Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка програмного модуля захисту даних навчально-методичного відділу КНТЕУ»

(за матеріалами навчально-методичного відділу КНТЕУ м.Київ)

Студента 2 курсу, 6 групи, факультету обліку, аудиту та інформаційних систем, денної форми навчання 122 «Комп'ютерні науки» _____ *(підпис студента)* Свічкаря Олексія Владиславовича

Науковий керівник, доктор технічних наук, професор кафедри комп'ютерних наук _____ *(підпис наукового керівника)* Яловець Андрій Леонідович

Гарант освітньої програми, доктор фізико-математичних наук, професор _____ *(підпис гаранта освітньої програми)* Пурський Олег Іванович

Київ 2019

Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук

Факультет обліку, аудиту та інформаційних систем

Напрям підготовки 122 «Комп'ютерні науки»

Затверджую

Завідувач кафедри

Пурський Олег Іванович

_____

«___» _____ 2018 р.

Завдання

на випускний кваліфікаційний проект студенту

Свічкарю Олексію Владиславовичу

1. Тема випускного кваліфікаційного проекту

Розробка програмного модуля захисту даних навчально-методичного відділу КНТЕУ

Затверджено наказом ректора від 7 листопада 2018 р. № 4181

2. Строк здачі студентом закінченого проекту _____

3. Цільова установка та вихідні дані до проекту

Мета проекту – розробка програмного модуля захисту даних для забезпечення цілісності, доступності та конфіденційності інформації і методичних матеріалів навчально-методичного відділу КНТЕУ.

Об'єкт дослідження – процеси перегляду, обробки та зберігання даних навчально-методичного відділу КНТЕУ.

Предмет дослідження – методи і технології захисту даних.

Гіпотеза дослідження – припущення, що розроблений програмний модуль захисту даних повністю усуне загрозу не санкціонованого доступу до матеріалів навчально-методичного відділу та унеможливить впровадження поширених інтернет-атак.

4. Перелік графічного матеріалу: 12 рисунків, 3 таблиці.

5. Зміст випускного кваліфікаційного проекту – перелік питань за кожним розділом

В першому розділі необхідно розглянути теоретичні засади інформаційної безпеки, захисту даних веб-додатків. Розглянути та обрати види захисту веб-додатків та серверів.

В другому розділі обґрунтувати проблематику, проаналізувати поширені атаки на веб-додатки, розглянути постановку задачі та обрати програмні засоби для реалізації рішення.

В третьому розділі провести структурно-функціональний аналіз, побудувати логічну модель, здійснити опис впровадження програмного забезпечення та провести функціональне тестування.

6. Календарний план виконання проекту

№ пор.	Етапи виконання випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		за планом	фактично
1	Вибір теми випускного кваліфікаційного проекту	01.11.18	01.11.18
2	Розробка та затвердження завдання на випускний кваліфікаційний проект	15.11.18	15.11.18
3	Вступ	01.06.19	01.06.19
4	Розділ 1.	25.06.19	25.06.19
5	Розділ 2.	02.09.19	02.09.19
6	Підготовка статті у збірник наукових статей магістрів	09.09.19	09.09.19
7	Розділ 3.	21.10.19	21.10.19
8	Висновки	01.11.19	01.11.19
9	Здача випускного кваліфікаційного проекту на кафедру науковому керівнику	05.11.19	05.11.19
10	Попередній захист випускного кваліфікаційного проекту	20.11.19	20.11.19

11	Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту	22.11.19	22.11.19
12	Представлення готового зшитого випускного кваліфікаційного проекту на кафедру	25.11.19	25.11.19
13	Публічний захист випускного кваліфікаційного проекту	03.12.19	

7. Дата видачі завдання «15» листопада 2018 р.

8. Керівник випускного кваліфікаційного проекту Яловець Андрій Леонідович

9. Керівник магістерської програми Пурський Олег Іванович

10. Завдання прийняв до виконання студент-дипломник Свічкар О.В.

11. Відгук керівника випускного кваліфікаційного проекту

_____

_____

_____

_____

_____

_____

_____

_____

_____

Керівник випускного кваліфікаційного проекту_____

12. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента Свічкаря О.В. може бути допущений до захисту в екзаменаційній комісії.

Керівник випускного кваліфікаційного проекту _____

Завідувач кафедри _____

« ___ » _____ 2019 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ 11
1.1 Поняття інформаційної безпеки 11
1.2 Критерії оцінки інформаційної безпеки 12
1.3 Аналіз та вибір методів захисту інформації 13
ВИСНОВКИ ДО РОЗДІЛУ 1 17
РОЗДІЛ 2. ОГЛЯД ПРОБЛЕМАТИКИ, ОПИС ТА ВИБІР ВИКОРИСТАНИХ СИСТЕМ ТА ПРОГРАМНИХ ЗАСОБІВ 18
2.1 Обґрунтування проблеми 18
2.2 Аналіз поширених атак на веб-ресурси в сфері освіти 19
2.3 Постановка задачі 20
2.4 Вибір програмних засобів для реалізації 22
ВИСНОВКИ ДО РОЗДІЛУ 2 27
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ЗАХИСТУ ДАНИХ 28
3.1 Проектування і структурно-функціональний аналіз 28
3.2 Розробка програмної частини 31
3.3 Впровадження програмного забезпечення 33
3.4 Функціональне тестування 36
ВИСНОВКИ ДО РОЗДІЛУ 3 38
ВИСНОВКИ ТА ПРОПОЗИЦІЇ 39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 42
ДОДАТОК А 44
ДОДАТОК Б 45
ДОДАТОК В 51
ДОДАТОК Г 52

АНОТАЦІЯ

Тема випускного кваліфікаційного проекту: «Розробка програмного модуля захисту даних навчально-методичного відділу КНТЕУ».

Об’єм роботи 55 сторінок, на яких розміщено 12 рисунків і 3 таблиці. При написанні використано 23 джерела.

Ключові слова: безпека, модуль, захист, веб-додаток, розробка.

Об’єктом дослідження є процеси перегляду, обробки та зберігання даних навчально-методичного відділу КНТЕУ.

Предметом дослідження виступають методи, моделі і технології захисту інформації та даних веб-додатків.

Випускний кваліфікаційний проект складається зі вступу, трьох розділів, висновків по написаним розділам, загальних висновків та пропозицій.

У вступі розкривається актуальність дослідження, ставиться мета роботи, визначаються об’єкт, предмет і гіпотеза дослідження.

В першому розділі розглянуто теоретичні засади інформаційної безпеки, критерії захисту інформації. Проаналізовано та обрано методи і засоби захисту інформації.

В другому розділі обґрунтувано проблематику, проаналізовано поширені загрози веб-додатків в сфері освіти і науки, здійснено постановку задачі та обрано програмні засоби для реалізації рішення.

В третьому розділі проведено структурно-функціональний аналіз, побудувано логічну модель, описано впровадження програмного модуля та проведено функціональне тестування.

Висновки містять основні результати та пропозиції щодо подальшого дослідження даної теми.

					КНТЕУ_122_2019			
Зм.	*Аркуш*	*№ документу*	*Підпис*	*Дата*	*Розробка програмного модуля захисту даних навчально-методичного відділу КНТЕУ*	*Стадія*	*Аркуш*	*Аркушів*
Зав. кафедри		Пурський О.І.				*А*	6	55
Керівник		Яловець А.Л.						
Гарант		Пурський О.І			*Анотація*	*Кафедра комп’ютерних наук група 2-6м*		
Розробив		Свічкар О.В.						
Перевірив		Яловець А.Л.						

ABSTRACT

The subject of the final qualification project: "Development of a software module for data protection of the KNUTE training and methodological department".

55 pages of work with 12 figures and 3 tables. 23 sources were used when writing.

Keywords: security, software, protection, web application, development.

The subject of the study is the processes of reviewing, processing and storing data of the KNUTE department of training and methodology.

The subject of the research is methods, models and technologies of protection of information and data of web applications.

The final qualification project consists of an introduction, three sections, conclusions from the written sections, general conclusions and proposals.

The introduction reveals the relevance of the study, establishes the purpose of the work, defines the object, subject and hypothesis of the study.

The first section discusses the theoretical foundations of information security, information security criteria. Methods and means of information security are analyzed and selected.

The second section discusses issues, analyzes common threats to web applications in education and science, executes the problem statement, and selects software to implement the solution.

In the third section, structural and functional analysis is performed, a logical model is built, the implementation of the software module is described, and functional testing is performed.

The conclusions provide the main conclusions and suggestions for further study of the topic.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне Забезпечення

ISO – International Organization for Standardization

CIA – Confidentiality, Integrity and Availability

HTTP(S) – HyperText Transfer Protocol (Secure)

БД – база даних

DDoS – Distributed Denial of Service

SSL – Secure Sockets Layer

ERP – Enterprise Resource Planning

XSS – Cross-Site Scripting

CSRF – Cross-Site Request Forgery

SQL – Structured Query Language

MVC – Model-View-Controller

DRY – Don`t repeat yourself

URL – Uniform Resource Locator

RSS – Rich Site Summary

API – Application Programming Interface

ВСТУП

Останнім часов відбувається швидкий перехід від традиційної паперової технології зберігання, обробки та передавання інформації до електронної.

У зв'язку з постіним розвитком технологій високошвидкісного доступу в Інтернет важливі компоненти бізнесу переміщаються в середовище Web. Системи типу Клієнт-Банк, публічні сайти організацій, інтернет-магазини, новинні, розважальні, торговельні майданчики, а також державні портали являються обов'язковою складовою всесвітньої мережі.

Доступ до подібних систем можливий з будь-якого куточка планети, за умови наявності Інтернету. Система однаково функціонує незалежно від пристрою та платформи, навідміну від традиційного комп'терного ПЗ. Саме тому, для зберігання та обробки даних навчально-методичного відділу КНТЕУ використовується середовище Web, а саме розподілений веб-додаток.

Проте, зафіксоване відставання технологій захисту інформації та через свою доступність веб-додатки часто стають привабливою здобиччю для зловмисників. З кожним роком все більш явною стає тенденція різкого збільшення ролі і значення електронної інформації в усіх сферах життя суспільства, а постійне ускладнення програмних засобів викликає зменшення їх надійності та збільшення кількості уразливих місць.

З розвитком інформаційних технологій почастішали випадки атак на веб-ресурси з метою отримання конфіденційної інформації. Втрата конфіденційних даних будь-якої організації або користувача несе в собі загрози, як мінімум, фінансових втрат, адже отриманою інформацією можуть скористатися конкуренти та зловмисники.

Відповідно захист інформації, що зберігається та обробляється у веб-

додатках має важливе значення. Необхідно забезпечувати доступність, цілісність і конфіденційність інформації.

Постає проблема інформаційної безпеки та захисту конфіденційної інформації і персональних даних, яка визначає актуальність обраної теми.

Для вирішення даної проблеми виникає необхідність створення такого програмного модуля, який зміг би надати безпечне та безперервне функціонування і забезпечити захист даних веб-додатку.

Метою випускного кваліфікаційного проекту є розробка програмного модуля, що забезпечить цілісність, безперервну доступність та конфіденційність даних навчально-методичного відділу КНТЕУ.

Об'єктом дослідження даної роботи є процеси перегляду, обробки та зберігання даних навчально-методичного відділу КНТЕУ.

Предметом дослідження виступає програмний модуль захисту даних.

Гіпотезою дослідження є припущення, що розроблений програмний модуль захисту даних повністю усуне загрозу не санкціонованого доступу до матеріалів навчально-методичного відділу та унеможливить впровадження поширених інтернет-атак.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

1.1 Поняття інформаційної безпеки

В сучасному суспільстві саме інформація стає найважливішим стратегічним ресурсом, основною виробничою силою, що забезпечує його подальший розвиток. Ось чому, подібно будь-яким іншим традиційно існуючим ресурсам, інформація також потребує особливого захисту.

Поряд з терміном «захист інформації» також широко використовується термін «інформаційна безпека». Якщо захист інформації характеризує процес створення умов, що забезпечують необхідну захищеність інформації, то інформаційна безпека відображає досягнутий стан такої захищеності.

Інформаційна безпека – це стан захищеності систем обробки і зберігання даних, при якому забезпечено конфіденційність, доступність і цілісність інформації. Метою реалізації інформаційної безпеки будь-якого об'єкта є побудова системи забезпечення інформаційної безпеки даного об'єкту [2].

Поняття інформаційної безпеки не обмежується безпекою технічних інформаційних систем чи безпекою інформації у чисельному чи електронному вигляді, а стосується усіх аспектів захисту даних чи інформації незалежно від форми, у якій вони перебувають [15].

При розробці програмного модуля захисту даних необхідно взяти до уваги, що впливи на безпеку системи можуть бути різного характеру, тому необхідно використовувати комплексний підхід для вирішення проблеми, а не діяти лише на окремі випадки в історії безпеки інформаційних систем.

1.2 Критерії оцінки інформаційної безпеки

Методологічною базою для визначення вимог захисту інформаційної системи від несанкціонованого доступу, створення захисних модулів та оцінки ступені захищеності являються критерії оцінки інформаційної безпеки.

З допомогою критеріїв можливо порівняти різні механізми захисту інформації та визначити необхідну функціональність таких механізмів у розробці модулів захисту даних.

Для характеристики основних критеріїв інформаційної безпеки та властивостей інформації як об'єкта захисту часто використовується модель CIA:

- цілісність – означає неможливість модифікації неавторизованим користувачем;
- доступність – властивість інформації бути отриманою авторизованим користувачем, за наявності у нього відповідних повноважень, в необхідний для нього час;
- конфіденційність – властивість інформації, яка полягає в тому, що інформація не може бути отримана неавторизованим користувачем.

Відповідно до властивостей інформації, виділяють перелік загроз її безпеки (рис 1.1).

Рис 1.1. Загрози безпеки інформації

Звідси можна виділити такі аспекти захисту даних:

- захист від несанкціонованого ознайомлення з даними;
- захист даних від несанкціонованої модифікації;
- захист (забезпечення) доступу до даних, а також можливості їх використання.

Доступність забезпечується як підтриманням систем в робочому стані так і завдяки способам, які дозволяють швидко відновити втрачені чи пошкоджені дані.

Для уніфікації стандартів в області інформаційних технологій загальні критерії оцінки безпеки описані в міжнародному стандарті ISO 15408.

1.3 Аналіз та вибір методів захисту інформації

Ключовим моментом в процесі розробки модулю є аналіз та вибір методів і засобів забезпечення захисту інформації.

Методи можна класифікувати наступним чином:

- перешкода;
- маскування;
- управління доступом;
- регламентація;
- примус;
- спонукання.

Для аналізу цих методів необхідно детально зупинитися на кожному з них. Таким чином,

1. Перешкода – метод фізичного перешкоджання шляху зловмисника до інформації, що захищається (сигналізація, замки і т.д.).

2. Маскування – метод захисту інформації шляхом її криптографічного закриття. Цей метод захисту широко застосовується за кордоном, як при зберіганні інформації, так і при обробці. При передачі інформації по каналах зв'язку великої протяжності цей метод дійсно надійним.

3. Управління доступом – метод захисту інформації, пов'язаний з регулюванням використання всіх ресурсів інформаційної системи (елементів баз даних, програмних і технічних засобів).

Управління доступом включає наступні функції захисту:

- ідентифікацію співробітників і ресурсів інформаційної системи;
- аутентифікацію (встановлення автентичності) об'єкту по пред'явленому їм ідентифікатору (імені). Як правило, до таких засобів відносяться паролі;
- перевірку повноважень – авторизація користувачів.

4. Регламентація – метод захисту інформації, що створює певні умови автоматизованої обробки, зберігання та передачі інформації, при яких можливість несанкціонованого доступу до неї (мережевих атак) зводилася б до мінімуму.

5. Примус – метод захисту, при якому користувачі системи змушені дотримуватися правил обробки, передачі і використання інформації (яка захищається) під загрозою матеріальної, адміністративної та кримінальної відповідальності.

6. Спонукання – метод захисту інформації, який мотивує користувачів не порушувати встановлені правила за рахунок дотримання сформованих моральних і етичних норм.

Всі перераховані методи інформаційної безпеки реалізуються за допомогою основних засобів захисту: фізичних, апаратних, програмних, апаратно-програмних, криптографічних, організаційних, законодавчих та морально-етичних.

Засоби забезпечення безпеки процесів переробки інформації, що використовуються для створення механізму захисту, поділяються на:

1. Формальні (виконують захисні функції по заздалегідь передбаченій процедурі без безпосередньої участі людини). До них входять:
   - фізичні засоби захисту, які призначені для зовнішньої охорони

території об'єктів і захисту компонентів інформаційної системи організації (механічні, електричні, електромеханічні, електронні, електронно-механічні пристрої та системи, які функціонують автономно).

- апаратні засоби захисту – це пристрої, вбудовані в блоки інформаційної системи (сервери, комп'ютери і т.д.) або сполучаються з нею спеціально для вирішення завдань захисту інформації. Вони призначені для внутрішнього захисту елементів обчислювальної техніки та засобів зв'язку.
- програмні засоби захисту, що призначені для виконання функцій захисту інформаційної системи за допомогою програмних засобів (антивірусний захист, міжмережеві екрани і т.д.).

2. Неформальні (визначаються цілеспрямованою діяльністю людини або регламентують цю діяльність). До них входять:
    - організаційні засоби – організаційно-технічні заходи, спеціально передбачаються в технології функціонування системи з метою вирішення завдань захисту інформації.
    - законодавчі засоби – нормативно-правові акти, за допомогою яких регламентуються права та обов'язки, а також встановлюється відповідальність всіх осіб і підрозділів, що мають відношення до функціонування системи, за порушення правил обробки інформації, наслідком чого може бути порушення її захищеності.
    - психологічні (морально-етичні) засоби – склалися в суспільстві або даному колективі моральні норми або етичні правила, дотримання яких сприяє захисту інформації, а порушення їх прирівнюється до недотримання правил поведінки в суспільстві або колективі.

Графічно така класифікація представлена на рис. 1.2.

Рис. 1.2. Методи та засоби захисту інформації

Серед усіх розглянутих методів, для впровадження у програмний модуль захисту даних, були обрані методи маскування, управління доступом та регламентації доступу. Такий вибір обумовлений тим, що для реалізування цих методів використовуються саме програмні засоби. До того ж, інші методи більш орієнтовані на фізичне перешкоджання крадіжки інформації та залежать від морально-етичних норм людей.

Хоча тут, необхідно зазначити, що методи управління доступом та регламентації доступу також були обрані, тому що є одними з найбільш дієвих та гнучких для налаштування у веб-додатках. Адже вони напряму взаємодіють з користувачами, визначаючи їхні ролі та права у системі та доступ до інформаційних ресурсів. Такі методи дають змогу забезпечити конфіденційність даних користувачів системи та розміщених матеріалів.

ВИСНОВКИ ДО РОЗДІЛУ 1

В розділі розглянуто теоретичні засади та критерії оцінки інформаційної безпеки, властивості та основні загрози безпеки інформації. Виділено такі аспекти захисту даних:

- захист від несанкціонованого ознайомлення з даними;
- захист даних від несанкціонованої модифікації;
- захист (забезпечення) доступу до даних, а також можливості їх використання.

Важливо зауважити, що доступ до даних забезпечується як підтриманням системи в робочому стані так і завдяки способам, які дозволяють швидко відновити втрачені чи пошкоджені дані.

Проаналізовано методи і засоби захисту інформації. Для впровадження у програмний модуль захисту було обрано методи:

- маскування;
- регламентації;
- управління доступом.

Вони є одними з найбільш дієвих та гнучких для налаштування у веб-додатках та напряму взаємодіють з користувачами, визначаючи їхні ролі і права у системі та доступ до інформаційних ресурсів.

РОЗДІЛ 2. ОГЛЯД ПРОБЛЕМАТИКИ, ОПИС ТА ВИБІР ВИКОРИСТАНИХ СИСТЕМ ТА ПРОГРАМНИХ ЗАСОБІВ

2.1 Обґрунтування проблеми

Визначення проблем, які вирішить розробка програмного модуля захисту даних навчально-методичного відділу КНТЕУ подано у табл. 2.1.

Таблиця 2.1

Обґрунтування проблеми захисту даних

Проблема	При переході від паперової технології зберігання, обробки та передавання інформації до електронної випливає відсутність безпечного та безперервного доступу до даних, наявність загрози несанкціонованого доступу та можливості впровадження атак з метою отримання або пошкодження даних.
Стосується	Процесів перегляду, обробки та зберігання даних навчально-методичного відділу КНТЕУ.
Наслідки	Можливість витоку, розголошення, модифікації, блокування та знищення даних.
Успішне вирішення	Надасть співробітникам можливість безпечного та безперервного доспуту до матеріалів навчально-методичного відділу КНТЕУ. Забезпечить цілісніть, доступність та конфіденційність даних, унеможливить впровадження поширених атак на ресурс.

2.2 Аналіз поширених атак на веб-ресурси в сфері освіти

Безпека веб-додатків знаходиться в першій десятці трендів і загроз інформаційній безпеці вже понад 10 років. Проте спеціалізованих засобів захисту веб-додатків досить мало, здебільшого це завдання покладають на розробників.

Проаналізувавши дані досліджень «Актуальні кіберзагрози 2018 року» та «Статистика атак на веб-додатки 2018-2019» компанії Positive Technologies можна виділити такі найпоширешіні атаки на веб-додатки: SQL Injection (27%), XSS (17%) та Path Traversal (14%). Сумарно вони складають більшу половину всіх виявлених кібератак на веб-ресурси компаній [К].

Будь-який сайт, доступний з інтернету, щодня сканується на наявність вразливостей або піддається різного роду атакам. Такі сканування не означають, що кожен сайт неодмінно намагаються зламати. Наприклад, зловмисник може просто перевіряти доступні адреси на наявність уразливих систем або відпрацьовувати якийсь новий скрипт на великому переліку адрес.

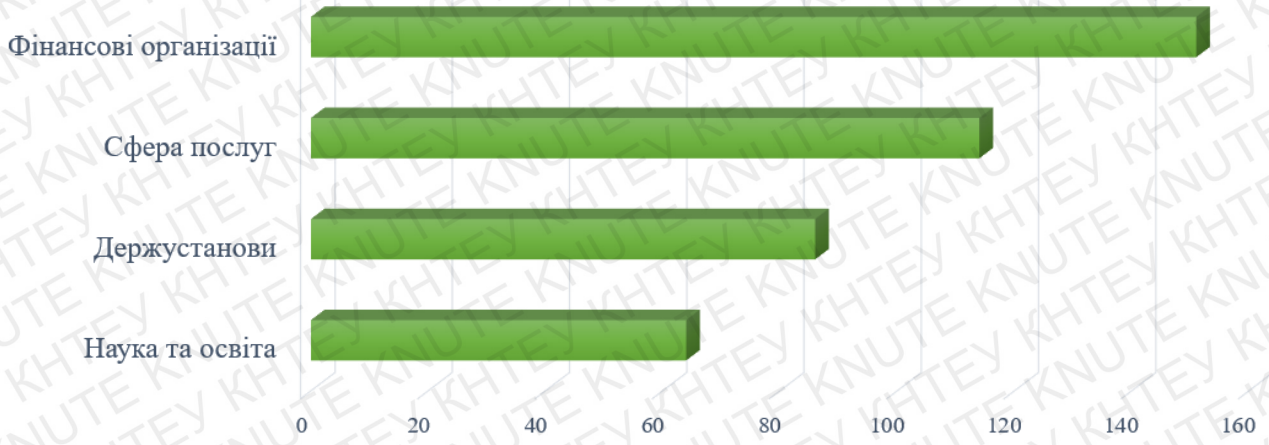

Рис. 2.1. Середня кількість атак в день на один веб-додаток

Статистичні дані, зображені на Рис. 2.1, показують, що у сфері науки та освіти зафіксовано найменшу кількість веб-атак, проте їх наявність вже є привідом для застосувань мір безпеки. Найпоширенішими в даній сфері є міжсайтове виконання сценаріїв (XSS) – 42%, впровадження SQL коду (SQL Injection) – 29%, вихід за межі каталогу (Path Traversal) – 15%, атаки, спрямовані на отримання інформації про веб-додаток (Information Leakage) та інші (Рис.2.2).

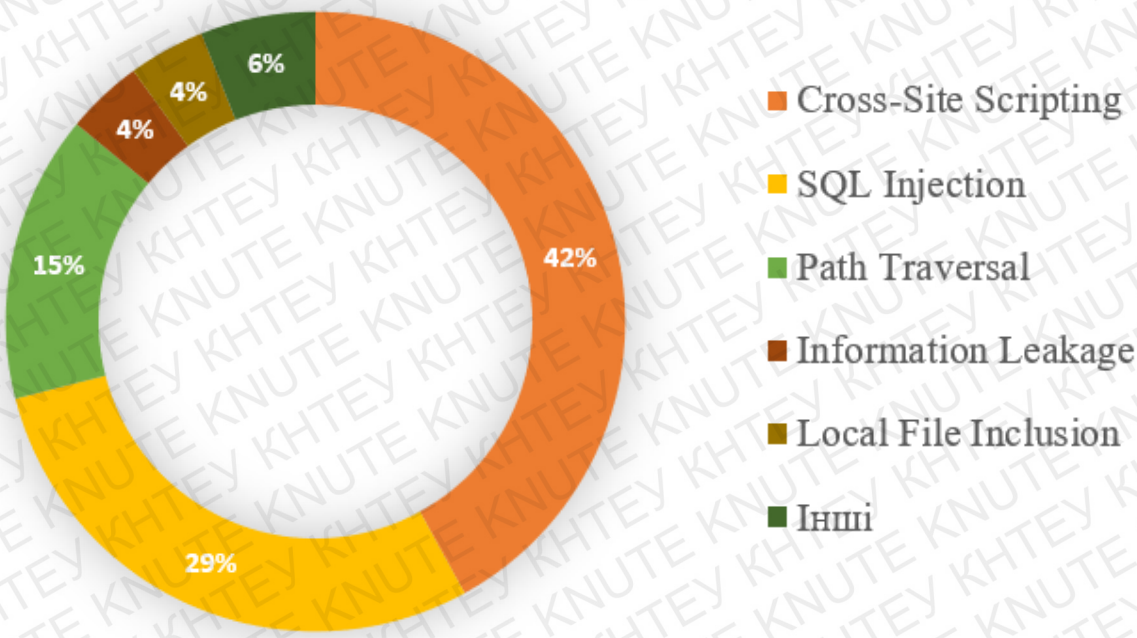

Рис 2.2. Топ поширених атак на веб-додатки організацій сфери освіти

При цьому, веб-додатки таких організацій схильні до небезпечних атак, спрямованих на отримання контролю над сервером і інформації з баз даних.

Отже, необхідно забезпечити захист від SQL ін'єкцій, міжсайтового виконання сценаріїв, правильно розмежувати права доступу до контенту та серверу, налаштувати міжмережевий екран.

2.3 Постановка задачі

Призначенням модулю є забезпечення цілісності, доступності та конфіденційності даних навчально-методичного відділу КНТЕУ. Для вирішення зазначених проблем буде проводитись власна розробка з урахуванням поширених вразливостей та атак на веб-ресурси в даному напрямку.

Розроблюваний модуль повинен мати такі функції:

- **вхідна інформація**:
  - користувач (відображене ім'я, електронна пошта, логін, пароль, статус персоналу, статус суперкористувача);
  - група користувачів (назва, учасники, права доступупу);
  - права доступу (назва, тип, напрямок);

- журнал логувань (тип контенту, редагуємий елемент, користувач);

- **методи захисту інформації:**
  - маскування;
  - управління доступом;
  - регламентація**;**
- **вихідна інформація:**
  - оброблені і збережені дані користувачів;
  - зашифровані паролі та інша чутлива інформація;
  - права доступу користувачів та груп користувачів до матеріалів;
  - журнал логувань для всіх можливих дій користувачів;
  - методичні матеріали, що зберігаються в БД.

Окрім цього, модуль повинен мати:

- модуль автентифікації користувачів з використання сесій, кук та шифрування;
- захист CSRF для всіх існуючих форм;
- проміжний шар для захисту від XSS атак та SQL-ін’єкцій.

Визначення позиції та мети створення модуля представлено в табл. 2.2.

Таблиця 2.2

Для	Навчально-методичного відділу КНТЕУ
Необхідно	Розробити програмний модуль захисту даних, що забезпечить цілісність, безперервну доступність та конфіденційність даних
Назва продукту	Програмний модуль захисту даних
Деталі	Використано: Python 3.7.3 як ядро системи, PostrgeSQL – СУБД користувачів та прав доступу. Графічний інтерфейс панелі адміністратора розроблено за допомогою технологій HTML5, CSS3 та мови програмування JavaScript

Продовження таблиці 2.2

Продукт	Усуває загрозу не санкціонованого доступу до матеріалів та унеможливлює впровадження поширених інтернет-атак. Таким чином дані цілісні, безперервно доступні та захищені.

2.4 Вибір програмних засобів для реалізації

Для ядра серверної частини програмного модуля захисту даних було обрано мову програмування Python 3.7.3, високорівневий відкритий Python-фреймворк Django 2.2.6 для розробки веб-системи, систему управління базами даних PostgreSQL та адаптер для бази даних Psycopg 2.

Для серверу обрано операційну систему Ubuntu Server 18.04, проксі-сервер Caddy та службу Supervisorctl, працюючою в парі з Gunicorn.

Для написання клієнтської частини панелі адміністрування обрано мову JavaScript. Графічний інтерфейс побудовано за допомогою HTML5 і CSS3.

2.4.1 Мова програмування Python

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, в тому числі структурну, об'єктно-орієнтовану, функціональну, імперативну і аспектно-орієнтовану. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних. Код в Python організовується у функції та класи, які можуть об'єднуватися в модулі [6].

Python давно себе зарекомендував, багато Гігантів інтернет індустрії використовують його для створення та підтримки своїх веб-додатків. Наприклад, компанія Google у багатьох своїх проектах, таких як Youtube, Google Search використовує Python. Також, на Python написані DropBox, Instagram, Reddit, Pinterest, Spotify та багато інших популярних веб-додатків.

В цілому про Python найкраще сказано в так званій філософії «Zen Python» [14].

Python підтримує «програмування в цілому», що робить його придатним для розробки великих систем. В табл. 2.3 представлено переваги Python над іншими мовами програмування.

Таблиця 2.3

Переваги Python над іншими мовами програмування

На відміну від	Python
Perl	Має більш чіткий синтаксис і більш просту архітектуру, що робить програмний код більш зручним для читання, простим в супроводі і знижує ймовірність появи помилок
Java	Простіше і зручніше, адже Python - це мова сценаріїв, а Java успадкувала складний синтаксис від таких мов програмування, як C та C++
C ++	Також простіше та зручніше, але нерідко він не може конкурувати з C ++, оскільки, будучи мовою сценаріїв, Python призначений для вирішення іншого кола завдань
Visual Basic	Більш потужний і більш стерпний. Відкрита природа Python також означає, що немає якоїсь окремої компанії, яка його контролює
PHP	Більш чіткий і більш універсальний. Іноді Python використовується для створення веб-сайтів, але він здатний вирішувати більш широке коло завдань, від управління роботами до створення анімаційних фільмів

Продовження таблиці 2.3

Ruby	Більш зрілий і має більш ясний синтаксис. На відміну від Ruby і Java, об'єктно-орієнтований стиль програмування є необов'язковим в Python – він не змушує використовувати ООП в проектах, де цей стиль непридатний
SmallTalk і Lisp	Володіє динамічними особливостями цих мов, але має більш простий і традиційний синтаксис, доступний як для розробників, так і для кінцевих користувачів

2.4.2 Django

Django – вільний фреймворк для веб-додатків на мові Python, що використовує шаблон проектування MVC та підтримується організацією Django Software Foundation [C].

MVC – це схема розділення даних додатку, користувацького інтерфейсу та управляючої логіки на три окремих компонента: модель, представлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно від інших [9].

Сайт на Django будується з одного або декількох додатків. Один з основних принципів фреймворка – DRY (принцип розробки програмного забезпечення, націлений на зниження повторень інформації, що формулюється так: «Будь-яка інформація повинна мати єдине, однозначне, авторитетне представлення в системі») [7].

Також, на відміну від інших фреймворків, обробники URL в Django конфігуруються явно за допомогою регулярних виразів, а не виводяться автоматично зі структури моделей контролерів [5].

Фреймворк Django має такі переваги:

- швидкість: Django був розроблений, щоб допомогти розробникам створити додаток настільки швидко, на скільки це можливо;

- повна комплектація: Django працює з десятками додаткових функцій, які помітно допомагають з аутентифікацією користувача, картами сайту, адмініструванням вмісту, RSS і багато чим іншим;
- безпека: Працюючи в Django, ви отримуєте захист від помилок, пов'язаних з безпекою, які ставлять під загрозу проект;
- масштабованість: фреймворк Django найкращим чином підходить для роботи з високими трафіками;
- різнобічність: менеджмент контенту, наукові обчислювальні платформи, навіть великі організації - з усім цим можна ефективно справлятися за допомогою Django [12].

2.4.3 Вибір системи управління базою даних

PostgreSQL (Додаток А) – вільна об’єктно-орієнтована система управління базами даних. Оскільки дана СУБД є об’єктно-орієнтованою, тому вона має певні переваги: наявність механізму успадкування, відповідність моделі даних в додатку і в базі даних, зручність в відображенні складних об’єктів, всі об'єкти на рівні джерела даних строго типізовані, велика кількість типів даних і можливість визначення нових типів даних [11].

2.4.4 Інтерфейс роботи з БД

Для роботи з базою даних використовується модуль Psycopg – адаптер бази даних PostgreSQL для мови програмування Python. В останній версії Psycopg був повністю переписаний для підтримки нового класу підключень і управління курсорами. Як і перша версія, Psycopg 2 був написаний з метою того, щоб бути малого розміру, швидкодіючим, і стабільним [11].

Перевагами адаптера Psycopg 2 є: орієнтованість на багатопочність з створення і знищенням великої кількості курсорів, підтримка асинхронних операцій, повна підтримка стандарту Python DBAPI-2.0, який орієнтований на уніфікацію доступу до різних СУБД, підтримка розширень DBAPI-2.0 для підвищення продуктивності потоків.

2.4.5 Мова програмування JavaScript

JavaScript – це особлива мова програмування, яка базується на об'єктному представленні браузера. Вона потрібна для того, щоб надати сайту більше інтерактивності в порівнянні зі звичайним статичним HTML-документом.

За допомогою JavaScript створюються динамічні документи HTML. JavaScript пов'язує воєдино всі будівельні блоки програми, це як би засіб побудови фундаменту, здійснює перевірку полів форм HTML до того, як вони передалися на сервер.

Управління програмою на даній мові програмування йде через локальне введення інформації. Користувач має можливість бачити в окремих вікнах повідомлення-застереження, які виводяться за допомогою JavaScript [13].

2.4.6 Мова розмітки HTML5

HTML5 – це інструмент для упорядкування Web-контенту. Він призначений для спрощення Web-проектування і Web-розробки за рахунок мови розмітки, що забезпечує стандартизований і інтуїтивно зрозумілий інтерфейс.

HTML5 пропонує нові теги і вдосконалення, в числі яких елегантна структура, органи управління формами, API-інтерфейси, мультимедійні функції, підтримка баз даних, істотно збільшена швидкість обробки [10].

2.4.7 Таблиця каскадних стилів CSS3

CSS – це мова, яка використовується як засіб оформлення веб-сторінок, а саме для роботи з шрифтами, кольорами, полями, таблицями, картинками, розташуванням елементів.

Основною метою, яка послужила для розробки мови CSS, було розділення розмітки вмісту, написаного на HTML, та подання документа, написаного на CSS. Результат цього поділу допомагає збільшити доступність документа, надати велику гнучкість, а також зменшити складність і повторюваність в структурному вмісті, створює зручне опис дизайну та стилістики web-сторінки і її вмісту. CSS дає можливість надати один документ в різних стилях або методах виведення [10].

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі проаналізовано поширені атаки на веб-ресурси в сфері освіти, обґрунтовано проблематику, а саме: при переході від паперової технології зберігання, обробки та передавання інформації до електронної випливає відсутність безпечного та безперервного доступу до даних, наявність загрози несанкціонованого доступу та можливості впровадження атак з метою отримання або пошкодження даних.

Сформовано постановку задачі, яка полягає в створенні модуля, що надасть співробітникам можливість безпечного та безперервного доспуту до матеріалів навчально-методичного відділу КНТЕУ, оброблятиме вхідну та вихідну інформацію.

Призначенням модулю є забезпечення цілісності, доступності та конфіденційності даних навчально-методичного відділу КНТЕУ. Для вирішення зазначених проблем проведена власна розробка з урахуванням поширених вразливостей та атак на веб-ресурси в сфері освіти і науки.

Для створення модуля було обрано наступні програмні засоби: для серверної частини – мова програмування Python 3.7.3, фреймворк для веб-додатків Django 2.2.6, система управління базою даних PostgreSQL та адаптер бази даних Psycopg 2; для користувацької частини – мова програмування JavaScript, мова розмітки HTML5 та таблиця каскадних стилів CSS3.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ЗАХИСТУ ДАНИХ

3.1 Проектування і структурно-функціональний аналіз

Для вирішення поставленої задачі та реалізації обраних методів захисту інформації дані про користувачів, права доступу та історію редагування необхідно зберігати в окремій базі даних.

Під час концептуального проектування БД користувачів було побудовано ER-модель та описано основні сутності і зв’язки.

Модель «сутність-зв'язок» (ER-модель) – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель – це мета-модель даних, тобто засіб опису моделей даних. Головними поняттями ER-моделі є сутність, зв'язок і атрибут. У діаграмах ER-моделі сутність подається у вигляді прямокутника, що містить ім'я сутності. При цьому ім'я сутності – це ім'я типу, а не деякого конкретного екземпляра цього типу.

Зв'язок – це асоціація, що графічно зображується та установлюється між двома сутностями. Ця асоціація завжди є бінарною і може існувати між двома різними сутностями або між сутністю і нею ж самою (рекурсивний зв'язок).

При побудові даної моделі, були виділені такі ключові сутності (Рис. 3.1):

- користувач (User) – клас наслідується від AbstractBaseUser та PermissionMixin і описує інформацію про користувачів системи. Основні атрибути: first_name, last_name, email, username, password;
- група (Group) – групи користувачів з певними правами доступу;
- права доступу (Permission) – описує права доступу до контента для корустувача або групи користувачів. Основні атрибути: content_type, codename;

- логування (LogEntry) – описує дії користувачів в системі. Основні атрибути: content_type (FK), user (FK), action_flag, action_time, change_message, object_id, object_repr;
- тип контенту (ContentType) – описує назву додатку та моделі для якої будуть надаватися права доступу та записуватися логи. Основні атрибути: app_label, model.

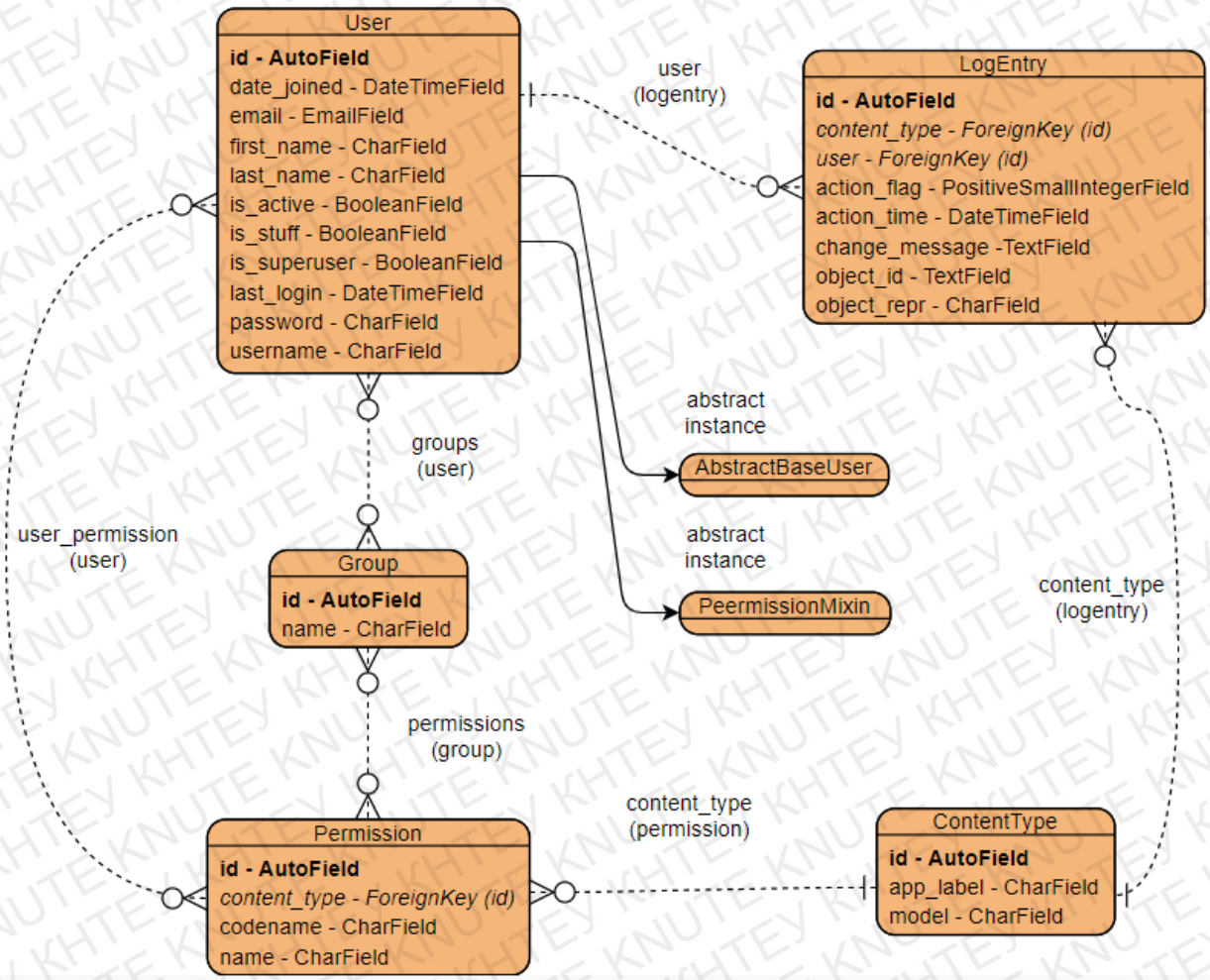

Рис. 3.1. ER-модель для модуля автентифікації

Для моделювання процесу виконання операцій використана діаграма діяльності, яка зображується графом, вершинами якого є стани (дій і/або видів діяльності), а дугами – переходи від одного стану (дій/виду діяльності) до іншого стану (дій/виду діяльності).

Головним напрямом використання діаграм діяльності є візуалізація

особливостей реалізації операцій класів, коли необхідно зобразити алгоритми їхнього виконання. При цьому кожен стан дії може бути виконанням операції деякого класу, а вид діяльності – послідовністю таких дій [4].

На рисунку 3.2 зображено діаграму діяльності для процесу «Авторизації користувача» з перевіркою валідності логіна та пароля (автентифікації), прав доступу та наявніть CSRF токену для захисту від міжсайтових запитів.

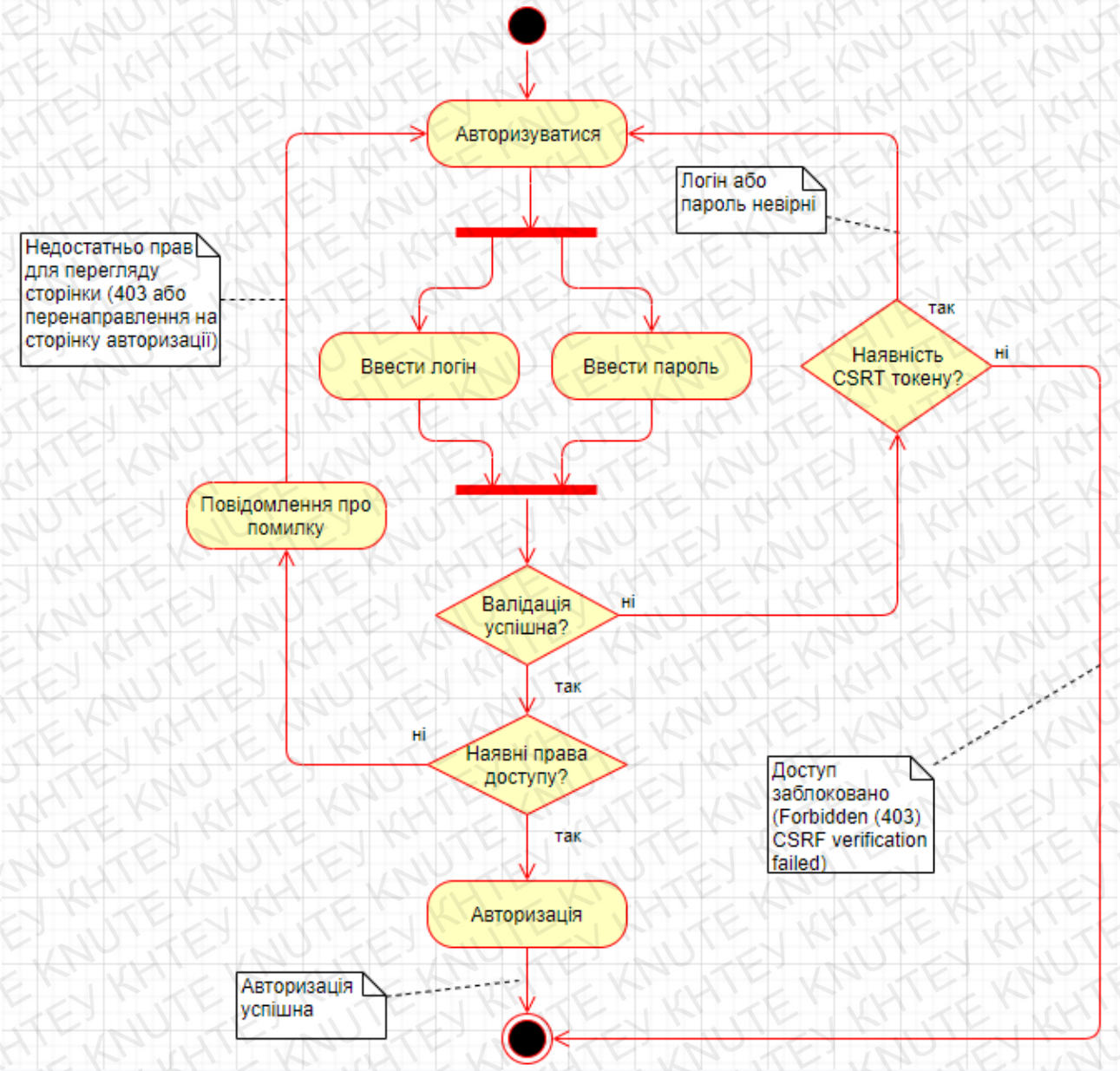

Рис. 3.2. Діаграма діяльності для процесу авторизації користувача

3.2 Розробка програмної частини

3.2.1 Модуль автентифікації користувачів

Для функціонування модулю автентицікації користувачів було описано класи моделей користувачів (User), груп (Group), прав доступу (Permission), журналу логування (LogEntry), що описані у файлі models.py (Додаток Б) та класи-обробники для моделей, що описані у файлі views.py (Додаток В).

Для шифрування паролів та іншої чутливої інформації використано алгоритм PBKDF2 с хешем SHA256.

У файл конфігурації (Додаток Г) підключено проміжні шари SessionMiddleware для зберігання сесій користувачів, AuthenticationMiddleware для системи автентифікаці (Рис.3.3).

```
MIDDLEWARE = [
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',

    # Custom middleware
    'diploma.apps.core.middleware.SecurityMiddleware',
    'diploma.apps.core.middleware.CsrfViewMiddleware',
    'diploma.apps.core.middleware.XFrameOptionsMiddleware',
]
```

Рис. 3.3. Підключення проміжних шарів у файл конфігурації settings.py.

3.2.2 Модуль захисту від XSS та CSRF

Для праввильного функціонування модуля захисту від XSS атак заголовки HTTP-запиту треба перевіряти до класу представлення, тому для розробки було вирішено використовувати проміжні шари (middleware). Було описано класс SecureMiddleware, що обробляє HTTP-заголовки і дозволяє примусово перенаправляти запит з HTTP та HTTPS або взагалі заборонити використання не захищеного протоколу, забороняти XSS атаки (Рис. 3.4)

Для захисту від X-Frame відображення та склікування було розроблено проміжний шар XFrameoptionMiddleware, який перевіряє HTTP-заголовки, блокуючи запит, якщо виявлена наявність “X-Frame-Option” (Рис. 3.5)

```python
class SecurityMiddleware(MiddlewareMixin):
    def __init__(self, get_response=None):...

    def process_request(self, request):
        path = request.path.lstrip("/")
        if (self.redirect and not request.is_secure() and
                not any(pattern.search(path) for pattern in self.redirect_exempt)):
            host = self.redirect_host or request.get_host()
            return HttpResponsePermanentRedirect("https://%s%s" % (host, request.get_full_path()))

    def process_response(self, request, response):
        if (self.sts_seconds and request.is_secure() and
                'Strict-Transport-Security' not in response):
            sts_header = "max-age=%s" % self.sts_seconds
            if self.sts_include_subdomains:
                sts_header = sts_header + "; includeSubDomains"
            if self.sts_preload:
                sts_header = sts_header + "; preload"
            response['Strict-Transport-Security'] = sts_header
        if self.content_type_nosniff:
            response.setdefault('X-Content-Type-Options', 'nosniff')
        if self.xss_filter:
            response.setdefault('X-XSS-Protection', '1; mode=block')
        return response
```

Рис. 3.4. Лістинг коду проміжного шару SecurityMiddleware

```python
from django.utils.deprecation import MiddlewareMixin


class XFrameOptionsMiddleware(MiddlewareMixin):

    def process_response(self, request, response):

        if response.get('X-Frame-Options') is not None:
            return response

        if getattr(response, 'xframe_options_exempt', False):
            return response

        response['X-Frame-Options'] = self.get_xframe_options_value(request, response)
        return response

    def get_xframe_options_value(self, request, response):
        return getattr(settings, 'X_FRAME_OPTIONS', 'SAMEORIGIN').upper()
```

Рис. 3.5. Лістинг коду проміжного шару XFrameoptionMiddleware

Для захисту даних від перехвату під час відправки форм було розроблено проміжний шар CsrfViewMiddleware. Таким чином, на клієнській частині у формі генерується csrf_token, який після відправки порівнюється зі згенерованим на сервері токеном і якщо вони не співпадають, то запит блокується.

3.3 Впровадження програмного забезпечення

3.3.1 Вимоги до апаратного та програмного забезпечення

Для користування системою користувачеві необхідно мати персональний комп'ютер, планшет або мобільний пристрій з браузером і доступом до мережі Інтернет. Єдиною вимогою до апаратного і програмного забезпечення клієнтських пристроїв є підтримка оновлених та стабільних версій будь-яких браузерів.

Сервер має відповідати мінімальним вимогам:

- процесор: "Xeon E5-XXXXv2" 1 ядро;
- RAM: "DDR3 ECC Reg" 1 ГБ;
- жорсткий диск: "SAS RAID 10" 16 ГБ;
- порт: 25 Мбіт/с;
- IP-адреса: статичний IPv4/IPv6;
- операційна система: Windows / Ubuntu / Debian / Fedora / FreeBSD.

3.3.2 Інсталяційний пакет

Інсталяційний пакет програмного забезпечення складається з:

- інтерпретатор: Python 3.6;
- менеджер пакетів: PIP;
- додаткові пакети Python записані у файлі requirements.conf: astroid==1.6.2, colorama==0.3.9, Django==2.2.1, isort==4.3.4, lazy-object-proxy==1.3.1, mccabe==0.6.1, psycopg2==2.7.4, pylint==1.8.3, pytz==2018.3, six==1.11.0, style==1.1.0, update==0.0.1, wrapt==1.10.11, django-autoslug=1.7.4.

3.3.3 Діаграма розміщення модуля захисту даних

Діаграма розміщення показує фізичне розташування мережі і місцезнаходження в ній різних компонентів. На діаграмі зображені компоненти програмного забезпечення і зв'язки між ними. При цьому виділяють два типи компонентів: виконувані компоненти і бібліотеки коду. Представлена на рис. 3.6.

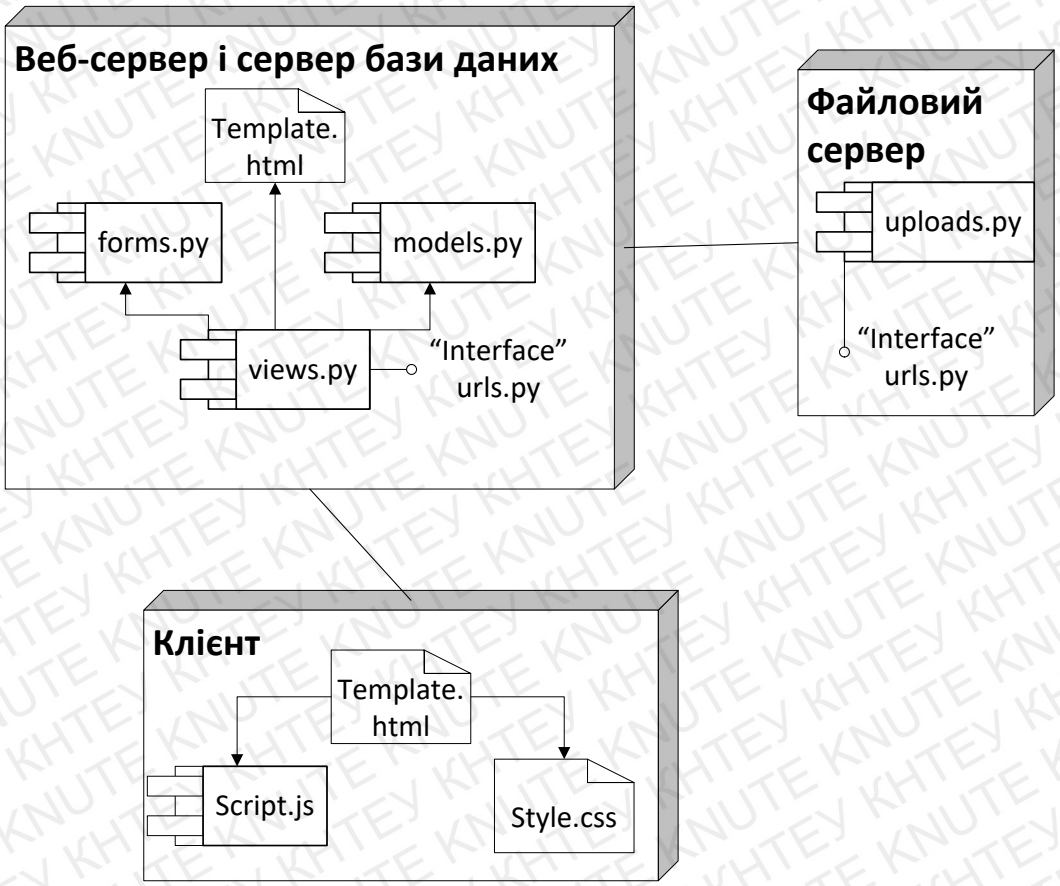

Рис. 3.6. Діаграма розміщення для програмного модуля

3.3.4 Архітектура програмного забезпечення

Організаційна структура інформаційної системи представлена у вигляді діаграми пакетів – структурна діаграма, основним змістом якої є пакети і відносини між ними. На рисунку 3.7 зображено діаграму пакетів розроблюваної системи.

За структурою MVC програмний модуль захисту даних в свою чергу має такі модулі:

- models.py (Додаток Б) – описані класи моделей, які записані в таблицю БД;
- views.py (Додаток В) – описані класи представлення моделей, що передаються до URL-адрес;
- forms.py – описані класи форм, за допомогою яких відбувається валідація та обробка даних зі сторони користувача;

- urls.py – опис URL-адрес, що описують роутінг запитів на класи представлень;
- settings.py (Додаток Г) – модуль конфігурацій додатку, підключення інстальованих пакетів та дані, щодо підключення проекту до БД.

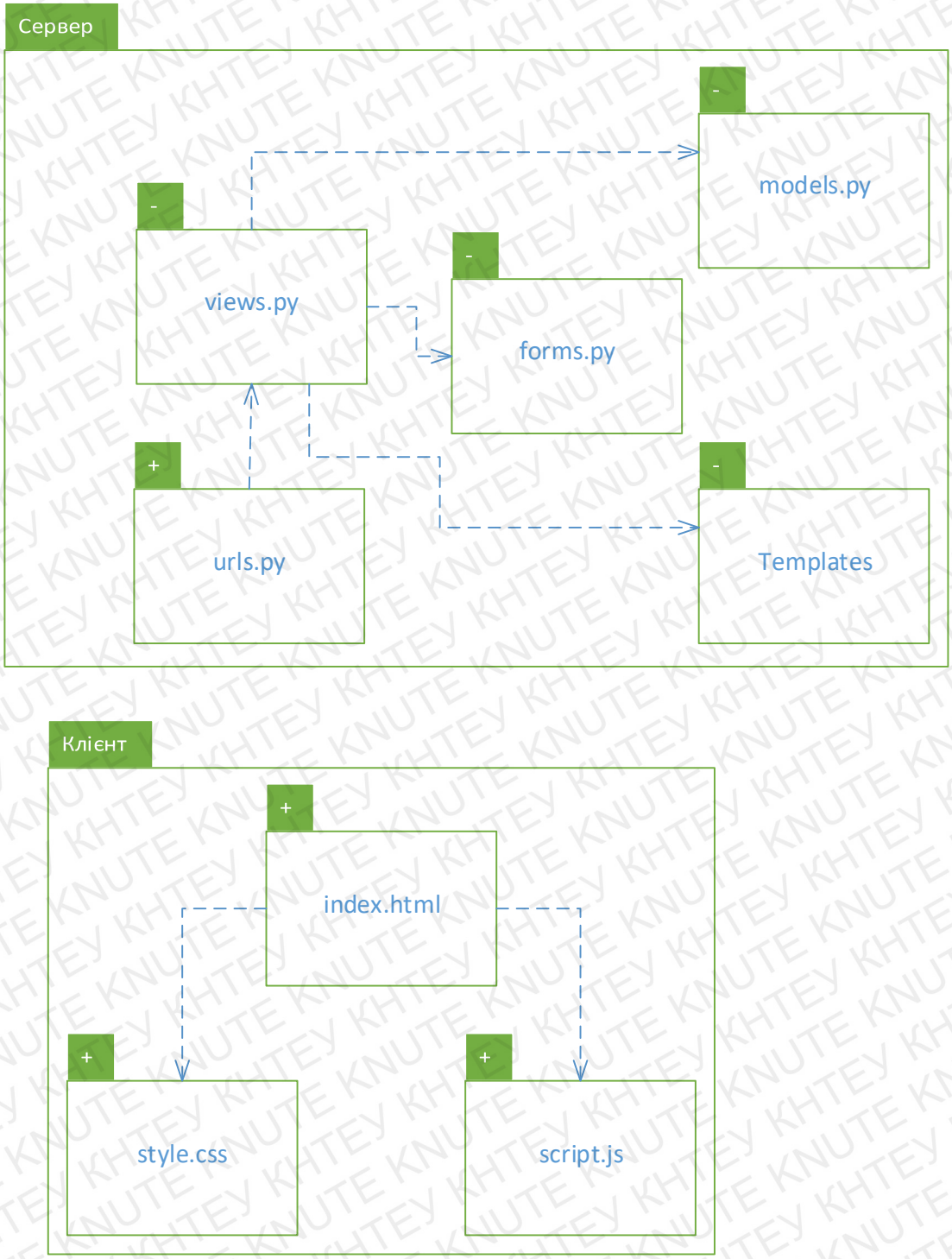

Рис. 3.7 Діаграма пакетів

3.3.5 Організаційна структура

Програмний модуль захисту даних був реалізований як веб-додаток – розподілений застосунок, а якому клієнтом виступає браузер, а сервером – веб-сервер. Браузер являється тонким клієнтом – логіка додатку, модулі безпеки зосереджуються на сервері, а функція браузера полягає у відображенні інформації, завантаженої мережею з сервера, і передачі даних користувача.

3.4 Функціональне тестування

Проведено тестування SQL ін'єкціями, що базуються на розриванні змінної та створення додаткового запиту, виводу даних, повне видалення бази даних.

Створення запиту на видалення бази даних через форму авторизації.

Таким запитом є DROP DATABASE. Засоби python/django не дозволяють виконати 2 запити до БД, тобто, необхідно використати команду UNION, яка об'єднає існуючий запит з SQL-ін'єкцією в формі логіну.

Запит після проходження валідації був відхилений. Видалення бази даних не відбулося. Отже, тестування SQL-ін'єкціями пройдено.

Для тестування на фільтрацію від XSS атак використовують скрипт виведення модального вікна з текстом чи посиланням. Наприклад "><script>alert(«Тестування»)</script> – скрипт закриває попередній елемент та виводить модальне вікно з текстом «Тестування». Для того, щоб перевірити систему на фільтрацію від атак необхідно вставити скрипт в поле вводу та відправити запит. Якщо фільтрація відсутня, або не допрацьована то скрипт буде виконано і відбудеться вивід модального вікна. Якщо фільтрація реалізована правильно то дані будуть екрановані і при перегляді html-коду значення поля буде також екранованим.

Отже, дані, введені в форму, екрануються та не мають можливості вплинути на хід виконання функцій системи.

Для початку користування системою, користувачу необхідно пройти авторизацію (Рис 3.8).

Навчально-методичний
відділ КНТЕУ

Логін

Пароль

Увійти

КНТЕУ © 2019

Рис 3.8. Сторінка авторизації користувача

Для тестування захисту форми авторизації від міжсайтової підробки запитів (XSS/CSRF атак) було використано Postman. Якщо POST запитом передати дані користувача на адресу авторизації отримаємо помилку доступу 403 (Рис. 3.9). Тобто захист від міжсайтової підробки запитів працює.

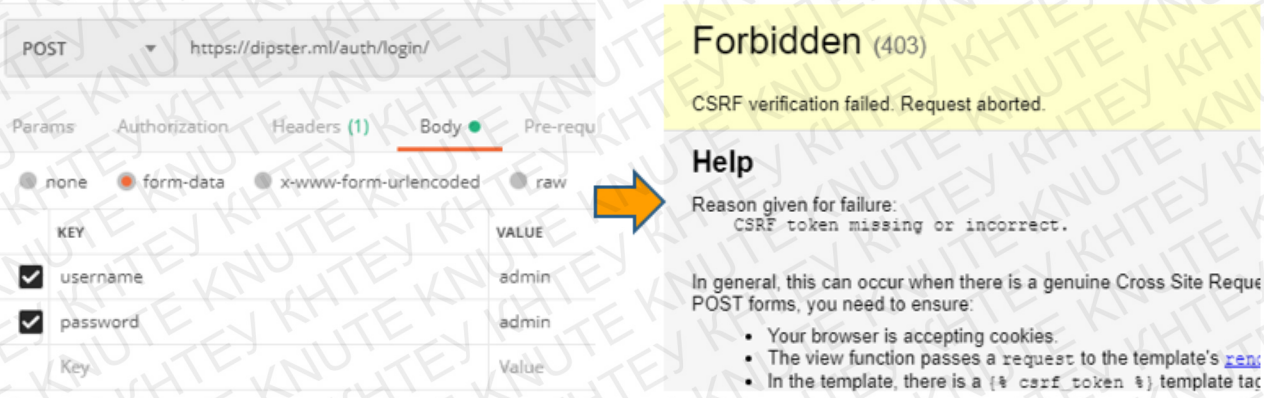

Рис. 3.9. Тестування модуля від CSRF атак.

Для тестування захисту від впровадження SQL можна передати строку ‘UNION DROP DATABASE’ яка видалить всю базу даних. В результаті запит був відхилений при проходженні валідації.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було проведено проектування і структурно-функціональний аналіз. Для побудови логічної моделі даних було використано засіб VisualParadigm Diagrams, за допомогою якого створено ER-діаграму, що описує ключові сутності. Побудовано діаграму діяльності для процесу «Авторизації користувача» для кращого розуміння алгоритму.

Описано розробку програмної частини модуля, а саме класів моделей користувачів, груп користувачів, прав доступу, представлень для модуля авторизації користувачів, класів проміжних шарів для реалізації захисту від поширених атак.

Перед впровадженням ПЗ виведено вимоги до апаратного і програмного забезпечення користувача та сервера. Модуль складається з веб-сервера і сервера бази даних, файлового сервера і клієнтської частини. Для реалізації продукту використано інтерпретатор Python 3.7.3, менеджер пакетів PIP та додаткові пакети Python.

Програмна частина поділена на модулі за структурою MVC: моделі визначені в models.py, представлення в views.py, контролер у urls.py. Модуль налаштувань, підключення інстальованих пакетів та дані, щодо підключення проекту до БД винесені в settings.py.

Для переконання в працездатності моделі було проведене успішне функціональне тестування SQL ін’єкціями, CSRF та XSS атаками.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Данаий дипломний проект на тему: «розробка програмного модуля захисту даних навчально-методичного відділу КНТЕУ» складається з трьох розділів.

В першому розділі розглянуто теоретичні засади та критерії оцінки інформаційної безпеки, властивості та основні загрози безпеки інформації. Виділено такі аспекти захисту даних:

- захист від несанкціонованого ознайомлення з даними;
- захист даних від несанкціонованої модифікації;
- захист (забезпечення) доступу до даних, а також можливості їх використання.

Проаналізовано методи і засоби захисту інформації. Для впровадження у програмний модуль захисту було обрано методи:

- маскування;
- регламентації;
- управління доступом.

Вони є одними з найбільш дієвих та гнучких для налаштування у веб-додатках та напряму взаємодіють з користувачами, визначаючи їхні ролі і права у системі та доступ до інформаційних ресурсів.

В другому розділі проаналізовано поширені атаки на веб-ресурси в сфері освіти, обґрунтовано проблематику, а саме: при переході від паперової технології зберігання, обробки та передавання інформації до електронної випливає відсутність безпечного та безперервного доступу до даних, наявність загрози несанкціонованого доступу та можливості впровадження атак з метою отримання або пошкодження даних.

Сформовано постановку задачі, яка полягає в створенні модуля, що надасть співробітникам можливість безпечного та безперервного доспуту до матеріалів навчально-методичного відділу КНТЕУ, оброблятиме вхідну та вихідну інформацію.

Призначенням модулю є забезпечення цілісності, доступності та конфіденційності даних навчально-методичного відділу КНТЕУ. Для вирішення зазначених проблем проведена власна розробка з урахуванням поширених вразливостей та атак на веб-ресурси в сфері освіти і науки.

Для створення модуля було обрано наступні програмні засоби: для серверної частини – мова програмування Python 3.7.3, фреймворк для веб-додатків Django 2.2.6, система управління базою даних PostgreSQL та адаптер бази даних Psycopg 2; для користувацької частини – мова програмування JavaScript, мова розмітки HTML5 та таблиця каскадних стилів CSS3.

В третбому розділі було проведено проектування і структурно-функціональний аналіз. Для побудови логічної моделі даних було використано засіб VisualParadigm Diagrams, за допомогою якого створено ER-діаграму, що описує ключові сутності. Побудовано діаграму діяльності для процесу «Авторизації користувача» для кращого розуміння алгоритму.

Описано розробку програмної частини модуля, а саме класів моделей користувачів, груп користувачів, прав доступу, представлень для модуля авторизації користувачів, класів проміжних шарів для реалізації захисту від поширених атак.

Перед впровадженням ПЗ виведено вимоги до апаратного і програмного забезпечення користувача та сервера. Модуль складається з веб-сервера і сервера бази даних, файлового сервера і клієнтської частини. Для реалізації продукту використано інтерпретатор Python 3.7.3, менеджер пакетів PIP та додаткові пакети Python.

Програмна частина поділена на модулі за структурою MVC: моделі визначені в models.py, представлення в views.py, контролер у urls.py. Модуль

налаштувань, підключення інстальованих пакетів та дані, щодо підключення проекту до БД винесені в settings.py.

Для переконання в працездатності моделі було проведене успішне функціональне тестування SQL ін’єкціями, CSRF та XSS атаками.

Розроблений програмний модуль захисту даних призначений для усунення загроз несанкціонованого доступу до матеріалів навчально-методичного відділу КНТЕУ та для унеможливлювання впровадження поширених інтернет-атак. Надає співробітникам можливість безпечного та безперервного доспуту до матеріалів навчально-методичного відділу КНТЕУ. Забезпечує цілісніть, доступність та конфіденційність даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про Основні засади розвитку інформаційного суспільства в Україні на 2007—2015 роки» від 09.01.2007 № 537-V

2. Аудит інформаційної безпеки: підручник / В. А. Ромака, А. Е. Лагун, Ю. Р. Гарасим та ін.; НАН України, Ін-т приклад. проблем механіки і математики ім. Я. С. Підстригача. — Львів: Сполом, 2015. — 363 с. — ISBN 978-966-919-123-6

3. Системи управління базами даних та знань: книга 2 / 8. А. Ю. Берко, О. М. Верес, В. В. Пасічник. – К.: Львів. «Магнолія-2006», 2012. – 584 с

4. Automatic Conversion from UML to CPN for Software Performance Evaluation. / Zhu L.-Z., Kong F.-Sh. / USA, Procedia Engineering. 2012

5. Django. The definitive guide to / Jacob Kaplan-Moss / USA, 2010. ISBN-13: 978-1-59059-725-5 - 560 p.

6. Django: Web Development with Python 1st Edition / Samuel Dauzon. – Birmingham, United Kingdom: Packt Publishing, 2016. – 730p

7. Mazes for Programmers: Code Your Own Twisty Little Passages / Jamis Buck. – United States: Pragmatic Bookshelf, 2015. – 286p.

8. Prediction of SADT – requirements for simple and complex correlations / Annett Knorr / Bundesanstalt (BAM), Berlin, 2016

9. Pro ASP.NET MVC 4, 4th edition with examples / Adam Freeman. — US: Springer, 2014. — 688 с.

10. Responsive Web Design with HTML5 and CSS3 / B. Frain. – Ed. 2. – Packt Publishing, 2015. – 312 p

11. Proceedings of the VLDB Endowment // Volume 5 Issue 12, August 2012 - ISSN: 2150-8097 2367502.2367523 p. 1850-1861

12. https://python-scripts.com/django-obzor - Співтовариство програмістів – веб-програмування [Електронний ресурс] – Режим доступу: 25.11.2019

13. http://www.vitaliypodoba.com/2015/06/webdev-howto - Побода В. Веб-розробка [Електронний ресурс] – Блог Віталія Пободи – Режим доступу: 25.11.2019

14. https://www.python.org/dev/peps/pep-0020 - Python Software Foundation [US] [Електронний ресурс] – Режим доступу: 25.11.2019

15. https://www.ptsecurity.com/ru-ru/research/analytics/web-application-vulnerabilities-statistics-2019 - Статистика уязвимостей веб-приложений 2018-2019 [Електронний ресурс] – Режим доступу: 25.11.2019

16. Ramon Nastase. Computer Networking – LLC, 2017y. - 82p.

17. Andrew Tanenbaum, David Wetherall. Upper Saddle River. Computer Networks, 5th Edition – New Jersy, 2014 – 203 p.

18. Martin Kleppmann. Designing Data-Intensive Applications – O'Reilly, 2017 – 559 p.

19. Nan Nguyen. Essential Cyber Security Handbook In Ukrainian – Nam H Nguyen ,2018 – 382p.

20. Richard E. Smith. Elementary Information Security. /Jones & Bartlett Publishers, Burlingston, 2015. - 890 p.

21. Peter W. Windows 10 For Senior For Dummies – Wiley, 2015 – 336 p.

22. INTERNATIONAL STANDARD - ISO/IEC 15408-3, third edition 2008-08-15

23. ISO/IEC 27032, Information technology — Security techniques — Guidelines for cybersecuritv — 2012 — 50 p.

ДОДАТОК А

Порівняння систем управління базами даних

	MS SQL Server	**PostgreSQL**	**MySQL**	**Oracle**
Ліцензія	Пропріетарна	PostgreSQL(Open Source license)	Пропріетарна/ GPL v2.0	Пропріетарна
Архітектура	Реляційна	Об’єктно-реляційна	Реляційна	Об’єктно-реляційна
Операційна система	Windows	Windows, Mac OS x, Linux, Unix, BSD	Windows, Mac OS x, Linux, Unix, BSD, Symbian	Windows, Mac OS x, Linux, Unix, BSD, z/OS
ACID	+	+	+	+
Збережені процедури і тригери	+	+	+	-
Створення резервних копій	+	+	+	+
Індекси	Expression, Partial, Full-text, Hash, Spatial	Bitmap, Expression, GIN, GiST, Partial, R-/R+ Tree, Reverse, Full-text	R-/R+ Tree, Full-text, Hash, Spatial	Bitmap, Expression, Partial, R-/R+ Tree, Reverse, Full-text, Hash, Spatial
Паралельні запити	+	+	-	+

ДОДАТОК Б

Лістинг класів моделей models.py, записаних у БД

```
from django.contrib import auth

from django.contrib.auth.base_user import AbstractBaseUser, BaseUserManager
from django.contrib.contenttypes.models import ContentType
from django.core.exceptions import PermissionDenied
from django.core.mail import send_mail
from django.db import models
from django.db.models.manager import EmptyManager
from django.utils import timezone
from django.utils.translation import gettext_lazy as _


class PermissionManager(models.Manager):
    use_in_migrations = True

    def get_by_natural_key(self, codename, app_label, model):
        return self.get(
            codename=codename,
            content_type=ContentType.objects.db_manager(self.db).get_by_natur
al_key(app_label, model),
        )


class Permission(models.Model):

    name = models.CharField(_('name'), max_length=255)
    content_type = models.ForeignKey(
        ContentType,
        models.CASCADE,
        verbose_name=_('content type'),
    )
    codename = models.CharField(_('codename'), max_length=100)

    objects = PermissionManager()

    class Meta:
        verbose_name = _('permission')
        verbose_name_plural = _('permissions')
        unique_together = (('content_type', 'codename'),)
        ordering = ('content_type__app_label', 'content_type__model',
                    'codename')
        def __str__(self):

            return "%s | %s | %s" % (
                    self.content_type.app_label,
                    self.content_type,
                    self.name,
                )
```

```
        def natural_key(self):
            return (self.codename,) + self.content_type.natural_key()
        natural_key.dependencies = ['contenttypes.contenttype']


    class GroupManager(models.Manager):
        """
        The manager for the auth's Group model.
        """
        use_in_migrations = True

        def get_by_natural_key(self, name):
            return self.get(name=name)


    class Group(models.Model):
        name = models.CharField(_('name'), max_length=150, unique=True)
        permissions = models.ManyToManyField(
            Permission,
            verbose_name=_('permissions'),
            blank=True,
        )

        objects = GroupManager()

        class Meta:
            verbose_name = _('group')
            verbose_name_plural = _('groups')

        def __str__(self):
            return self.name

        def natural_key(self):
            return (self.name,)

    class UserManager(BaseUserManager):
        use_in_migrations = True

        def _create_user(self, username, email, password, **extra_fields):
            if not username:
                raise ValueError('The given username must be set')
            email = self.normalize_email(email)
            username = self.model.normalize_username(username)
            user = self.model(username=username, email=email, **extra_fields)
            user.set_password(password)
            user.save(using=self._db)
            return user

        def create_user(self, username, email=None, password=None, **extra_fields)
    :
            extra_fields.setdefault('is_staff', False)
            extra_fields.setdefault('is_superuser', False)
            return self._create_user(username, email, password, **extra_fields)
```

```
    def create_superuser(self, username, email, password, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)

        if extra_fields.get('is_staff') is not True:
            raise ValueError('Superuser must have is_staff=True.')
        if extra_fields.get('is_superuser') is not True:
            raise ValueError('Superuser must have is_superuser=True.')

        return self._create_user(username, email, password, **extra_fields)


def _user_get_all_permissions(user, obj):
    permissions = set()
    for backend in auth.get_backends():
        if hasattr(backend, "get_all_permissions"):
            permissions.update(backend.get_all_permissions(user, obj))
    return permissions


def _user_has_perm(user, perm, obj):
    for backend in auth.get_backends():
        if not hasattr(backend, 'has_perm'):
            continue
        try:
            if backend.has_perm(user, perm, obj):
                return True
        except PermissionDenied:
            return False
    return False

def _user_has_module_perms(user, app_label):
    for backend in auth.get_backends():
        if not hasattr(backend, 'has_module_perms'):
            continue
        try:
            if backend.has_module_perms(user, app_label):
                return True
        except PermissionDenied:
            return False
    return False
```

```
class PermissionsMixin(models.Model):
    is_superuser = models.BooleanField(
        _('superuser status'),
        default=False,
        help_text=_(
            'Designates that this user has all permissions without '
            'explicitly assigning them.'
        ),
    )
    groups = models.ManyToManyField(
        Group,
        verbose_name=_('groups'),
        blank=True,
        help_text=_(
            'The groups this user belongs to. A user will get all permissions
'
            'granted to each of their groups.'
        ),
        related_name="user_set",
        related_query_name="user",
    )
    user_permissions = models.ManyToManyField(
        Permission,
        verbose_name=_('user permissions'),
        blank=True,
        help_text=_('Specific permissions for this user.'),
        related_name="user_set",
        related_query_name="user",
    )

    class Meta:
        abstract = True

    def get_group_permissions(self, obj=None):
        permissions = set()
        for backend in auth.get_backends():
            if hasattr(backend, "get_group_permissions"):
                permissions.update(backend.get_group_permissions(self, obj))
        return permissions

    def get_all_permissions(self, obj=None):
        return _user_get_all_permissions(self, obj)

    def has_perm(self, perm, obj=None):
        if self.is_active and self.is_superuser:
            return True
        return _user_has_perm(self, perm, obj)

    def has_perms(self, perm_list, obj=None):
        return all(self.has_perm(perm, obj) for perm in perm_list)
```

```
    def has_module_perms(self, app_label):
        if self.is_active and self.is_superuser:
            return True

        return _user_has_module_perms(self, app_label)


class User(AbstractBaseUser, PermissionsMixin):
    username_validator = UnicodeUsernameValidator()

    username = models.CharField(
        _('username'),
        max_length=150,
        unique=True,
        help_text=_('Required. 150 characters or fewer. Letters, digits and @/
./+/-/_ only.'),
        validators=[username_validator],
        error_messages={
            'unique': _("A user with that username already exists."),
        },
    )
    first_name = models.CharField(_('first name'), max_length=30, blank=True)
    last_name = models.CharField(_('last name'), max_length=150, blank=True)
    email = models.EmailField(_('email address'), blank=True)
    is_staff = models.BooleanField(
        _('staff status'),
        default=False,
        help_text=_('Designates whether the user can log into this admin site.
'),
    )

    is_active = models.BooleanField(
        _('active'),
        default=True,
        help_text=_(
            'Designates whether this user should be treated as active. '
            'Unselect this instead of deleting accounts.'
        ),
    )
    date_joined = models.DateTimeField(_('date joined'), default=timezone.now)

    objects = UserManager()

    EMAIL_FIELD = 'email'
    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = ['email']

    class Meta:
        verbose_name = _('user')
        verbose_name_plural = _('users')
        abstract = True
```

```
def clean(self):
    super().clean()
    self.email = self.__class__.objects.normalize_email(self.email)

def get_full_name(self):
    full_name = '%s %s' % (self.first_name, self.last_name)
    return full_name.strip()

def get_short_name(self):
    return self.first_name

def email_user(self, subject, message, from_email=None, **kwargs):
    send_mail(subject, message, from_email, [self.email], **kwargs)
```

ДОДАТОК В

Лістинг коду модулю представлень views.py

```
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.utils.decorators import method_decorator

from apps.users.forms import RegisterForm
from django.http import JsonResponse


class ProfileView(TemplateView):
    template_name = 'users/profile.html'
    login_url = '/users/login/'

    @method_decorator(login_required)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context[‘content_type’] = self.request.user.content_set.all().order_by('-
created_at')
        return context


class RegisterView(FormView):
    template_name = 'users/register.html'
    form_class = RegisterForm
    success_url = reverse_lazy('core:index')

    def form_valid(self, form):
        user = form.save()
        login(self.request, user)
        return super().form_valid(form)

    def form_invalid(self, form):
        return JsonResponse(form.errors)


class LoginView(FormView):
    template_name = 'users/login.html'
    form_class = AuthenticationForm
    success_url = reverse_lazy('core:index')

    def form_valid(self, form):
        user = form.get_user()
        login(self.request, user)
        return super().form_valid(form)
```

Лістинг коду модулю налаштувань settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
PROJECT_DIR = os.path.dirname(os.path.abspath(__file__))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'l!6gfx!4hw1sa0#yc1))#8$972^^5w+rl3g4)w!!r@18$tufj4'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = os.environ.get('DEBUG', False)

ALLOWED_HOSTS = ['*']

# Application definition

LOCAL_APPS = [
    'apps.core',
    'apps.users',
    'apps.security',
]

INSTALLED_APPS = [
    'jet.dashboard',
    'jet',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'corsheaders',
    'qr_code',
] + LOCAL_APPS

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'apps.order.middlewares.DetectDevice',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'dipster.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(PROJECT_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'dipster.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases
    DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'dipster',
        'USER': 'www',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}

# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilari
tyValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidato
r',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidat
or',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValida
tor',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Kiev'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATICFILES_DIRS = (
    os.path.join(PROJECT_DIR, 'static'),
)
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static_content', 'static')

# Django REST Framework

REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    'DEFAULT_PERMISSION_CLASSES': [],
    'TEST_REQUEST_DEFAULT_FORMAT': 'json'
}

# Jet ADMIN

JET_THEMES = [
    {
        'theme': 'default', # theme folder name
        'color': '#47bac1', # color of the theme's button in user menu
        'title': 'Default' # theme title
    },
    {
        'theme': 'green',
        'color': '#44b78b',
        'title': 'Green'
    },
    {
        'theme': 'light-green',
        'color': '#2faa60',
        'title': 'Light Green'
    },
```

```
    {
        'theme': 'light-violet',
        'color': '#a464c4',
        'title': 'Light Violet'
    },
    {
        'theme': 'light-blue',
        'color': '#5EADDE',
        'title': 'Light Blue'
    },
    {
        'theme': 'light-gray',
        'color': '#222',
        'title': 'Light Gray'
    }
]

JET_SIDE_MENU_COMPACT = True

try:
    from dipster.local_settings import *
except ImportError:
    pass

ACCESS_KEY_HB = os.environ.get('ACCESS_KEY_HB')
SECRET_KEY_HB = os.environ.get('SECRET_KEY_HB')

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.db.DatabaseCache',
        'LOCATION': 'exchange_rate_cache',
    }
}

# REDIS related settings
REDIS_HOST = 'localhost'
REDIS_PORT = '6379'
BROKER_URL = 'redis://' + REDIS_HOST + ':' + REDIS_PORT + '/0'
BROKER_TRANSPORT_OPTIONS = {'visibility_timeout': 3600}
CELERY_RESULT_BACKEND = 'redis://' + REDIS_HOST + ':' + REDIS_PORT + '/0'
```