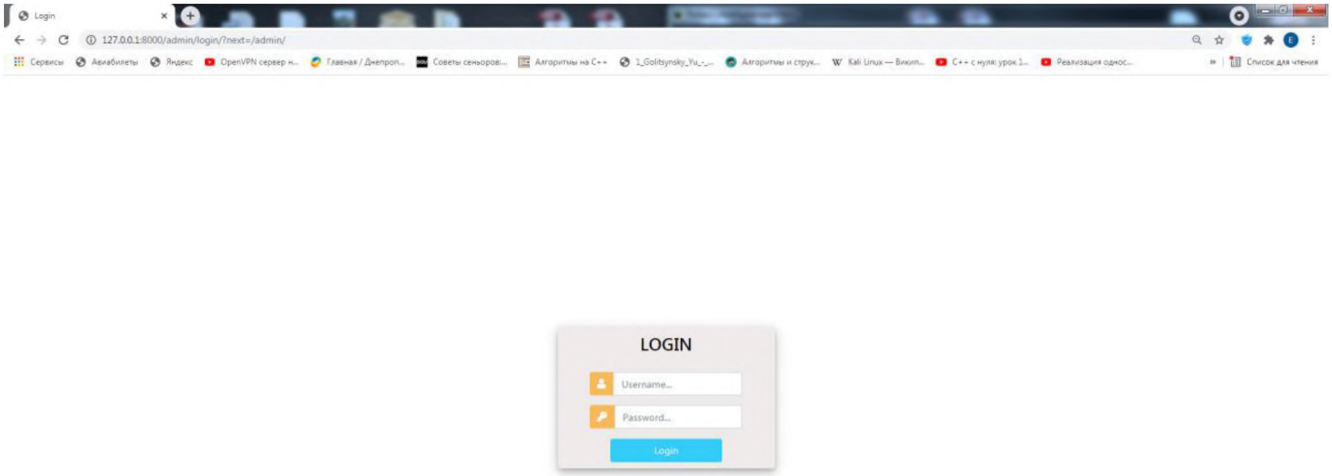


Рисунок В.1 - Вигляд результату роботи файлу login.html

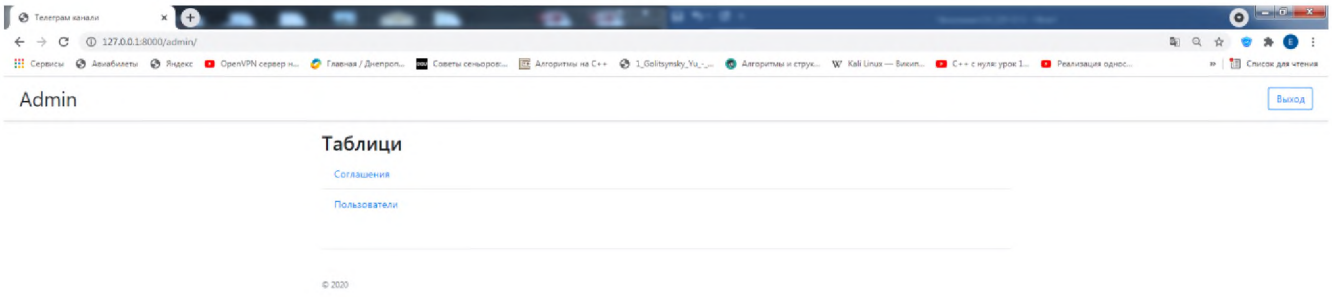


Рисунок В.2 - Вигляд результату роботи файлу Homepage.html

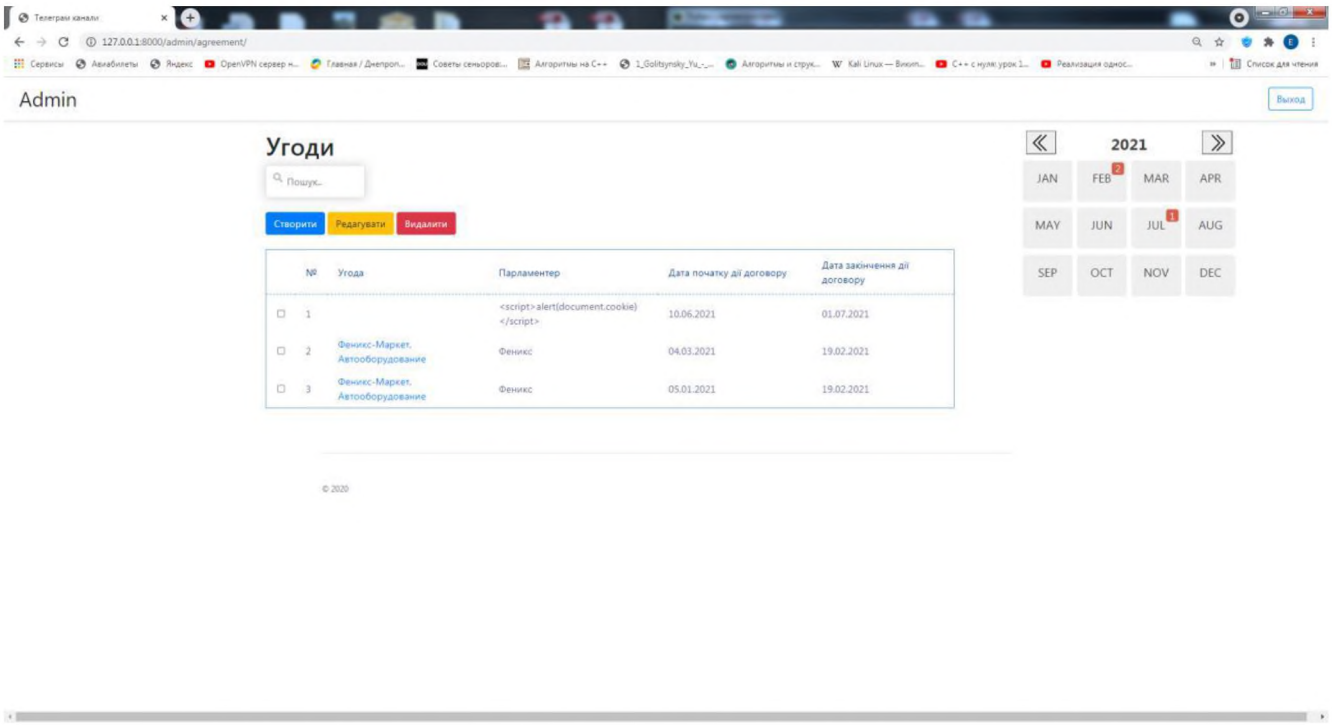


Рисунок В.3 - Вигляд результату роботи файлу agreement.html

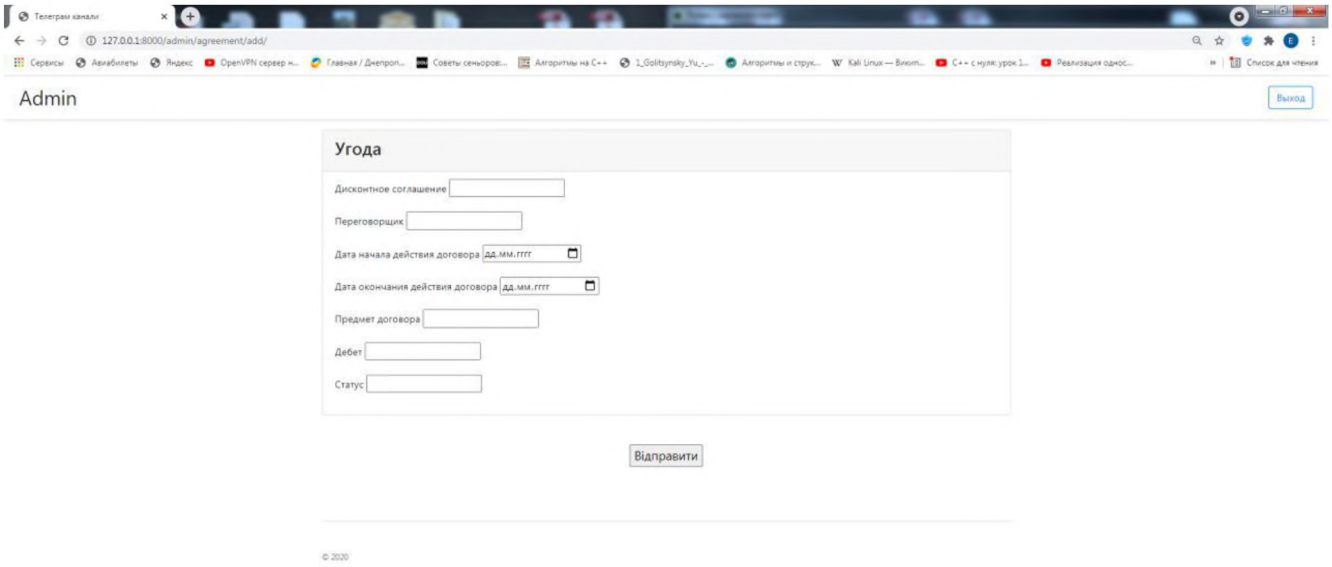


Рисунок В.4 - Вигляд результату роботи файлу AddAgreement.html

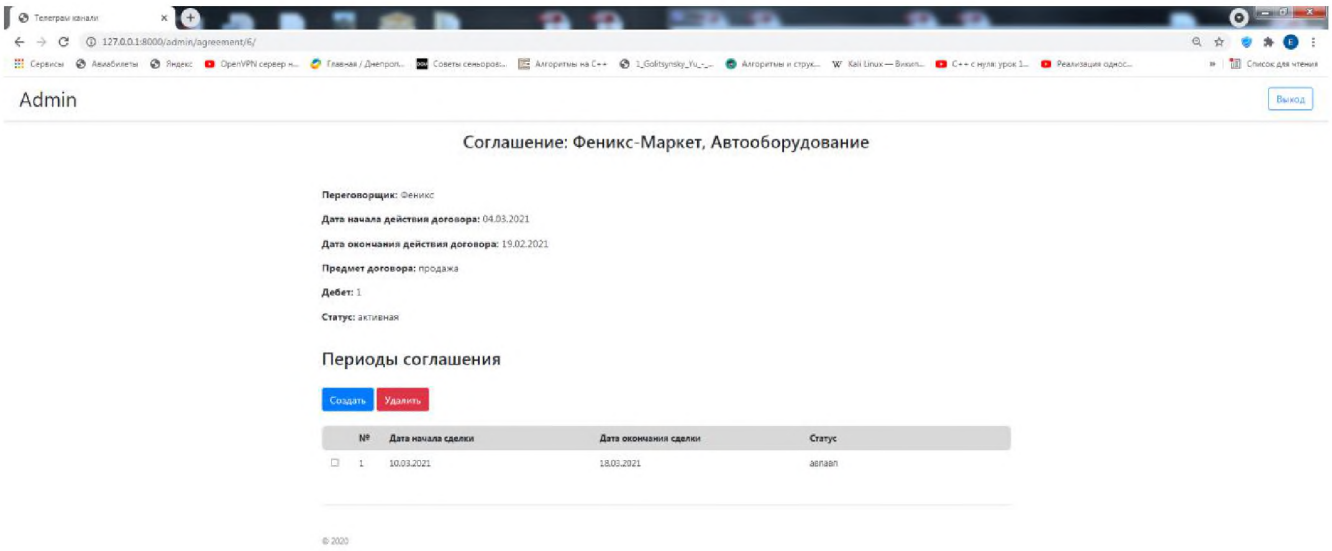


Рисунок В.5 - Вигляд результату роботи файлу AgreementPage.html

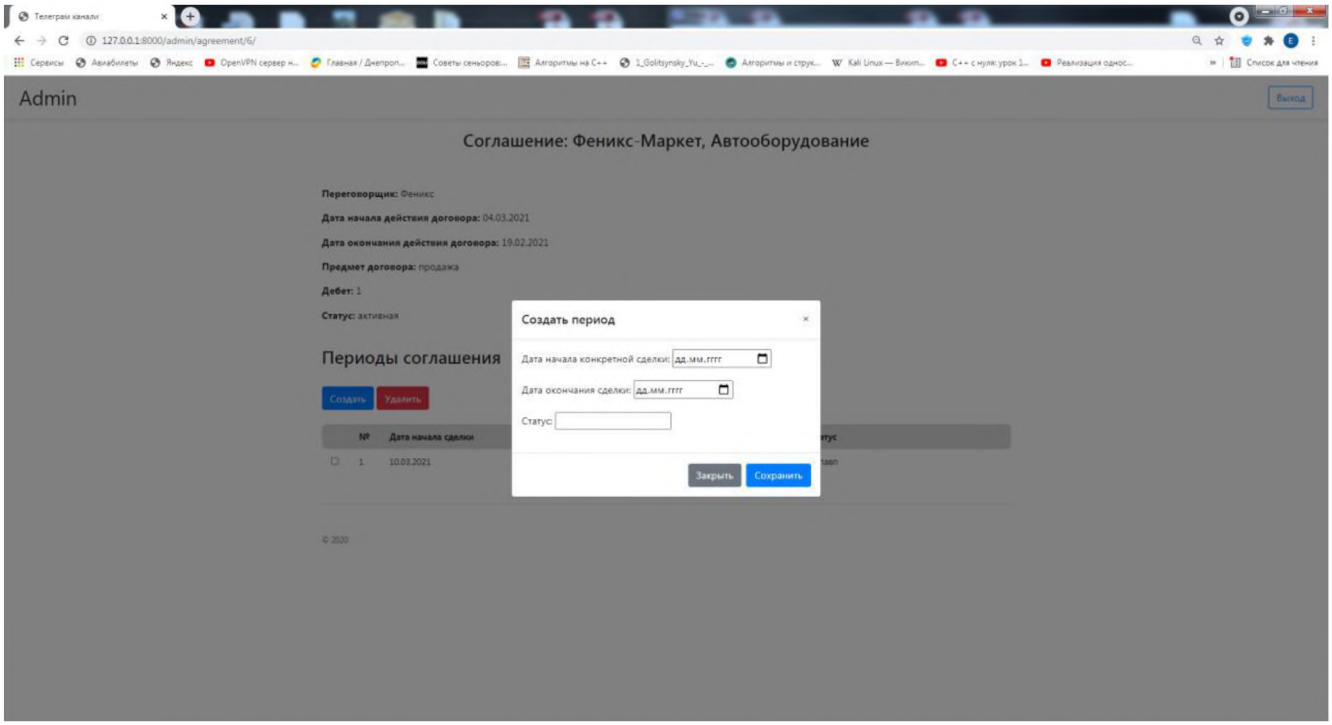


Рисунок В.6 - Вигляд результату роботи файлу AgreementPage.html

ДОДАТОК Г. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітка
Документація				
1	А4	Реферат	1	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	1	
5	А4	Стан питання. Постановка задачі	19	
6	А4	Спеціальна частина	17	
7	А4	Економічний розділ	6	
8	А4	Висновок	1	
9	А4	Перелік посилань	2	
10	А4	Додаток А	6	
11	А4	Додаток Б	31	
12	А4	Додаток В	3	
13	А4	Додаток Г	1	
14	А4	Додаток Д	1	
15	А4	Додаток Е	1	
16	А4	Додаток Є	1	
17	А4	Додаток Ж	1	

ДОДАТОК Д. Перелік документів на оптичному носії

1. Васильков Є.К_125-17-1.docx
2. Васильков Є.К_125-17-1.pdf
3. Васильков Є.К_125-17-1.pptx
4. Васильков Є.К_125-17-1

ДОДАТОК Е. Відгук керівника дипломної роботи

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник дипломної роботи
професор Кагадій Т.С. Підпис: _____

ДОДАТОК Є. Відгук керівника економічного розділу

Керівник економічного розділу
к.е.н., доц. Пілова Д.П. Підпис: _____

ДОДАТОК Ж.
В І Д Г У К
на кваліфікаційну роботу студента групи 125-17-1
Василькова Євгена Костянтиновича

на тему: «Способи реалізації послуг безпеки для WEB-додатків при використанні фреймворків Django та Flask»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, викладених на_____сторінках.

Метою кваліфікаційної роботи є визначення переліку та рівнів реалізації послуг безпеки при застосуванні фреймворків Django та Flask.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 125 «Кібербезпека». Для досягнення поставленої мети в кваліфікаційній роботі вирішуються наступні задачі: аналіз сучасного стану безпеки WEB-додатків, аналіз особливостей технологій розробки WEB-додатків, обґрунтування вимог до захисту WEB-додатків, визначення переліку та рівнів реалізації послуг безпеки при застосуванні фреймворків.

Виконано порівняльний аналіз ефективності використання фреймворків Django та Flask.

Практичне значення результатів кваліфікаційної роботи полягає у розробці програмної реалізації елементів механізмів захисту WEB-додатків.

До недоліків роботи можна віднести недостатнє обґрунтування рівнів реалізації деяких послуг безпеки

Оформлення пояснювальної записки до кваліфікаційної роботи виконано з незначними відхиленнями від стандартів.

За час дипломування Васильков Є.К. проявив себе фахівцем, здатним достатньо самостійно вирішувати поставлені задачі та заслуговує присвоєння кваліфікації бакалавра за спеціальністю 125 Кібербезпека, освітньо-професійна програма «Кібербезпека» .

Рівень запозичень у кваліфікаційній роботі не перевищує вимог "Положення про систему виявлення та запобігання плагіату".

Кваліфікаційна робота заслуговує оцінки «_____».

Керівник кваліфікаційної роботи, професор **Кагадій Т.С.**

Керівник спец. розділу, ст. викладач **Кручінін О.В.**