

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

Пояснювальна записка

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект підсистеми захисту Web-сайту інтернет-магазину»

Виконав студент 4-го курсу, групи КБ-1
спеціальності 125 «Кібербезпека»



Гаврилюк Б.М

Керівник к.т.н., доцент, доцент кафедри КІ та КБ



Пулеко І.В

Рецензент к.т.н., доцент кафедри КІ та КБ



Пірог О.В

Житомир – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

(назва прописними)

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ

(назва прописними)

СПЕЦІАЛЬНІСТЬ КІБЕРБЕЗПЕКА

(назва прописними)

ЗАТВЕРДЖУЮ

Зав. кафедри комп'ютерна

інженерія та кібербезпека

(назва кафедри)

Беріменко А.А.

« 12 » 04 20 21 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу

Студента Гаврилюка Богдана Миколайовича _____

Тема роботи: «Проект підсистеми захисту Web-сайту інтернет-магазину» _____

Затверджена Наказом університету від «12» ____ 04 2021 ____ р.
№ 134/С _____

Термін здачі студентом закінченої роботи 20.06.2021 р. _____

Вихідні дані роботи (зазначається предмет і об'єкт дослідження) _____

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пулеко І.В.	<u>12.04.21</u>	<u>18.06.21</u>
2	Пулеко І.В.	<u>12.04.21</u>	<u>18.06.21</u>
3	Пулеко І.В.	<u>12.04.21</u>	<u>18.06.21</u>

Керівник Беріменко А.А. (підпис)

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 березня 2021 – 5 квітня 2021	<i>виконано</i>
2	Проектування структури системи	6 квітня 2021 – 12 квітня 2021	<i>виконано</i>
3	Написання програмного коду	13 квітня 2021 – 26 квітня 2021	<i>виконано</i>
4	Тестування	27 квітня 2021 – 4 травня 2021	<i>виконано</i>
5	Захист	4 червня 2021	<i>виконано</i>

Студент


(підпис)

Керівник


(підпис)

РЕФЕРАТ

Випускна робота бакалавра складається з проекту підсистеми захисту інтернет магазину та пояснювальної записки. Пояснювальна записка до випускної роботи містить 80 сторінок, 24 ілюстрацій та 1 таблицю.

Мета дипломної роботи – підвищення рівня захищеності сайту інтернет-магазину шляхом розробки підсистеми захисту.

Об'єкт дослідження – процес захищеності сайту інтернет-магазину.

Предмет дослідження – підсистеми захисту сайту інтернет-магазину.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів підсистеми, опис об'єктної структури підсистеми Також наводиться результати тестування підсистеми.

КЛЮЧОВІ СЛОВА: Django, Python, Web-API.

					КБ.КВ.Б – 125 – 04 – ПЗ				
Зм.	Арк.	№ докум.	Підпис	Дата	Проект підсистеми захисту Web- сайту Інтернет магазину Пояснювальна записка	Літ.	Арк.	Аркуші	
Розроб.		Гаврилюк Б.М		23.06.21					
Керівник		Пудеко І.В		23.06.21			2		
н. контр.		Пірог О.В		23.06.21		Житомирська політехніка, група КБ-1			
Зав. каф.		Єфіменко А.А.		23.06.21					

ABSTRACT

The bachelor's graduation consists of a draft security subsystem for an Internet shop and an explanatory note. The Explanatory Note for Graduation contains 80 pages, 24 illustrations and 1 table.

The goal of the diploma work is to boost a security of a site of the Internet shop by developing a sub- system of the security.

The object of study the process of security of the online store site.

The subject of the research is the sub- systems of the site security of an Internet shop.

The work defines the main tasks for the development of the system, analyzes analogues of the developed system. The operating scenarios of the software complex, general structure of the software complex, description of algorithms of interaction of individual modules of the subsystem, description of the object structure of the subsystem are given.

KEYWORDS: Django, Python, Web-API.

					КБ.КВ.Б – 125 – 04– ПЗ	
						3

ЗМІСТ

РЕФЕРАТ.....	6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ	9
1.1. ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ	9
1.2. АНАЛІЗ МЕТОДІВ ЗАХИСТУ САЙТУ ІНТЕРНЕТ-МАГАЗИНУ	12
1.3. АНАЛІЗ ІСНУЮЧИХ ПІДСИСТЕМ ЗАХИСТУ САЙТУ ІНТЕРНЕТ-МАГАЗИНУ	19
1.4. АНАЛІЗ ВИМОГ ДО ПІДСИСТЕМИ ЗАХИСТУ САЙТУ ІНТЕРНЕТ-МАГАЗИНУ ...	26
1.5. ПОСТАНОВКА ЗАВДАННЯ.....	27
Висновки по 1 розділу	27
РОЗДІЛ 2. РОЗРОБКА САЙТУ ІНТЕРНЕТ-МАГАЗИНУ ТА ПІДСИСТЕМИ ЙОГО ЗАХИСТУ	29
2.1. РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ПІДСИСТЕМИ ЗАХИСТУ САЙТУ ІНТЕРНЕТ- МАГАЗИНУ	29
2.2. РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ	38
2.3. РОЗРОБКА ДВОФАКТОРНОЇ СИСТЕМИ ЗАХИСТУ	50
2.4. АЛГОРИТМ PBKDF2 З ХЕШЕМ SHA256.....	51
Висновки по 2 розділу	52
РОЗДІЛ 3. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ПІДСИСТЕМИ ЗАХИСТУ САЙТУ ІНТЕРНЕТ-МАГАЗИНУ	53
3.1. РОЗГОРТАННЯ МОДЕЛІ ЕКСПЕРИМЕНТУ	53

					КБ.КВ.Б – 125 – 04– ПЗ	
						4

3.2. Дослідження та порівняння налаштувань безпеки сайту при різних вразливостях	56
3.3. Рекомендації по застосуванню розробленої підсистеми захисту сайту Інтернет-магазину	59
Висновки по 3 розділу	60
Висновки	61
Список використаної літератури.....	63
Додатки.....	65
Додаток А	65
Додаток Б	67
Додаток В.....	69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД - Система управління базами даних

ПЗ - Програмне забезпечення

ООП - Об'єктно-орієнтоване програмування

ІТ - Інформаційні технології.

ЕОМ - Електронна обчислювальна машина

БД - Бази даних.

XML - Extensible Markup Language - розширювана мова розмітки

WASS - Web Application Scanning - сканування веб-додатків

WAF- Web application firewall - брандмауер веб-додатків.

TLS - Transport Layer Security - захист на транспортному рівні

SSL - Secure Sockets Layer - рівень захищених сокетів

SQL - Structured query language - мова структурованих запитів

SPAN - Standard Portfolio Analysis of Risk- система аналізу портфельних ризиків по ф'ючерсних і опціонних інструментах.

SOAP - Simple Object Access Protocol - простий протокол доступу до об'єктів

SHA256 - Secure Hash Algorithm Version - безпечний алгоритм хешування

RSA - Аббревіатура від прізвищ Rivest, Shamir та Adleman-криптографічний алгоритм з відкритим ключем

PBKDF2 - Password-Based Key Derivation Function - стандарт формування ключа на основі пароля

OWASP - Open Web Application Security Project - відкритий проєкт з безпеки веб-застосунків — онлайн-спільнота

ORM - Object-relational mapping - Об'єктно-реляційна проєкція

HTTP - HyperText Transfer Protocol - протокол передачі гіпертексту

DNS - Domain Name System-client - система доменних імен

CMS - Content Management System - Система керування вмістом

API - Application Programming Interface - інтерфейс програмного додатку

					КБ.КВ.Б – 125 – 04– ПЗ	
						6

ВСТУП

Актуальність роботи. Забезпечення інформаційної безпеки [1] на сьогоднішній день виступає в якості одного з найголовніших вимог до інформаційних систем. Причиною цього є наявність нерозривного зв'язку між інформаційними технологіями та основними бізнес-процесами в усіх організаціях, будь то державна служба, промислове підприємство, фінансова структура, оператори телекомунікацій.

Забезпечити внутрішню інформаційну безпеку – загальносвітова проблема. У перші роки впровадження корпоративних локальних мереж основний головний біль компанії полягав в несанкціонованому доступі до комерційних даних шляхом зовнішнього злому (хакерська атака), на сьогоднішній день з цією проблемою успішно справляються. У всіх ІТ-відділах поважаючих себе компаній, як правило, є велике число антивірусів, файрволи, які захищають програми та інші засоби боротьби з зовнішніми атаками.

З позиції інформаційної безпеки безліч сучасних компаній мають вигляд фортець, оточених кількома потужними стінами. Однак виходячи з практики, все одно відбувається витік інформації. Так, стає явною необхідність створення ефективних і економічно вигідних систем безпеки інформаційних систем підприємства.

У сучасності великим попитом користуються інтернет-магазини, що є повноцінним комерційним підприємством з великим товарообігом та значною економікою.

Оскільки одним з головних факторів успішної діяльності комерційного підприємства є забезпечення інформаційної безпеки було вирішено розробити підсистему захисту сайту інтернет-магазину.

Мета дипломної роботи – підвищення рівня захищеності сайту інтернет-магазину шляхом розробки підсистеми захисту.

Об'єкт дослідження – процес захищеності сайту інтернет-магазину.

Предмет дослідження – підсистеми захисту сайту інтернет-магазину.

Для досягнення поставленої мети були вирішені наступні **завдання**:

- аналіз інформаційних систем;
- аналіз і класифікація джерел загроз інформації;
- аналіз існуючих підсистем захисту;
- розробка підсистему захисту сайту інтернет-магазину;
- тестування працездатності розробленої підсистеми захисту.

					КБ.КВ.Б – 125 – 04– ПЗ	
						8

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1. Обґрунтування актуальності теми роботи

Більшість сучасних інформаційних систем створюються у вигляді Web-сайтів, тому безпеці необхідно приділити особливу увагу. Як тільки з'являється пристрій з операційною системою і програмним додатком, воно відразу ж привертає інтерес зловмисників.

Вони починають його вивчати і зламувати. В останній рік цієї тенденції було присвячено багато доповідей на великих хакерських конференціях.

Розробники не завжди приділяють особливу увагу повному захисту та безпеці створених сайтів, хоча вона і відіграє дуже важливу роль, тому що кожен сайт так чи інакше може бути підданий злому хакерів, якщо враховувати те, що сайти можуть бути, як там не є, різноманітні : електронні бібліотеки, соціальні мережі, Web-портали різних навчальних закладів, онлайн-магазини, офіційні сайти різних організацій, банків та інше.

В першу чергу, необхідно звернути увагу на безпеку веб-сервера, так як саме він відповідає за прийом і обробку HTTP-запитів[2] від клієнтів до веб-сайту. Саме він забезпечує функціонування численних веб-сайтів по всьому світі, й так само відповідає за базові послуги і зберігає персональні дані, як користувачів, так і відвідувачів сайту. Необхідність захисту серверів - це одне з найважливіших завдань будь-якої організації.

Навіть під час появи перших Web-сайтів зломщики здійснювали атаки на сайти важливих організацій, наприклад, такий як, американський Сітібанк (1994р.), в результаті чого було вкрадено 12 млн. Доларів, жертвою так само стали і сайти НАТО, ЦРУ, і Мін'юсту США.

В даний час, здавалося б, таких загроз повинно стати набагато менше чи не бути зовсім, але, на жаль, це далеко не так: як і раніше, зараз проводиться досить велика кількість кібератак, причому в основному на такі глобальні організації, як урядові установи і банки. Це обумовлюється тим, що на них

					КБ.КВ.Б – 125 – 04– ПЗ	9

можна «заробити» велику кількість грошей або ж отримати / змінити інформацію, яка здатна якимось чином вплинути важливі для держави події.

Досить важливим моментом є те, що географічне розташування сервера, аж ніяк, не впливає на його захист. Провести атаку на нього можна з будь-якої точки доступу. Це пов'язано з тим, що Web-сервери в силу своєї відкритості розраховані на передачу інформації між користувачами і саме тому мають безліч вразливостей. Наприклад, зловмисник може внести якісь зміни (модифікації) в код HTTP сервера або сервера бази даних, або самих сторінок веб-сайту, помінявши його початкову функціональність.

Найбільш поширені дії правопорушників кібератак над зламаними сайтами, які найчастіше застосовуються і до цього дня:

- опублікували неправдиву інформацію на сайті;
- дістали несанкціонований доступ до сервера, де були зібрані дуже вартісні відомості;
- атакували базу даних про власників пластикових карт банку;
- вкрали дані, що стосуються номерів рахунку основних власників банківської карти, їх імен, прізвищ, контактної інформації, а потім викрали велику кількість грошей;
- створили віруси, які легко зламували паролі, а після використовували заражені машини для розсилки спаму або як бази для зберігання вкраденої інформації;
- зробили атаки на інтернет-сайти офіційних (урядових) установ різних країн;
- поширили віруси, які привели до значного уповільнення швидкості роботи інтернету (були випадки, що в результаті таких вірусів деякі регіони виявилися і зовсім відрізаними від мережі);
- DoS-атаки (Denial of Service- "відмова в обслуговуванні") на мережу з метою переповнення її додатковими запитами так, щоб корисний трафік або знижувався до мінімуму, або зовсім переривався;

Зараз існують різні програми, а так само спеціальне Web-тестування, які можуть визначити приблизний коефіцієнт безпеки і надійності сайту. Завдяки такому тестуванню можна заздалегідь оцінити безпеку сайту і його вразливість.

Передбачається, що кожен третій сайт відстежується сторонніми людьми. Це досить неприємний і насторожуючий факт для Web-розробників. Маючи всю необхідну інформацію про уразливість сайту, можна з легкістю його зламати, тому головне завдання розробників в питанні про надійність і захист сайту полягає в тому, щоб запобігти якнайбільше цих вразливостей або хоча б як слід їх заховати.

Так само існує кібершпіонаж, тобто коли зловмисники не атакують інформаційну систему (або сайт), а просто займаються крадіжкою всієї необхідної інформації для своєї вигоди.

Більші підприємства підходять до інформаційної безпеки (ІБ)[3] з набагато більшою відповідальністю, ніж компанії малого та середнього бізнесу (СМБ), саме тому вони страждають найбільше. За статистикою, саме у них виникає більше інцидентів в контексті інформаційної безпеки.

Головними жертвами DDoS-атак в основному залишаються інтернет-магазини, телекомунікації і торгові майданчики (рис 1.1) [4].

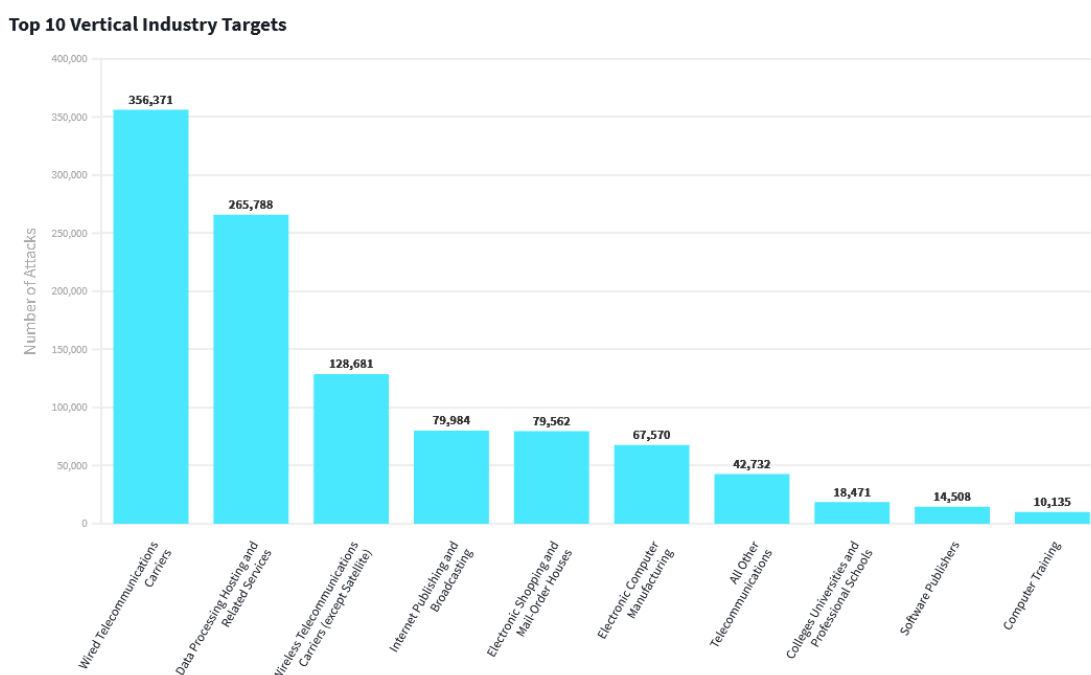


Рисунок 1.1 – Статистика жертв DDoS-атак

Впровадження сучасних засобів захисту є невід'ємною частиною заходів щодо забезпечення інформаційної безпеки. Захищати свій сайт - це одне з найважливіших завдань Web-розробників. Якщо не приділяти цьому особливу увагу – можна придбати (створити) безліч збитків, як у матеріальному плані, так і в політичному. У зв'язку з тим, що інтерес зломисників до атак на Web-сайти не замовкає навіть зараз (якщо врахувати те, що всі зломисні діяння караються законом), розробникам необхідно в першу чергу задуматися про його безпеку. Існує безліч засобів захисту Web-сайтів, але якими б вони не були ефективними все одно необхідно періодично тестувати Web-сайти на уразливості, якими в будь-який момент може скористатися зломщик.

Таким чином, питання створення систем захисту інтернет-магазинів є актуальним.

1.2. Аналіз методів захисту сайту інтернет-магазину

Засоби захисту веб-сайтів (додатків) - програмні або апаратно-програмні комплекси, призначені для забезпечення захисту веб-додатків і різних веб-сервісів.

До засобів захисту веб-сайтів можна віднести наступні:

- Web Application Firewall (WAF).
- Засоби аналізу веб-сайтів на наявність вірусів.
- Балансувальник навантаження на веб-додатки.
- Засоби захисту від DDoS.
- Сканери захищеності веб-додатків (WASS).

Засоби захисту веб-сайтів забезпечують захист від наступних типів загроз:

- Загроза зміни контенту веб-сайту. Дана атака дозволяє зломисникові розмістити на веб-сайті будь-яку інформацію, включаючи текст сторінок, а також дані платіжних реквізитів.

- Загроза видалення даних. Дана атака дозволяє зловмисникові видалити всі дані, включаючи інформацію про паролі, бази даних клієнтів, постачальників, товарів та інші.
- Загроза застосування шкідливих програм. В результаті такої атаки користувачі і власники сайту можуть не відразу помітити будь-які зміни. Проте шкідлива програма зможе виробляти шкідливі дії, включаючи переадресацію на шкідливий або шахрайський сайт, крадіжку персональних даних користувачів, зараження відвідувачів веб-сайту вірусами і так далі.
- Загроза розсилки спаму. В результаті даної атаки веб-додаток буде використовуватися зловмисниками для розсилки небажаних повідомлень. Це загрожує занесенням веб-сайту в спам-листи, що не дозволить надалі відправляти санкціоновані повідомлення.
- Загроза збільшення навантаження на ресурси, або DDoS-атака. Під цим мається на увазі відправка на адресу веб-сервера спеціальних запитів, які приведуть до утруднення доступу користувачів до сайту або падіння прикладного та системного програмного забезпечення веб-сервера.

Розглянемо детальніше кожен метод захисту.

WAF (Web Application Firewall) - міжмережевий екран для веб-додатків. Це інструмент для фільтрації трафіку, що працює на прикладному рівні і захищає веб-додатки методом аналізу трафіку HTTP / HTTPS і семантики XML / SOAP. WAF може встановлюватися на фізичний або віртуальний сервер і виявляє найрізноманітніші види атак. Діє міжмережевий екран як проксі-сервер, але з огляду на можливості вивчати HTTPS-трафік методом перевірки сертифіката конкретного сервера WAF розрахований на виконання додаткових операцій: балансування навантаження на сервер, термінацію трафіку SSL і т.д. WAF може працювати з кластеризацією і акцелерацією додатків.

WAF може вбудовуватися в мережу як:

Монітор. Моніторинг мережі в режимі реального часу за допомогою порту SPAN.

Gateway. 3 режими проксі: transparent, bridge і reverse. Працює WAF за такими моделям безпеки:

Negative. Якийсь «чорний список», який забороняє прийом конкретної інформації, прописаної в налаштуваннях. Захищає веб-додатки на прикладному рівні (аналог IPS), але вміє оцінювати потенційні загрози детальніше і частіше застосовується для забезпечення захисту від «популярних» і специфічних типів атак. Аналізує уразливості конкретних веб-додатків.

Positive. «Білий список», що дозволяє прийом конкретної інформації, яка була заздалегідь зазначена в налаштуваннях. Дозволяє отримати максимальний захист, тому що застосовується в якості доповнення до моделей. Задіє другий тип логіки: правила, що визначають, що конкретно дозволено.

Приклад роботи Negative: забороняти заздалегідь заданий «поганий» запит GET по HTTP і вирішувати все інше.

Приклад роботи Positive: дозволяти зазначені раніше запити GET по HTTP для заданої адреси і забороняти все інше.

Можливості WAF

Швидко реагувати на будь-які різновиди атак на веб-додатки, які входять в OWASP Top 10 (Open Web Application Security Project - відкритий проект забезпечення безпеки веб-ресурсів)[5].

Захист забезпечується заданими активними правилами.

Перевіряти чи надходять на додаток трафік HTTP / HTTPS і інші запити, адресовані веб-додатків, після чого приймати рішення на підставі заданих правил і політики (блокувати, дозволити, відправити повідомлення).

Підтримувати стабільну роботу моделей безпеки Negative і Positive, а також дотримуватися всіх заданих в їх рамках правил.

Перевіряти і аналізувати контент, створений за допомогою HTML і DHTML, а також CSS і прикладних протоколів передачі HTTPS, HTTP.

Запобігати витоку інформації, перевіряючи що виходить від веб-додатків трафік HTTP / HTTPS, і приймати задані заходи на підставі заданих активних правил.

Постійно вести журнал подій і записувати в нього всі виконані операції, аналітичну інформацію та інші події, що відбулися.

Аналізувати веб-сервіси (частково публічні) методом аналізу XML (eXtensible Markup Language), обміну структурованими повідомленнями SOAP і перевіряти HTTP веб-сервера на наявність моделей взаємодії.

Перевіряти всі вхідні дані, що застосовуються для відправки / отримання інформації від веб-додатків.

Захищати від атак, спрямованих конкретно на сам Web Application Firewall.

Термінувати TLS і SSL – розшифровувати і перевіряти трафік перед тим, як відправити його веб-додатком.

Головна відмінність брандмауера від інших методів захисту веб-додатків – глибокий аналіз трафіку протоколів прикладного рівня.

Засоби аналізу веб-сайтів на наявність вірусів.

Статичний аналіз сторінок - це пошук шкідливих вставок (переважно javascript), спам-посилань і спам-контенту, фішингових сторінок та інших статичних елементів на перевірених сторінці і в підключаються файлах. Виявлення подібних фрагментів виконується на основі бази сигнатур або деякого набору регулярних виразів. Якщо шкідливий код постійно присутній на сторінці або в завантажуваних файлах, а також відомий веб-сканеру (тобто він доданий в базу сигнатур), то веб-сканер його виявить. Але так буває не завжди. Наприклад, шкідливий код може завантажуватися з іншого ресурсу або виконувати якісь несанкціоновані дії за певних умов:

- по завершенні завантаження сторінки в неї додається javascript, який виконує drive-by download атаки
- користувач йде зі сторінки, в цей момент підвантажується код і відкриває popunder з контентом "для дорослих"
- відвідувач сайту знаходиться на сторінці кілька секунд і тільки після цього його перенаправляють на платну підписку за смс

Якщо заздалегідь невідомо, який код провокує дані несанкціоновані дії, то виявити його статичним аналізом надзвичайно складно. На щастя, є аналіз динамічний або іноді його ще називають "поведінковим". Якщо веб-сканер розумний, він буде не просто аналізувати вихідний код сторінки або файлів, але ще і намагатися робити якісь операції, емулюючи дії реального відвідувача. Після кожної дії або за певних умов робот сканера аналізує зміни і накопичує дані для підсумкового звіту: завантажує сторінку в декількох браузерях (причому, не просто з різних User-Agent'ів, а з різними значеннями об'єкта navigator в javascript, різними document.referer і т.п.), прискорює внутрішній таймер, відловлює редіректи на зовнішні ресурси, відстежує те, що передається в eval (), document.write () і т.п. Високого рівня веб-сканер завжди буде перевіряти код сторінки і об'єкти на ній як до початку виконання всіх скриптів (відразу після завантаження сторінки), так і через деякий час, оскільки сучасні "шкідливий" динамічно додають або приховують об'єкти на javascript, а також виконують фонові завантаження всередині динамічних фреймів. Наприклад, код зараженого віджета може через 3 секунди або по руху миші завантажити скрипт, який вставить на сторінку javascript з перенаправленням на завантаження небезпечного .apk файлу. Природно, ніякий статичний аналіз (крім як заздалегідь знати, що віджет небезпечний) або пошук по файлах таке не виявить.

Балансування навантаження [6] – це спосіб розподілу навантаження між декількома мережевими пристроями, такими як сервери, маршрутизатори, міжмереві екрани. Такий підхід забезпечує оптимальне використання потужностей, а також збільшення швидкості реагування серверів на запити і забезпечення відмовостійкості. Можна налаштувати балансування навантаження на багаторівневі комутатори, DNS-сервери, серверні кластери, проксі-сервери, веб-сервери, мережеві адаптери, віртуальні машини, додатки і сервера інспектування вмісту.

Балансування навантаження може виконуватися на різних рівнях моделі OSI:

					КБ.КВ.Б – 125 – 04– ПЗ	
						16

Мережевий. Такий метод має на увазі використання одної IP-адреси на різних мережевих пристроях, серверах. Існує кілька способів балансування навантаження на мережевому рівні:

- DNS-балансування, коли одному доменному імені присвоєно кілька IP-адрес.
- NLB-кластер, або об'єднання декількох серверів в кластер.
- За територіальною ознакою, коли в різних регіонах налаштовуються однакові сервіси, які мають однакові IP-адреси.

Транспортний. Такий спосіб має на увазі централізований розподіл запитів до сервера між різними пристроями. При такому балансуванні навантаження можуть використовуватися різними алгоритмами (наприклад, запит буде направлятися найменш завантаженому сервера або того сервера, який знаходиться наступним в черзі, так званий кругової перебір).

Прикладний. При такому балансуванні навантаження існує певна програма, яка самостійно приймає рішення про перенаправлення клієнтського запиту на відповідний сервер в залежності від того, до чого звертається клієнт. Цей спосіб застосовується при роботі з базами даних (додаток може перенаправляти запити на читання до одного сервера, а запити на запис – до іншого).

Балансування навантаження може використовуватися для реалізації наступних цілей:

- Виключення можливості виділення ресурсів сервера на єдиний запит в той час, як інші запити будуть відхилятися.
- Ефективне використання ресурсів, коли не допускається простоювання сервера, в той час як завантажені інші.
- Збільшення продуктивності або зменшення часу, необхідного для виконання запиту.
- Зменшення часу, який потрібен для надання відповіді на запит клієнта.

При розробці варіантів балансування навантаження слід враховувати, що адміністратор повинен повністю розуміти, яким чином буде відпрацьовуватися балансування навантаження, яким чином можна буде в подальшому масштабувати архітектуру. Балансування навантаження забезпечує:

- управління трафіком.
- збільшення швидкості роботи додатків. захист додатків і збереження інформації.
- моніторинг трафіку.

Балансування навантаження широко використовується в корпоративних системах захисту, забезпечуючи рівномірне навантаження на сервери, для захисту веб-сервісів і сайтів, системах фільтрації веб-контенту і т. п. Також існують різні онлайн-сервіси по організації балансування за певним методом.

Засоби захисту від DDoS-атак[7] - це спеціалізовані програмно-апаратні і програмні засоби, призначені для захисту вер-серверів (веб-сайтів) організації від розподілених атак типу «Відмова в обслуговуванні». Даний тип атак націлений на порушення доступності (повне або часткове) різних публічних сервісів, в тому числі веб-сайтів, баз даних і серверів додатків, при якому у легітимних користувачів пропадає можливість отримання доступу.

Принцип атаки полягає в генерації великої кількості паразитного трафіку, на обробку якого буде виділятися безліч обчислювальних ресурсів. В результаті атаки організації можуть понести великі фінансові втрати через те, що їх замовники не зможуть скористатися сервісами компанії. У загальному сенсі захист від DDoS можна організувати двома способами:

- Скориставшись послугами зовнішніх компаній за моделлю SaaS (Security as a Service). Таке рішення дозволить уникнути глибокого занурення в технічну складову питання захисту від DDoS-атак, а також скоротить витрати на інформаційну безпеку.
- Шляхом впровадження власних засобів захисту від атак типу «Відмова в обслуговуванні», а також застосування організаційно-технічних

заходів, які націлені на збільшення пропускної спроможності каналів зв'язку та використання механізмів балансування навантаження.

Засоби захисту від DDoS-атак on-premise зазвичай розташовуються на кордоні мережі в розрив, для своєчасного виявлення і фільтрації шкідливого або паразитного трафіку. Такі засоби вимагають постійного оновлення, так як сценарії DDoS-атак постійно змінюються. Ціна на спеціалізоване обладнання, як правило, залежить від заявленої швидкості обробки трафіку і пропускної здатності. З огляду на високу вартість засобів захисту, оптимальний комплекс може не впоратися з обробкою дуже великої DDoS-атаки.

1.3. Аналіз існуючих підсистем захисту сайту інтернет-магазину

Сучасна торгівля вже освоїла такий канал залучення і обслуговування клієнтів, як Інтернет. Перевага цього каналу оцінили вже навіть найбільш консервативні продавці і з екзотичної іграшки електронна комерція перетворилася в бізнес з багатомільйонними оборотами.

Економічна криза тільки прискорив зростання електронної комерції - продажу через Інтернет дозволяють продавцям оптимізувати складські запаси, а також заощадити на торгових площах і персоналі. Деякі рітейлери навіть перетворюють вже наявні магазини в шоу-руми, в яких немає складів, а після вибору вподобаного товару покупець замовляє його через Мережу з терміналу, встановленого в магазині.

Однак, як і завжди в бізнесі, нові можливості несуть з собою і нові ризики. Так, веб-сайт з можливістю вибрати товар, оплатити замовлення і замовити доставку, приносить прибутку і знижує витрати, проте власникам доводиться дбати про те, щоб він нормально функціонував, не допускав шахрайських операцій, не допускав витоків призначених для користувача даних і інших інцидентів, що впливають на обсяг продажів, прибуток, лояльність клієнтів і санкції регуляторів ринку.

Перший фокус захисту - це постійна доступність магазину. Немає сенсу проводити рекламні компанії та оптимізувати складські замовлення, якщо

					КБ.КВ.Б – 125 – 04– ПЗ	19

користувачі, перейшовши по рекламованій адресі, не отримають доступу до сайту.

Але і сайт, який зберігає свою доступність, теж може нести в собі загрозу. На сайті може зберігатися конфіденційна інформація - дані облікових записів (особистих кабінетів) клієнтів сайту, які можуть містити не тільки персональні дані - прізвища, імена, по батькові, номер мобільного телефону, адресу доставки і т.п., але і дані прив'язаних до особистого кабінету платіжних карт, які зловмисники можуть використовувати для оплати товарів.

Чим більше електронний магазин, тим більше у нього зареєстрованих користувачів, тим більший інтерес представляє база облікових записів клієнтів компанії, яка зберігається на сайті. Така інформація коштує хороших грошей тому, що з її допомогою можна отримати пряму вигоду - використовувати дані кредитної картки для оплати своїх покупок або оплачувати товари від імені легальних користувачів, але вказувати іншу адресу доставки.

А зовсім недавно був зламаний сайт компанії Uber. І зловмисники розплачувалися за свої поїздки засобами кредитних карт, прив'язаних до облікових записів нічого не підозрюючих користувачів. Зрозуміло, що такі інциденти не додають клієнтам, особливо постраждалим, лояльності до сервісу.

Важливо, щоб інформація на сайті відповідала тому, що хотіли розмістити його розробники. Це особливо важливо для комерційних сайтів – спотворення інформації про товар може привести до відмови клієнта від покупки або відмови магазину від поставки оплаченого по неправильно виставленої ціною товару. Трапляються більш складні атаки, ніж підміна ціни або опису товару: наприклад, зловмисники підміняють телефон «гарячої лінії» або посилання з кнопки «купити» на відповідні дані реального або шахрайського магазину, перенаправляючи на нього покупців, вже готових купити товар після ознайомлення.

Довіряючи програмістам реалізовувати бізнес-процеси, компанії часто не перевіряють додаток, що запускається на наявність так званих «декларованих можливостей» – функціоналу, не передбаченого в технічному завданні. Це

					КБ.КВ.Б – 125 – 04– ПЗ	
						20

можуть бути як службові функції, наприклад, налагодження гілки, що спрощують деякі операції в процесі тестового функціонування сервісу, так і навмисні «закладки», що реалізують можливості зловмисних операцій.

Існує багато систем захисту веб-додатків, як інтегрованих, так і спеціалізованих: різного роду сканери, анти-DDoS продукти і сервіси, міжмережеві екрани прикладного рівня (WAF-web application firewall). Сьогоднішня проблема зовнішніх (на сленгу - "навісних") систем захисту в тому, що вони не встигають перебудовуватися так швидко, як змінюється об'єкт захисту.

Два основних способи зміни налаштувань систем захисту - це зміни вручну і самонавчання. Жоден з цих способів не працює при частоті змін навіть сто змін на добу. За експлуатаційної документації на WAF, наприклад, самонавчання такого продукту займає від однієї до трьох годин, тобто якщо об'єкт захисту змінюється частіше ніж один раз на годину, то всі нові функції будуть сприйматися системою захисту як аномалії і блокуватися. Для того ж, щоб постійно налаштовувати систему захисту вручну, треба мати солідну професійну команду, яка працює в цілодобовому режимі.

Неможливість «включити захист і забути про нього» диктує дві виграшні стратегії: пасивну та активну.

Пасивна стратегія полягає в тому, щоб включити всі системи захисту в пасивному режимі, тобто відключити можливість блокування шкідливих операцій. При цьому трафік (а точніше – його копія) до торгового сайту та операції на ньому аналізується різними системами – від анти-DDoS до антифрода, збираючи, нормалізує і корелюючи повідомлення про події. За подіями стежить чергова зміна інформаційної безпеки, яка, запідозривши атаку, приймає рішення її відобразити і робить це вручну – наприклад, перемикає трафік на центр очищення від трафіку DDoS-атаки, скидає підозріле з'єднання, блокує скомпрометований обліковий запис і т.п.

Плюсом такого підходу є можливість використання паралельно багатьох систем захисту (котрась із них так визначить атаку). Мінусом же є делегування

прийняття рішення операторам системи контролю. Це вимагає наймати дорогу команду інформаційної безпеки, що при інтенсивному трафіку не гарантує відсутності інцидентів – буває, що рішення операторами приймаються, коли вже пізно - атака вже пройшла.

Активна стратегія виражається у виборі рішень, здатних без помилкових спрацьовувань працювати в режимі блокування, і інтеграції їх таким чином, щоб налаштування змінювалися автоматично. Будь-яка зміна на захищеному сайті повинна негайно досліджуватися на предмет наявності вразливостей, знайдені вразливості повинні відправлятися на виправлення відповідальним за цей функціонал – адміністраторам, інженерам або програмістам – а до виправлення можливість зловмисної експлуатації функціоналу має блокуватися «навісними» системами інформаційної безпеки.

Найбільш часта зв'язка активних систем безпеки – статичний сканер досліджує код додатка, знаходить в ньому уразливості, які аналізуються динамічним сканером для відкидання помилкових спрацьовувань, далі знайдені і підтверджені можливістю експлуатації уразливості передаються WAF і анти-DDoS, які блокують небезпечні звернення до ці вразливостей. Паралельно інформація про уразливість до рекомендацій щодо виправлення передається програмістам або інженерам, в залежності від того, що треба міняти - функціонал додатка, вміст сайту або налаштування інфраструктури.

Плюсом такого рішення є активність, інтегрованість і незалежність від людини - одного разу налаштоване рішення працює само собою, без помилкових спрацьовувань відбиваючи атаки. Клієнти таких рішень про атаки дізнаються зі звітів і можуть використовувати своїх кваліфікованих фахівців для більш творчої роботи, ніж чергування. Такі рішення особливо стали затребувані в умовах економічної кризи, коли не кожна торгівельна площа може дозволити собі кілька змін кваліфікованих фахівців інформаційної безпеки для цілодобових чергувань.

Розглянемо декілька прикладів систем захисту веб-сайтів.

Всі способи захисту ділять на три великі групи(рис. 1.2):

					КБ.КВ.Б – 125 – 04– ПЗ	
						22



Рисунок 1.2 – Способи захисту веб-сайтів

Захист CMS

CMS - система управління сайтом і всією інформацією на ньому. Це серце, а тому захист починається саме з нього. Використовуйте наступні методи і максимально забезпечте ресурс.

SSL-сертифікат Протокол SSL забезпечує надійний захист даних в інтернеті за рахунок зашифрованої передачі інформації. Для того щоб такий рівень безпеки був доступний, важливо мати SSL-сертифікат - електронний цифровий підпис вашого сайту, що містить:

- доменне ім'я;
- відомості про юридичну особу, на яку оформляється сертифікат;
- реальне місцезнаходження власника;
- термін дії;
- основні дані про постачальника сертифіката.

За допомогою цього підпису підтверджується, що домен належить реальній компанії, а власник використовує секретний ключ законно. Сайт захищений SSL від фішинг атак.

Є два способи отримати SSL-сертифікат: звернутися до провайдера, що надає підтримку SSL, або в центр сертифікації. Наприклад, для сайтів, розміщених на сервісах Google (Google Мій бізнес, Blogger), а також для тих,

хто співпрацює з партнерами (Bluehost, Shopify, Weebly, Wix), сертифікат видається безкоштовно.

Якщо звертатися в спеціалізовані компанії для сертифікації, доведеться витратитися – до 100 \$.

Однією важливою умовою безпечного функціонування ресурсу - регулярне оновлення CMS. Кожна нова версія движка усуває помилки і закриває незахищені місця, через які легко зламати сайт.

Додатковий захист ресурсу забезпечують спеціальні плагіни. Їх дуже багато, кожен пропонує комплексне рішення для оборони від тих чи інших атак, сканування на наявність вірусів і спроб злому, створення копій ресурсу та інших функцій. Вони швидко встановлюються, та є простими в настройках і ефективно працюють.

Наведемо кілька прикладів популярних плагінів для WordPress, одного з найпопулярнішого CMS:

iThemes Security

Плагін, що пропонує 30 видів захисту від:

- SQL-ін'єкцій;
- Шелл;
- XSS;
- злому через FTP або SSH;
- злому phpMyAdmin;
- злому через сусідів по хостингу;
- злому через уразливість в движку;
- брутфорс-атак і ботів, які шукають уразливості ресурсу;
- захист від ДДОС-атак на сайт.

iThemes Security знаходить проломи і підбирає способи їх усунення, відображає будь-які атаки на файлову систему і бази даних, інформуючи вас про зміни. До того ж плагін може змінювати IP користувача, адреса адмінки, входу на сайт, шлях до папки wp-content. З його допомогою створюються бекапи для швидкого відновлення втрачених даних.

Wordfence Автоматично перевіряє на зараження і спроби злому. До основних функцій відносяться розпізнавання шкідливого трафіку, погроз, блокування спроб впровадження вірусного коду і користувачів, які порушують правила взаємодії з сайтом, забезпечення безпечного входу. При подальшому скануванні надає звіт з виявленими можливими проблемами і результатами боротьби з ними.

Захист сервера

Для того щоб сервер не був доступний для зловмисників, виконуються такі правила технічної безпеки:

- Обмежуються користувачі в правах на доступ до бази даних.
- Відстежується, що вводить користувач на сайті.
- Постійно змінюються паролі. Створити складний пароль допоможуть онлайн-генератори.
- Створюються бекапи сайту: за допомогою панелі управління хостингу, плагінів або звернення до тех. підтримки хостингу.

У більшості хостингів є можливість робити копії сайту. Для цього потрібно зайти в панель управління, знайти цю функцію і слідувати інструкціям. Прикладом слугує хостинг ukraine.com.ua.

Створювати бекапи можна і плагінами. Вони бувають платними і безкоштовними, з можливістю автоматичного копіювання, роблять повні або часткові копії, зберігають на комп'ютер або в хмарний сервіс. Наприклад, для WordPress є такі плагіни, як BackUpWordPress, BackWPup, BackupBuddy, Backup by Supsysic і дуже багато інших. Вибирайте в залежності від ваших завдань, розміру сайту, функціональних можливостей.

Комп'ютери того, хто користується адмін. панеллю і має доступ до таємної інформації на сайті, а також тих, у кого є вихід до сервера, потрібно перевіряти на наявність вірусів, чи впораються з цим антивіруси, постійний захист веб-сайтів гарантує збереження даних та інформації, важливо оновлювати програми до актуальної версії.

Найнадійнішими і популярними антивірусами на ринку СНГ є:

					КБ.КВ.Б – 125 – 04– ПЗ	
						25

- Bitdefender Antivirus Free Edition;
- ESET NOD32 Internet Security;
- Avast Free Antivirus;
- Kaspersky Internet Security;
- BullGuard Internet Security;
- Norton Security Deluxe;
- Trend Micro Internet Security;
- Avira Antivirus Pro.

1.4. Аналіз вимог до підсистеми захисту сайту інтернет-магазину

На основі вищеперерахованого аналізу, все це може привести до різних проблем - від крадіжки паролів користувачів, злому сайту і до повної непрацездатності веб-ресурсу. Існує безліч інших способів і механізмів, здатних завдати шкоди вашому сайту, які не опишеш в рамках однієї статті. Але є також і ефективні інструменти для захисту даних. Виділено наступні вимоги до розроблюваної підсистеми захисту інтернет-магазину:

- Одна з них – протокол https, який відрізняється від звичайного http тим, що при обміні даними між користувачем і сервером генерується спеціальний ключ. Він оновлюється при кожній новій сесії, що не дає зломисникам перехопити дані. Такий протокол використовується банківськими системами, інформаційними сайтами і порталами і іншими ресурсами, які відповідають за безпеку даних користувачів – що називається, «головою».
- Інша вимога до сайту з інформаційної безпеки – налаштування двофакторної автентифікації, тобто двох кроків для перевірки справжності введення даних. Як правило, 2fa автентифікація здійснюється за допомогою одноразового коду, QR-код який система відправляє на телефон або пошту користувача, або за використанням біометричних даних.

- Складні та зашифровані паролі. Безпека користувачів один з головніших факторів працездатності онлайн-ресурсу. Всі збережені у базі даних паролі мають бути якісно зашифровані.
- Доступ до адміністративної панелі. Оскільки будь-який інтернет магазин використовує CMS, вона повинна бути ретельно захищена розмежуванням доступу.

1.5. Постановка завдання

У даній роботі розглядається питання розробки підсистеми захисту сайту інтернет-магазину. Узагальнюючи, можна виділити наступні завдання для розроблюваної підсистеми захисту:

- Система захисту інформації повинна бути представлена як щось захищене. Цілісність системи буде виражатися в наявності єдиної мети її функціонування, інформаційних зв'язків між елементами системи, ієрархічності побудови підсистеми управління системою захисту інформації.
- Система захисту інформації повинна забезпечувати безпеку інформації, засобів інформації, захист інтересів учасників інформаційних відносин і неможливість несанкціонованого доступу зломисника до інформації, що захищається.
- Система захисту інформації в цілому, де застосовуються методи і засоби захисту повинні бути по можливості прозорими для законного користувача, не створювати йому великих додаткових незручностей, пов'язаних з процедурами доступу до інформації.

Система захисту інформації повинна забезпечувати оцінку загроз зовнішніх дестабілізуючих факторів і захист від них.

Висновки по 1 розділу

У даному розділі проведено аналітичну роботу над доведенням актуальності теми роботи, а саме розробки підсистеми захисту сайту інтернет-

					КБ.КВ.Б – 125 – 04– ПЗ	27

магазину. Виділено види загроз, що існують у сучасності, приведено приклади та статистику жертв одного з найпоширенішого виду атак – DDoS.

Виділено методи захисту веб-сайтів, розглянуті переваги та недоліки таких методів, як:

- Web Application Firewall (WAF).
- Засоби аналізу веб-сайтів на наявність вірусів.
- Балансувальник навантаження на веб-додатки.
- Засоби захисту від DDoS.
- Сканери захищеності веб-додатків (WASS).

Проведено аналіз існуючих систем захисту веб-сайтів, наведено конкретні приклади.

Виділено перелік вимог до розроблюваної підсистеми захисту сайту інтернет-магазину.

РОЗДІЛ 2. РОЗРОБКА САЙТУ ІНТЕРНЕТ-МАГАЗИНУ ТА ПІДСИСТЕМИ ЙОГО ЗАХИСТУ

2.1. Розробка структурної схеми підсистеми захисту сайту інтернет-магазину

Задля планування структурної схеми підсистеми захисту, першим кроком варто розглянути структуру самого інтернет-магазину, а саме його внутрішніх складових.

Патерн проектування (або шаблон проектування) – це повториме архітектурне рішення[8]. У різних місцях додатки можуть (і будуть) використовуватися різні патерни. Патерни аж ніяк не є панацеєю. Але вони дозволяють відносно швидко скористатися досвідом вирішення складних завдань іншими програмістами. Очевидно, що для розробки інформаційних систем потрібно підібрати найбільш вдалий патерн для конкретного завдання.

Почнемо з першого та головного - Model-View-Controller. MVC[- це фундаментальний патерн, який знайшов застосування в багатьох технологіях, дав розвиток нових технологій.

Архітектуру MVC можна охарактеризувати двома пунктами:

- Уявлення - це візуальна проекція моделі
- Контролер - це з'єднання між користувачем і системою

Контролер приймає вхідні дані, а уявлення - вихідні, однак велике число операцій відбувається і між ними. Дана архітектура добре підходить для невеликих та середніх проектів.

Відмінності MVC & MVVM & MVP. Найбільш поширені види MVC-патерну, це:

- Model-View-Controller
- Model-View-Presenter
- Model-View-View Model

Розглянемо і порівняємо кожен з них.

Model-View-Presenter(див. рис. 2.1)

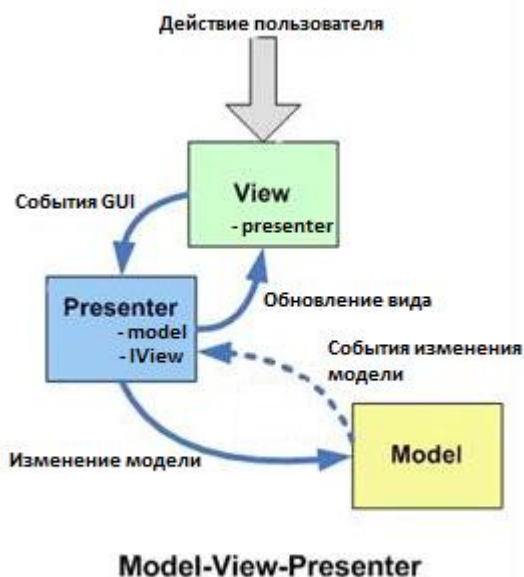


Рисунок 2.1 – Схема Model-View-Presenter

Дана архітектура полегшує unit-тестування, презентер (presenter) простий для написання тестів, а також може багаторазово використовуватися, тому що уявлення може реалізувати кілька інтерфейсів.

З точки зору того, як краще і коректніше створювати інтерфейси, необхідно розглядати MVP і MVC тільки як основні ідеї, а не патерни розробки.

Як і інші подібні патерни (MVC, MVVM), MVP дозволяє прискорити розробку і розділити відповідальність різних фахівців; додаток зручніше тестувати і підтримувати. Найчастіше його використовують розробники мобільних додатків, однак він набагато популярніший, ніж спочатку може здатися. Справа в тому, що цей патерн активно застосовується в інтернеті, хоча там його і називають MVC.

Основна відмінність MVP і MVC в тому, що в MVC оновлена модель сама каже, що потрібно показати інші дані. Якщо ж цього не відбувається і з додатком потрібен посередник у вигляді представника, то патерн варто називати MVP.

Model-View-View Model(див. рис. 2.2)

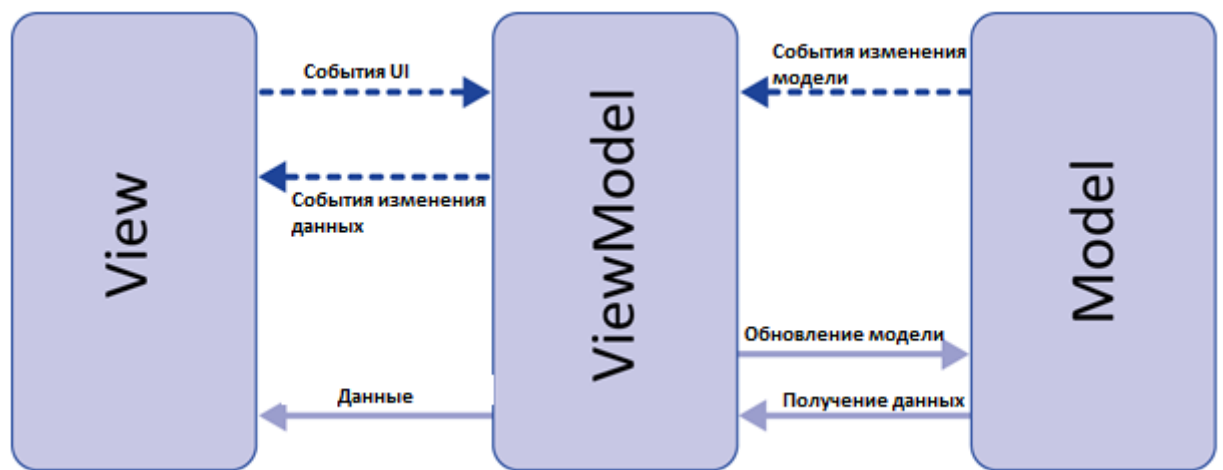


Рисунок 2.2 – Схема Model-View-View Model

Існує інший спосіб біндингу: замість прив'язування уявлення до інтерфейсу, прив'язуються елементи уявлення до параметрів view-моделі - така архітектура називається MVVM. Наприклад, поля email, password, і розмітка визначені за допомогою зв'язувань. Коли ми змінюємо параметри в нашій моделі, в розмітку теж вносяться зміни.

ViewModel не може контактувати зі View безпосередньо. Замість цього вона являє легко зв'язуючі властивості і методи у вигляді команд. View може прив'язуватися до цих властивостей, щоб отримувати інформацію з ViewModel і викликати на ній команди (методи). Це не вимагає того, щоб View знала про ViewModel. XAML Databinding використовує рефлексію, щоб зв'язати View і ViewModel. Таким чином, ви можете використовувати будь-яку ViewModel для View, яка надає необхідні якості.

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатку від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.

Model-View-Controller(див. рис. 2.3)

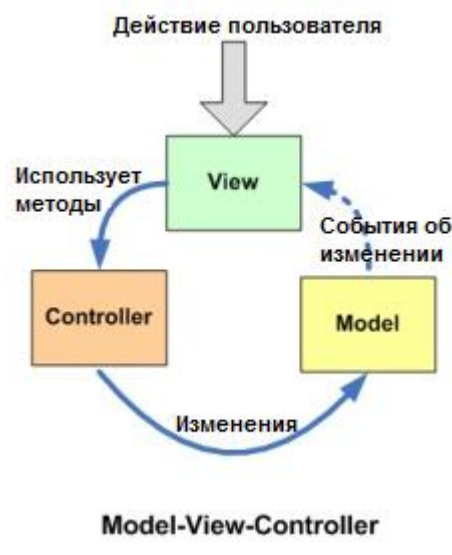


Рисунок 2.3 – Схема Model-View-Controller

Основна ідея цього патерну в тому, що і контролер і уявлення залежать від моделі, але модель ніяк не залежить від цих двох компонентів. Ознаки контролера:

- Контролер визначає, які уявлення має бути відображено в даний момент;
- Події вистави можуть вплинути тільки на контролер. Контролер може вплинути на модель і визначити інше уявлення.
- Можливо кілька подань тільки для одного контролера;

Контролер перехоплює подію ззовні і відповідно до закладеної в нього логіки, реагує на цю подію змінюючи модель, за допомогою виклику відповідного методу. Після зміни модель використовує подію про те що вона змінилася, і всі підписані на це події уявлення, отримавши його, звертаються до моделі за оновленими даними, після чого їх і відображають.

Реалізація MVVM і MVP-патернів, на перший погляд, виглядає досить простою та схожою. Однак, для MVVM зв'язування уявлення з View-моделлю здійснюється автоматично, а для MVP - необхідно програмувати. MVC, очевидно, має більше можливостей по управлінню поданням.

Оскільки патерн MVC дає змогу розділяти логіку, моделі та шаблони – реалізація підсистеми захисту спрощується, оскільки її реалізація буде проводитись у контролерах.

Для впровадження повноцінної системи захисту обов'язковим є використання захищеної СУБД.

Система управління базами даних (СУБД)[9] - це комплекс програмно-мовних засобів, що дозволяють створити бази даних і управляти даними. Іншими словами, СУБД - це набір програм, що дозволяє організовувати, контролювати і адмініструвати бази даних. Більшість сайтів не можуть функціонувати без бази даних, тому СУБД використовується практично повсюдно.

Основні функції СУБД:

- керування даними в зовнішній пам'яті (на дисках);
- керування даними в оперативній пам'яті з використанням дискового кешу;
- журналізациа змін (збереження історії), резервне копіювання і відновлення бази даних після збоїв;
- підтримка мов БД (мова визначення даних, мова маніпулювання даними).

Кожна СУБД ґрунтується на будь-якої моделі даних, це є одним з ознак класифікації. За моделлю даних СУБД бувають:

- Ієрархічні. У цій моделі даних використовується уявлення БД у вигляді дерева, що складається з даних різних рівнів.
- Мережеві. Дана модель є розширенням ієрархічного підходу. Ієрархічна модель має на увазі, що запис-нащадок може мати строго одного нащадка, в той час як в мережевій структурі нащадок може мати будь-яку кількість предків.
- Реляційні. СУБД, орієнтовані на організацію даних як набір пов'язаних записів і атрибутів в двовимірній таблиці.
- Об'єктно-орієнтовані. Для управління БД, заснованими на об'єктній моделі даних. Як правило ґрунтуються на об'єктно-орієнтованих мовах програмування.

- Об'єктно-реляційні. Об'єднує в собі концепції реляційної моделі з додатковими об'єктно-орієнтованими можливостями.

Сьогодні, як і раніше найбільш популярними при створенні веб-додатків і сервісів залишаються реляційні бази даних. Для керування базами даних використовується мова SQL (Structured Query Language – структурована мова запитів)[10]. Спочатку SQL був інструментом роботи користувача з базою даних, однак з часом мова ускладнився і став швидше інструментом розробника, ніж кінцевого користувача.

В основі реляційних систем лежить реляційна модель даних. Принципи реляційної моделі були закладені в 1969-1970 рр. американським вченим Е.Ф. Коддом (E.F. Codd), в той час працювали в корпорації IBM. Будучи математиком за освітою, він привніс в область управління базами даних суворі математичні принципи і точність, яких не вистачало раннім системам. Хоча реляційний підхід утвердився не відразу, можна відзначити, що майже всі створені з кінця 70-х рр. продукти баз даних засновані саме на реляційному підході. Переважна більшість наукових досліджень в області баз даних протягом останніх 35 років також проводилося саме в цьому напрямку.

Різні рейтинги найпопулярніших СУБД очолюють MySQL, Microsoft SQL Server, PostgreSQL, SQLite.

MySQL.

Вважається однією з найпоширеніших СУБД. MySQL - реляційна СУБД з відкритим вихідним кодом, головними плюсами якої є її швидкість і гнучкість, яка забезпечена підтримкою великої кількості різних типів таблиць. Крім того, це надійна безкоштовна система з простим інтерфейсом і можливістю синхронізації з іншими базами даних. У сукупності ці фактори дозволяють використовувати MySQL як великим корпораціям, так і невеликим компаніям.

Microsoft SQL Server

Як випливає з назви, фірмова СУБД, розроблена Microsoft. Оптимальна для використання в операційних системах сімейства Windows, проте може працювати і з Linux. Система дозволяє синхронізуватися з іншими

програмними продуктами компанії Microsoft, а також забезпечує надійний захист даних і простий інтерфейс, проте відрізняється високою вартістю ліцензії і підвищеним споживанням ресурсів. В цілому, однак, зберігає свою популярність, в чималому ступені через те, що продукти корпорації Microsoft використовуються багатьма компаніями.

PostgreSQL

СУБД PostgreSQL – ще одна популярна і безкоштовна система. Найбільше застосування знайшла для управління БД веб-сайтів і різних сервісів. Вона універсальна, тобто підійде для роботи з більшістю популярних платформ. При цьому PostgreSQL - об'єктно-реляційна СУБД, що дає їй деякі переваги над іншими безкоштовними СУБД, в більшості є реляційними.

SQLite

SQLite – це програмна бібліотека, яка реалізує автономний, безсерверний, транзакційний механізм бази даних SQL з нульовою конфігурацією. SQLite є одним з найбільш швидкозростаючих механізмів баз даних, але він росте з точки зору популярності, а не його розміру. Вихідний код для SQLite знаходиться у відкритому доступі.

Переваги та недоліки кожної СУБД представлено у таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки СУБД

СУБД	Переваги	Недоліки
Postgre	Високо функціональна і безкоштовна СУБД з відкритим кодом, хороша підтримка з боку співтовариства розробників	Місцями низька продуктивність при обробці великих обсягів інформації, мала популярність продукту
MS SQL	Високо функціональна, зручна при використанні середовища Windows	Платна, складна в адмініструванні

MySQL	Безкоштовна СУБД, має відкритий код, займає мало місця, легке адміністрування mysql з набором всього необхідного функціоналу, велика спільнота розробників, працює на безлічі платформ, високий рівень захисту	Посередня ефективність обробки даних
SQLite	Підтримка мови запитів версії SQL92 (без особливостей типу процедур) База даних зберігається в одиночному файлі на диску	Важко масштабується Не підходить для великих баз даних(медіатеки)

MySQL – це ефективна бібліотека, вбудована в додаток, який її використовує. Будучи серверною БД, вона надає відмінний набір інструментів для більш простої та захищеної обробки будь-яких видів даних. Коли додаток використовує MySQL, їх зв'язок проводиться за допомогою функціональних і прямих викликів файлів, що містять дані, а не якогось інтерфейсу, що підвищує швидкість і продуктивність операцій. Для максимально ефективної та швидкої розробки підсистеми захисту прийнято рішення використовувати саме MySQL.

Сама підсистема захисту безпосередньо стосується саме захисту користувачів та їх даних, завдяки двофакторній системі авторизації, що підвищує рівень безпеки даних не тільки користувачів, а й даних у БД, оскільки доступ до адміністративної панелі становиться більш захищеним. Додатковим

кроком у створенні підсистеми захисту є використання алгоритму pbkdf2 з хешем sha256, що розглядатиметься у розділі 2.4.

Структурно двофакторна підсистема захисту зображена на схемі(рис. 2.4):

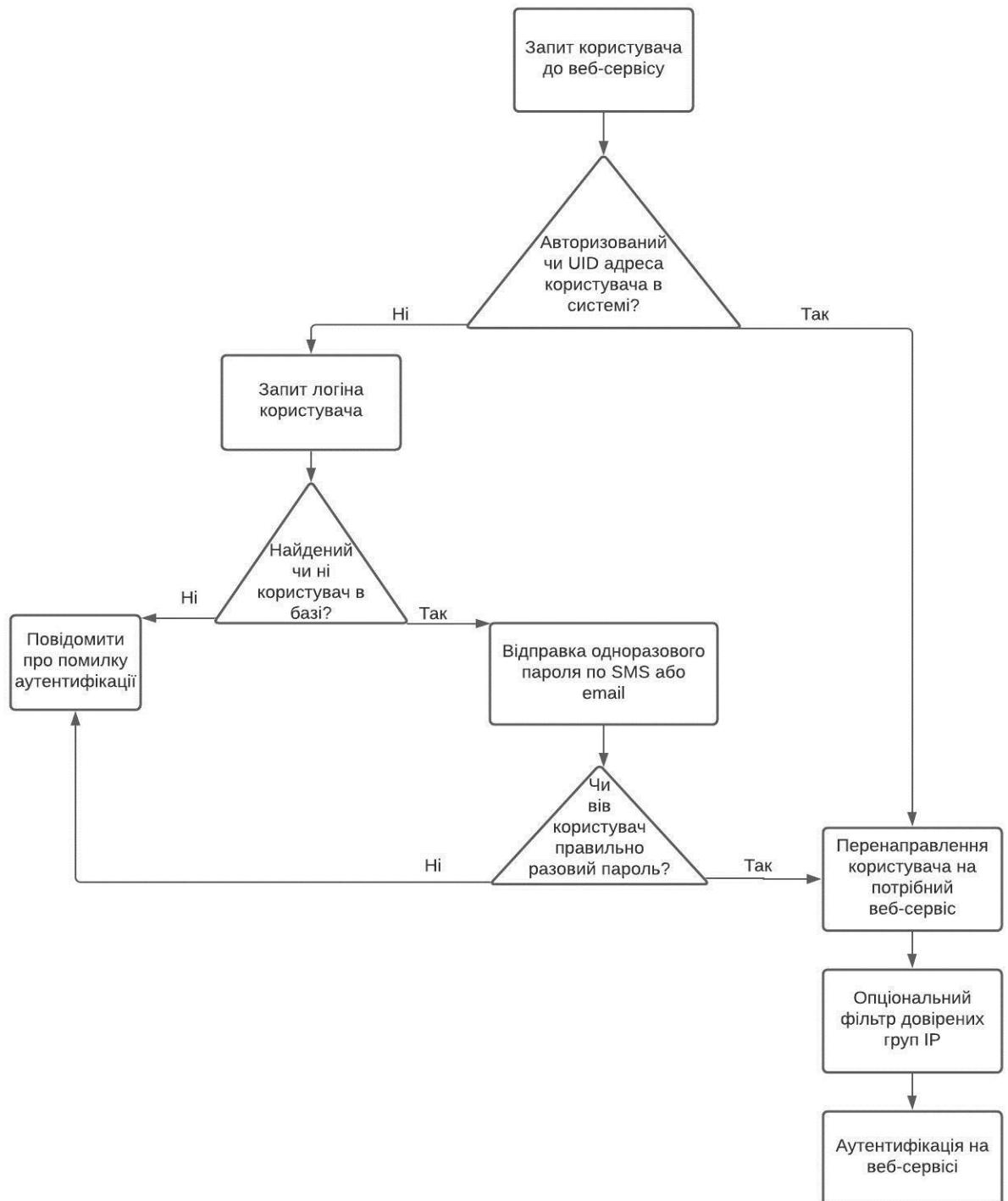


Рисунок 2.4 – Структура двофакторної підсистеми захисту

Дана система дозволить покращити показники захисту будь-якого веб-ресурсу, оскільки доступ зловмисників до даних користувачів стає більш складним, підвищується вірогідність цілісності даних.

2.2. Розробка інтернет-магазину

Основою будь-якого програмного забезпечення є мова програмування. У сучасності існує декілька категорій мов програмування, що значно відрізняються один від одного.

За 60 з гаком років розвитку ЕОМ були розроблені сотні мов програмування, багато з яких використовуються і зараз (наприклад Бейсік і Фортран були вперше застосовані вже в кінці 1950-х років), щорічно з'являється кілька нових мов промислового застосування (не рахуючи десятків експериментальних). Для того, щоб розібратися в них, мови класифікують за найважливішим ознаками:

- еволюційним - покоління мов (1GL, 2GL, 3GL, 4GL, 5GL ...);
- функціональним - за призначенням, виконуваним функціям (описові, логічні, математичні);
- рівню мови - тобто рівню узагальнення в словах-операторах мови (низького, середнього, високого ...);
- області застосування - тобто де застосовується мова (системні, мережеві, вбудовані та ін.

Базова ієрархія мов програмування є системно-паралельною ієрархією, тобто пакет тісно пов'язаних ієрархій: етапів програмування, поколінь мов програмування і самих мов - протоколів перетворення структурної та алгоритмічної інформації: структурно-дескриптивного опису даних і алгоритму їх обробки. Тому всі мови діляться на два полярних типу: дескриптивні (декларативні) і алгоритмічні (командні). Однак, так як в будь-якому алгоритмі існує необхідність опису даних і структур, і в будь-якій конструкції - порядок її

складання, то реальні мови є частково декларативними, й частково алгоритмічними, що відбивається в наявності описової та командної (рецептурної) частин будь-якої комп'ютерної програми.

Певна категорія мов програмування відрізняється від іншої за методами, які використовуються при написанні програм(див. рис. 2.5).

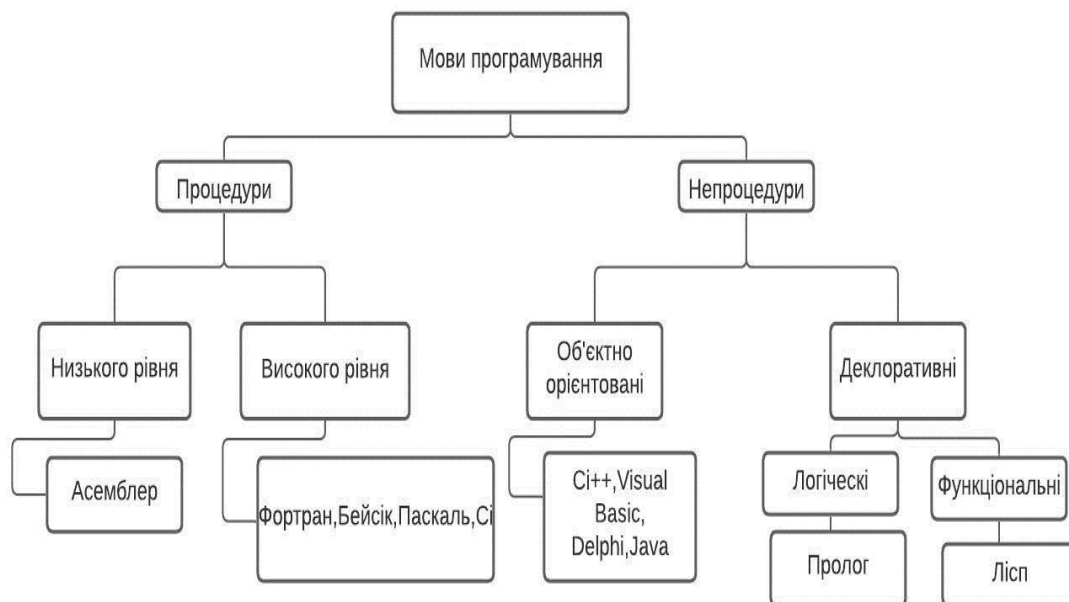


Рисунок 2.5 – Класифікація мов програмування

Існуючі мови програмування класифікують за чотирма основними функціональними групами: процедурні, об'єктно-орієнтовані, функціональні і логічні. Дамо короткі визначення кожного підходу.

Процедурне програмування має на увазі можливість створення в рамках програми локальних змінних, функцій і процедур. Будь-яка програма оперує даними і містить код для обробки цих даних. У мовах, що підтримують парадигму процедурного програмування, пріоритет залишається за кодом для обробки даних.

Декларативні мови програмування - До них відносяться функціональні і логічні мови програмування. Функціональне програмування-це спосіб складання програм, в яких єдиним дією є виклик функції. У функціональному програмуванні не використовується пам'ять, як місце для зберігання даних, а, отже, не використовуються проміжні змінні, оператори присвоювання і цикли.

Ключовим поняттям в функціональних мовах є вираз. Програма, написана на функціональній мові, являє собою послідовність опису функцій і виразів.

Об'єктно-орієнтоване програмування [11] – Методологія об'єктно-орієнтованого програмування (ООП) має на увазі уявлення всієї програми або її частин об'єктами. У кожного об'єкта є тип – в ООП він називається класом. Класи можна оголошувати або успадковувати і створювати з них екземпляри. Власне, об'єкт – це і є екземпляр класу. Зазвичай об'єкт об'єднує в собі дані і методи для роботи з ним.

Непроцедурні мови – сучасний напрямок системного програмування, що дозволяє сконцентрувати увагу розробника на описі цілей і правил, а не на послідовності дій по їх реалізації (тобто описується «що робити» замість «як робити»). Використання непроцедурного мови забезпечує максимально можливу простоту і зрозумілість програм для розробника-технолога, переносючи складності процедурної реалізації на системний рівень, що скорочує трудомісткість і терміни розробки, збільшує надійність ПО.

Програми, складені табличною мовою, зручно описують складні ситуації, що виникають при системному аналізі.

У сучасності найбільш популярними вважаються об'єктно-орієнтовані мови високого рівня. Розглянемо детальніше парадигму об'єктно-орієнтованого програмування задля визначення переваг її використання.

Парадигма ОПП лежить в основі багатьох популярних мов. ГО-мови: C++, C#, Java, JavaScript, Objective-C, Python[12], PHP, Perl, Ruby, Scala, Swift та ін. Тобто топові мови, ряд мов загального призначення, мови для Android, веб і iOS розробки. Популярні сучасні мови – використовують мультипарадигмальність, по принципах ООП можна писати і на мовах з іншими парадигмами.

ООП як стиль написання програм передбачає побудову структури, що складається з взаємодіючих об'єктів. В рамках ООП потрібно мислити об'єктами. Об'єкти розташовані в ієрархії, самостійні і якимось чином взаємодіють.

Програма складається з модулів - блоків, які вирішують якісь завдання. Зміни в цих ділянках можуть не відбиватися на інших ділянках.

Об'єкти можуть бути абстрактними сутностями в програмі, а можуть уособлювати об'єкти фізичного середовища. Бізнес-процеси теж можна уявити як взаємодію і стану об'єктів. Можна зустріти такі аналогії, як виробництво, пристрій і водіння автомобіля, пристрій міста або будинку та ін. В світі ми всюди бачимо класи об'єктів, і об'єкти якимось взаємодіють в ієрархії. Очевидними перевагами ООП є:

- Можливість легкої модифікації (при грамотному аналізі і проектуванні)
- Можливість відкату при наявності версій
- Більш легка розширюваність «Більш природна» декомпозиція програмного забезпечення, яка істотно полегшує його розробку.
- Скорочення кількості міжмодульних викликів і зменшення обсягів інформації, що передається між модулями.
- Збільшується показник повторного використання коду.

Лідерами серед об'єктно-орієнтованих та за сумісництвом веб-орієнтованих мов заслужено вважаються PHP, Java, Python. Проведемо порівняльний аналіз задля визначення найкращого претенденту на розробку підсистеми захисту сайту інтернет-магазину.

Python та Java.

Інтерпретований і динамічно типізований Python проти компільованої і статично типізованої Java. І найперше спостереження: набір інструкцій на Python не потрібно компілювати перед запуском, в той час як програма, написана на Java, повинна бути трансльована в байт-код JVM, перш ніж віртуальна машина інтерпретує його в машинний код. Простіше кажучи, Python швидше запускається і довше виконується, коли для Java, навпаки, характерно більш тривалий час запуску, проте більш швидкий час виконання.

Розробка на Python, поза всякими сумнівами, дешевше, ніж такий на Java, де зазвичай потрібно чималі, як тимчасові, так і матеріальні витрати. Скажімо

					КБ.КВ.Б – 125 – 04– ПЗ	41

так, якщо є багато грошей, Java підійде ідеально. В іншому випадку, вибір Python виявиться більш раціональним рішенням.

Будь то штучний інтелект або машинне навчання, внутрішній устрій, філософія і функціональність Python-а дають йому перевагу перед усіма іншими мовами в цих відносно нових ІТ-областях. Крім того, є всі ознаки того, що ця тенденція збережеться і в майбутньому.

Резюмуючи, можна відзначити, що Python легкий у вивченні і простий в роботі. Якщо на чолі стоїть швидкість розробки, Python буде відмінним вибором. Але Java, проте, все ще ідеально підходить для вирішення дійсно складних завдань і краще за всіх інших мов задовольняє вимогам великого бізнесу.

Python та PHP.

PHP - широко використовуваний мову з відкритим вихідним кодом, який забезпечує роботу близько 80% серверів сайтів. Новачкові дуже легко освоїти PHP, і він пропонує безліч зручних функцій і рішень для веб-розробки.

Python, в даний час, конкурує з PHP в цій області, при тому, що в бізнес-процеси не є його пріоритетних напрямів. Більш того, діапазон використання мови Python досить широкий і все ще продовжує рости. Він може бути успішно застосований в машинному навчанні, аналізі даних, статистики та навіть наукових дослідженнях. З точки зору універсальності, він сильно випереджає PHP.

Оскільки версій Python-а відчутно менше, ніж модифікацій PHP, Python є більш організованим, безпечним і простим в обслуговуванні.

З огляду на можливості застосування Python в широкому спектрі галузей, що виходять за рамки можливостей PHP, мова зі змінним логотипом переживає величезний сплеск популярності, в той час як любов і ступінь використання програмістами PHP помітно знизилися і продовжують йти на спад.

Якщо потрібно розробити веб-сайт, блог або простий веб-сервіс, то особливих відмінностей між Python і PHP з точки зору продуктивності, швидкості роботи або призначеного для користувача дизайну не знайти. Однак,

якщо концепція проекту більш різноманітна і включає в себе, наприклад, технології машинного навчання або аналіз даних, то Python виявиться найбільш адекватним варіантом. Крім того, час життя мови Python дуже ймовірно перевершить таке для PHP, адже за порівняльним графіком(див. рис. 2.6) очевидний спад популярності PHP, що з часом унеможливить оновлення та модифікацію додатків, написаних за допомогою PHP.

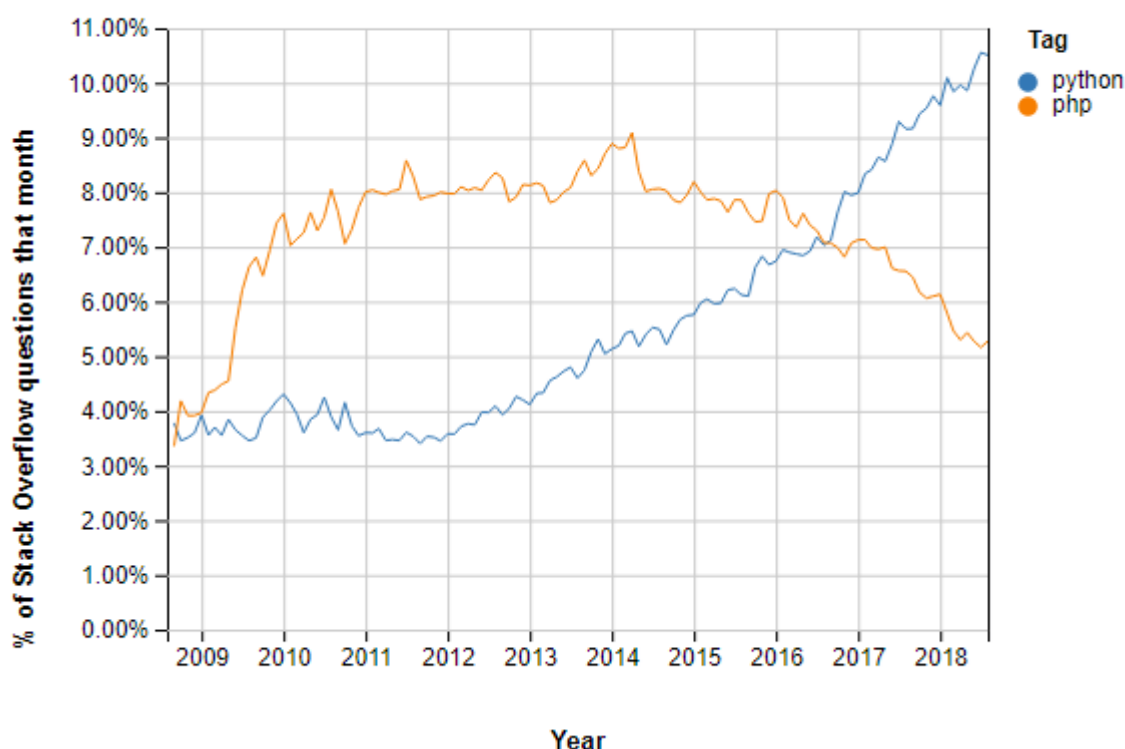


Рисунок 2.6 – Графік порівняння популярності PHP та Python

Наведений вище порівняльний аналіз доводить, що найбільш актуальною та ефективною мовою для розробки сучасних веб-сайтів є Python. Якщо розглядати можливості Python у веб-розробці то існує два найпопулярніших веб-фреймворки – Django та Flask. Згідно з даними опитування розробників Python в 2019, Django і Flask є найпопулярнішими веб фреймворками серед розробників.

Flask і Django – обидва великі, розширювані веб фреймворки, які, в корені своєму, надають аналогічний функціонал в обробці запитів, підтримки документів, але розрізняються в масштабі відповідальності. Велика частина відмінностей між двома фреймворками є наслідком різних підходів, інші - з

відмінних основних проектних рішень. Нижче представлено невеликий список ключових відмінностей:

- Об'єкт Request - Flask використовує локальні потоки, а Django передає запит там, де це потрібно.
- Форми - Django доступний з вбудованими формами, які інтегруються з ORM і адмінкою сайту.
- Flask не підтримує форми за замовчуванням, але ви можете використовувати WTForms, щоб заповнити цю прогалину.
- Бази даних – Django доступний з вбудованою ORM, яка використовує патерн MVC і системою міграції, яка може керувати базами даних. Flask не може цим похвалитися, проте є інструменти, такі як SQLAlchemy, які надають аналогічний функціонал (або навіть більше).
- Аутентифікація і привілеї користувачам – Django надає додаток аутентифікації, яке надає реалізацію за замовчуванням для призначеного для користувача керування і привілеїв, та алгоритм PBKDF2 з хешем SHA256 для захисту даних. Flask надає безпечні куки в якості інструменту вашої власної реалізації.
- Панель адміністратора – Django включає в себе повністю інтегрований адмін-інтерфейс для управління даними додатка. Flask не має таких функцій, але Flask-Admin - дуже популярне розширення, яке можна використовувати для створення аналогічного адміністративного інструменту.

Використання Django дозволяє просто та швидко створювати веб-додатки, а Flask дозволяє більш чітко налаштовувати конфігурацію додатку, проте знижує швидкість розробки додатку. Django має велику кількість додаткових бібліотек, що зменшує потребу написання власного коду, що в свою чергу підвищує ефективність розробки. Найголовнішою перевагою Django є ORM[13], що спрощує взаємодію з базою даних.

ORM містить інструмент для автоматичної генерації і управління міграціями баз даних. Використовуючи надані команди управління, можна значно просунутися, змінюючи визначення вашої Django Model, при цьому

велика частина міграцій баз даних управляється розробником: існує ряд ситуації, в яких автоматично згенеровані міграції можуть значно допомогти.

Django ORM можна назвати самовпевненим інструментом, в якому є певні обмеження, але його можна налаштовувати, розширювати і в кінцевому рахунку можна отримати чистий SQL, якщо знадобитися обійти ці обмеження.

Flask принципово не включає в себе ніяких модулів для управління базами даних. За філософії Flask, кожен сам вибирає який саме модуль для управління базами даних йому потрібен і не веб-фреймворку це вирішувати.

Задля максимально ефективною розробки вирішено використовувати веб-фреймворк Django.

У сучасності всі веб-сторінки створюються за допомогою мови гіпертекстової розмітки – HTML 5, та таблиці каскадних стилів – CSS 3, що і будуть використовуватись при розробці веб-орієнтованих систем.

У таблиці 2.2 наведено повний список інструментальних засобів реалізації системи захисту сайту інтернет-магазину

Таблиця 2.2 – Список інструментальних засобів реалізації системи захисту сайту інтернет-магазину

№	Компонент	Обраний інструмент
1	СУБД	SQLite 3
2	Мова програмування	Python(Django)
3	Патерн проектування	MVC
4	Парадигма програмування	ООП
5	Зовнішній вигляд сторінок	HTML/CSS

Завдяки архітектурі MVC, якої дотримується Django, структура проекту поділяється на views.py(Контролери), models.py(Моделі), та templates(Відображення), що дозволяє легко обробляти об'єкти у БД.

Структура розробленого проекту виглядає наступним чином(рис. 2.7-2.8):

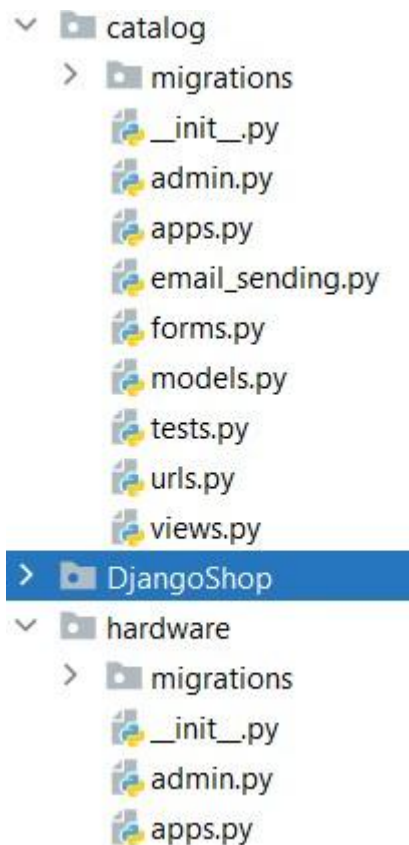


Рисунок 2.7 – Структура розробленого проекту.

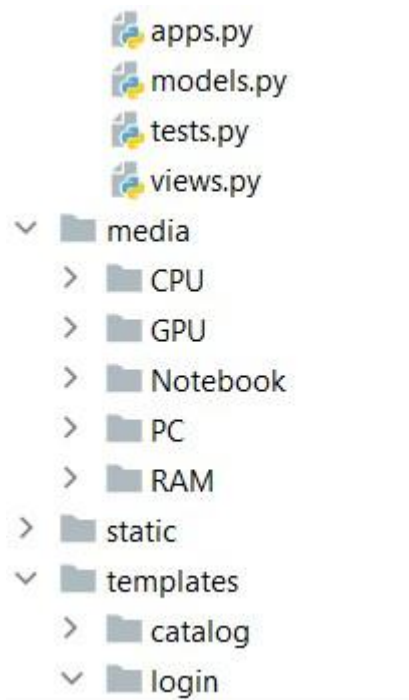


Рисунок 2.8 – Структура розробленого проекту.

Для покращення структуризації системи використовуючи Django прийнято поділяти логічні елементи додатку на apps(додатки), що використано і у даному випадку, «catalog» - структурний додаток, що створено для реалізації

моделей, представлень та налаштувань маршрутизації для системи інтернет-магазину. Даної логіки слід дотримуватись задля того, щоб у майбутньому, при спробі розширити проект мати змогу безпечно створювати нові додатки, не чіпаючи вже існуючі.

Першим кроком для роботи з Django є файл налаштувань settings.py(додаток А), що містить налаштування проекту, такі як: часова зона, налаштування підключення до бази даних, встановлені модулі і т.п.

Наступним кроком є створення додатку «catalog», що створюється завдяки команді manage.py startapp «назва». Після даних маніпуляцій, структура проекту виглядає як на рис. 2.7.

Кожен додаток має власні налаштування маршрутизації, що зберігаються у файлах під назвою urls.py та мають наступний вигляд:

```
urlpatterns = [
    path('', views.IndexView.as_view(), name='index'),
    path('login/', views.SigninView.as_view(), name='login'),
    path('signup/', views.SignupView.as_view(), name='signup'),
    path('logout/', views.Logout.as_view(), name='logout'),
    path('ram/', views.RAMListView.as_view(), name='ram_list'),
    path('cpu/', views.CPUListView.as_view(), name='cpu_list'),
    path('gpu/', views.GPUListView.as_view(), name='gpu_list'),
    path('pc/', views.PCListView.as_view(), name='pc_list'),
    path('notebook/', views.NotebookListView.as_view(), name='notebook_list'),
    path('ram/<str:slug>', views.RAMDetailView.as_view(), name='ram'),
    path('cpu/<str:slug>', views.CPUDetailView.as_view(), name='cpu'),
    path('gpu/<str:slug>', views.GPUDetailView.as_view(), name='gpu'),
    path('pc/<str:slug>', views.PCDetailView.as_view(), name='pc'),
    path('notebook/<str:slug>', views.NotebookDetailView.as_view(),
name='notebook'),
    path('search/', views.SearchView.as_view(), name='search'),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Для обробки URL адрес використовуються контролери(views.py), повний лістинг файлу представлений в додатку В.

```
class SignupView(generic.CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'login/signup.html'

class CPUListView(generic.ListView):
    template_name = 'catalog/cpu_list.html'
    queryset = CPU.objects.all()
```

```

context_object_name = 'cpus'

def get_context_data(self, *, object_list=None, **kwargs):
    context = super(CPUListView, self).get_context_data()
    return context

def get_queryset(self):
    title = self.request.GET.get('title')
    products = []
    if title:
        if CPU.objects.filter(title__contains=title):
            products.append(CPU.objects.filter(title__contains=title))
        if GPU.objects.filter(title__contains=title):
            products.append(GPU.objects.filter(title__contains=title))
        if RAM.objects.filter(title__contains=title):
            products.append(RAM.objects.filter(title__contains=title))
        if PC.objects.filter(title__contains=title):
            products.append(PC.objects.filter(title__contains=title))
        if Notebook.objects.filter(title__contains=title):
            products.append(Notebook.objects.filter(title__contains=title))
    return products

def get_context_data(self, *, object_list=None, **kwargs):
    context = super(SearchView, self).get_context_data()
    context['title'] = self.request.GET.get('title', None)
    return context

```

Дані контролери є фактично обробниками моделей, що видають шаблону динамічні дані, запитувані користувачем завдяки Django ORM.

Оскільки для кожної системи важлива структуризація даних за допомогою баз даних, Django ORM надає можливість створювати моделі(таблиці) даних за допомогою класів з полями-атрибутами, повний лістинг файлу представлений в додатку Б.

```

class AbstractProduct(Model):
    title = CharField(_('Product title'), max_length=100)
    image = ImageField(_('Product image'), upload_to=upload_path)
    description = TextField(_('Product description'), max_length=1000,
default='')

```

```

manufactured = IntegerField(
    _('Year of manufacture'),
    validators=[MinValueValidator(1), MaxValueValidator(CURRENT_YEAR)]
)
slug = AutoSlugField(db_index=True, primary_key=True, populate_from='title')
manufacturer = ForeignKey(Manufacturer, on_delete=CASCADE)
posted = DateTimeField(_('Post time'), auto_now=True)
price = IntegerField(_('Product price'), validators=[MinValueValidator(1)])
sale_price = IntegerField(_('Sale price'),
validators=[MinValueValidator(1)], null=True, blank=True)
sale_active = BooleanField(_('Activate sale'), default=False)
link = URLField(verbose_name='Де купити', null=True, default='#')

def __str__(self):
    return f"{self.title}"

def get_current_price(self):
    if self.sale_active:
        return self.sale_price
    return self.price

class Meta:
    abstract = True

```

Відображення сторінок відбувається завдяки HTML документам, повний перелік яких представлено у додатку Б.

Адміністративна панель у Django створюється автоматично, завдяки файлам з відповідною назвою – admin.py, що реєструють моделі у адміністративній панелі:

```

@admin.register(GPUBus)
class GPUBusAdmin(admin.ModelAdmin):
    pass

@admin.register(PC)
class PCAdmin(admin.ModelAdmin):
    pass

```

2.3. Розробка двофакторної системи захисту

Задля впровадження двофакторної системи захисту потрібно модифікувати LoginView та SignupView, що забезпечують реєстрацію та авторизацію.

Алгоритм двофакторної системи автентифікації можна розділити на такі складові:

1. Користувач вводить дані для автентифікації
2. Проводиться перевірка валідності введених даних
3. Якщо дані валідні, користувач отримує на пошту письмо, що містить унікальний ключ, який користувач має ввести на сторінці підтвердження другого етапу автентифікації.
4. Якщо код введено вірно – користувач авторизований.
5. Якщо код введено невірно – користувач деавторизований.

Даний алгоритм використовує доступ до найпопулярнішого поштового сервісу у світі – Gmail, завдяки Google API.

Реалізація даного алгоритму передбачає модифікацію LoginView, а саме – виклик методу `send_email`, створення об'єкту коду підтвердження що посилається на користувача, при валідності даних у формі та переадресація на сторінку введення коду, повний лістинг класу LoginView представлений в додатку В.

```
def form_valid(self, form):  
    code = randint(1000000, 9999999)  
    Code.objects.create(user=form.cleaned_data['username'], code=code)  
    send_mail(form.cleaned_data['username'], code=code)  
    return super(SigninView, self).form_valid(form)
```

Метод відправки email виглядає наступним чином:

```
import smtplib # Импортируем библиотеку по работе с SMTP  
  
from email.mime.multipart import MIMEMultipart # Многокомпонентный объект  
from email.mime.text import MIMEText # Текст/HTML  
  
def send_mail(email_to: str, code: str):  
    addr_from = "пошта" # Адресат  
    password = "пароль" # Пароль  
    addr_to = email_to # Получатель
```

```

msg = MIMEMultipart() # Створення повідомлення
msg['From'] = addr_from # Адресат
msg['To'] = addr_to # Отримувач
msg['Subject'] = 'CONFIRMATION CODE' # Тема повідомлення

body = f"ENTER THIS CODE: {code}!"
msg.attach(MIMEText(body, 'plain')) # Додаємо в повідомлення текст

server = smtplib.SMTP('smtp.gmail.com', 587) # Створення об'єкту SMTP
server.starttls() # Починаємо шифрований обмін по TLS
server.login(addr_from, password) # Отримуємо доступ
server.send_message(msg) # Відправляємо повідомлення
server.quit()

```

Сторінка перевірки коду визначає, чи вірно введено код, та якщо дані валідні, перенаправляє користувача на головну сторінку. Якщо дані невірні – користувача деавторизовано повний лістинг коду представлений в додатку В:

```

def form_valid(self, form):
    if form.cleaned_data['code'] ==
Code.objects.get(user=self.request.user).code:
        Code.objects.get(user=self.request.user).delete()
        return super(SigninCodeView, self).form_valid(form)
    Code.objects.get(user=self.request.user).delete()
    return super(SigninCodeView, self).form_invalid(form)

```

2.4. Алгоритм pbkdf2 з хешем sha256

Функція отримання ключа на основі паролю PBKDF2 (Password-Based Key Derivation Function Version 2) - це функція отримання ключа, розроблена RSA Laboratories, яка використовується для отримання стійких ключів на основі хеша. Вона працює шляхом застосування псевдослучайной хеш-функції (на кшталт SHA-256, SHA-512, і т. Д.) До рядка, в нашому випадку - до паролю, разом з сіллю і повторенням цього процесу велике число раз. Даний процес може бути узагальнено наступною діаграмою(рис. 2.8):

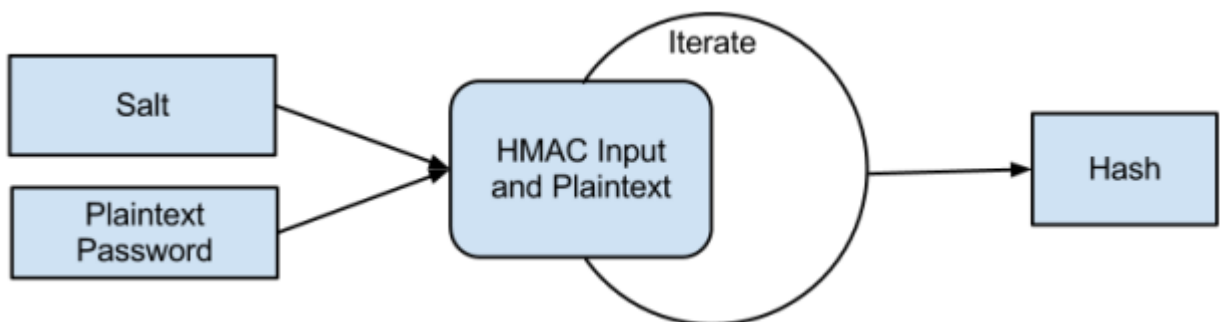


Рисунок 2.9 – Алгоритм PBKDF2

При використанні цього алгоритму хешується сіль і відкритий текст для отримання першого хеша, потім в циклі той же алгоритм використовується для обчислення хеша від відкритого тексту і результату попередньої ітерації, після чого повертається результат застосування операції XOR до всіх обчислених хешів. Виконавши на серверній стороні дану операцію 100000 (як стверджують LastPass) або більше разів, згенерується сильний, стійкий до злому хеш, який можна безпечно використовувати для зберігання паролів.

Висновки по 2 розділу

У даному розділі проведено роботу над розробкою сайту інтернет-магазину та підсистемою його захисту. Проведено порівняльний аналіз патернів проектування та виділено переваги та недоліки кожного з них, виділено патерн MVC, прийнято рішення дотримуватись саме цього патерну.

Проведено аналіз існуючих СУБД, приведені переваги та недоліки таких СУБД як PostgreSQL, MySQL, SQLITE, MsSQL Server.

Прийнято рішення використовувати MySQL, оскільки дана СУБД вважається найбезпечнішою.

Приведено структурну схему додатку, проведено аналіз існуючих сучасних мов програмування. Виділено мову Python, фреймворк Django.

Розроблено додаток інтернет-магазину за допомогою виділених інструментів.

Проведено аналіз двофакторної системи захисту, алгоритму PBKDF2, впроваджено дані системи у додаток сайту інтернет-магазину.

РОЗДІЛ 3. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ПІДСИСТЕМИ ЗАХИСТУ САЙТУ ІНТЕРНЕТ-МАГАЗИНУ

3.1. Розгортання моделі експерименту

Перед початком роботи з Django у першу чергу слід встановити інтерпретатор Python версії 3.8 або вище. Для його встановлення потрібно перейти на сторінку <https://www.python.org/downloads/> та знайти посилання на завантаження останніх версій мови.

Після натискання на кнопку буде завантажено відповідну поточну ОС майстра установки Python.

Тут можна задати шлях, по котрому буде встановлений інтерпретатор. Можна залишити за замовчуванням `C: \ Users \ [ім'я_користувача] \ AppData \ Local \ Programs \ Python \ Python36 \`. Крім того, у самому низу потрібно відзначити флажок "Додати Python 3.9 до PATH", щоб додати шлях до інтерпретатора в змінних середовища.

Існують різні способи встановлення Django. Розглянемо рекомендований спосіб. Для установки буде потрібен пакетний менеджер `pip`. Менеджер `pip` дозволяє завантажувати пакети і управляти ними. Нерідко, при установці `python` також встановлюється і менеджер `pip`. У цьому випадку можна перевірити версію менеджера, виконавши в командному рядку / терміналі наступну команду:

```
Pip -v
```

Але якщо `pip` не встановлено, то побачимо помилку типу "`pip`" не є внутрішньою або зовнішньою командою, що виконується програмою або пакетним файлом. У цьому випадку треба встановити `pip`.

Віртуальне середовище або `virtualenv`[14] не є невід'ємною частиною розробки на Django. Однак її рекомендується використовувати, так як вона дозволяє створити безліч віртуальних середовищ Python на одній операційній системі. Завдяки віртуальному середовищі додаток може запускатися незалежно від інших додатків на Python

					КБ.КВ.Б – 125 – 04– ПЗ	53

В принципі можна запускати додатки на Django і без віртуального середовища. В цьому випадку всі пакети Django встановлюються глобально.

Для роботи з віртуальним середовищем спочатку необхідно встановити пакет virtualenv за допомогою наступної команди

```
Pip install virtualenv
```

Тепер створимо віртуальну середу. Спочатку визначимо каталог для віртуальних середовищ, де будуть розташовуватися всі пов'язані файли і папки. Наприклад, нехай це буде каталог C: \ virtualenv. Перш за все перейдемо в командному рядку/ терміналі в цей каталог за допомогою команди cd. Потім для створення віртуального середовища виконаємо наступну команду:

Virtualenv Назва

Для використання віртуальне середовище треба активувати. І кожен раз, коли ми будемо працювати з проектом Django, пов'язану з ним віртуальне середовище треба активувати. Наприклад, активуємо вище створене середовище, яка розташовується в поточному каталозі в папці hello. Якщо наша ОС - Windows, то в папці hello / Scripts / ми можемо знайти файл activate.bat, який активує віртуальне середовище. Тому для Windows активація віртуального середовища буде виглядати таким чином:

```
hello\Scripts\activate.bat
```

Для Linux і MacOS активація буде проводитися за допомогою наступної команди:

```
source ~/.hello/bin/activate
```

Після закінчення роботи з віртуальним середовищем ми можемо її деактивувати. Для цього в тій же папці hello / Scripts / ми можемо знайти файл deactivate.bat і таким же чином запусити його.

Після активації віртуального середовища для установки Django виконаємо в консолі наступну команду

```
Pip install Django
```

Дана команда є фінальним кроком для встановлення Django.

Після цього потрібно перейти в директорію даного проєкту, та виконати команду

`Python manage.py runserver`

Дана команда запустить локальний сервер і дасть змогу використовувати даний проєкт за адресою 127.0.0.1:8000/.

Для доступу потрібна авторизація яка буде виглядати наступним чином(рис. 3.2):

The image shows a web login form for Grappelli. It has a dark header with the word 'Grappelli' in white. Below the header is a blue bar with the text 'Log in'. Underneath is a light gray box containing two input fields: 'Username' and 'Password'. To the right of the 'Password' field is a blue button with the text 'Log in'.

Рисунок 3.1 – Сторінка авторизації

Оскільки жодного користувача ще не створено, потрібно виконати команду

`Python manage.py createsuperuser`

Дана команда викликає скрипт створення супер-користувача(адміністратора), що виглядає наступним чином(рис. 3.3):

```
C:\Users\Bodja\Desktop\4 курс\DjangoShop-master>python manage.py createsuperuser
Username: admin
Email address:
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
C:\Users\Bodja\Desktop\4 курс\DjangoShop-master>
```

Рисунок 3.2 – Створення адміністратора за допомогою командного рядка Windows 10.

Після даного кроку потрібно авторизуватись. На даному етапі система захисту сайту інтернет-магазину та сам інтернет-магазин готові до використання.

3.2. Дослідження та порівняння налаштувань безпеки сайту при різних вразливостях

Першим кроком варто перевірити працездатність алгоритму шифрування паролей PBKDF2 з хешем sha256, що автоматично впроваджено Django.

Переглядаючи облікові записи у адміністративній панелі очевидно, що паролі зашифровані(рис. 3.3):

Username	admin
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.	
Password	algorithm: pbkdf2_sha256 iterations: 216000 salt: UQxkB7***** hash: DfS7zh***** Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form .

Рисунок 3.3 – Алгоритм PBKDF2 у системі Django

Перевіримо роботу двофакторної системи захисту.

Першим кроком, зареєструємо користувача(рис. 3.4):

Регістрація нового користувача

Username*
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
Password*
- Your password can't be too similar to your other personal information. - Your password must contain at least 8 characters. - Your password can't be a commonly used password. - Your password can't be entirely numeric.
Password confirmation*
Enter the same password as before, for verification.
Зареєструватися

Рисунок 3.4 – Процес реєстрації користувача

Зареєстровано користувача з логіном testmaildjangoshop@gmail.com, після натискання на кнопку «Реєстрація», при успішному введенні даних користувача переадресовано до сторінки авторизації(рис. 3.5):

					КБ.КВ.Б – 125 – 04– ПЗ	
						56

Username:

Password:

Вхід

Рисунок 3.5 – Авторизація користувача(Перший етап)

Після коректного введення даних користувач отримує на пошту письмо з кодом(рис. 3.6), який потрібно ввести на наступній сторінці(рис. 3.7):

CONFIRMATION CODE **Входящие x**

bbfallen.work@gmail.com

кому: я ▼

ENTER THIS CODE: 7643337!

Рисунок 3.6 – Письмо з кодом на пошті

Введіть код з письма на пошті:

Введіть код:

Увійти

Рисунок 3.7 – Авторизація користувача(Другий етап)

Якщо код введено вірно, користувача переадресовано на головну сторінку.

Перевіримо роботу підключеної бібліотеки AXES. Вона працює з налаштуваннями за замовчуванням і зберігає спроби користувачів у вашій базі даних та блокує користувачів, якщо вони перевищують максимальну кількість спроб. Першим кроком спробуємо декілька раз авторизуватися з неправильним паролем імітуючи атаку брутфорс (рис 3.8)

Рисунок 3.8 - Імітація атаки грубої сили.

Після трьох спроб ми отримуємо блокування акаунта користувача та IP-адреси (рис 3.9).



Рисунок 3.9 - Результат атаки грубої сили.

Спробуємо провести атаку типу XSS. Для цього в формі введемо скрипт, який у разі спрацювання виведе на екран 'test', результат атаки представлений на (рис 3.10).

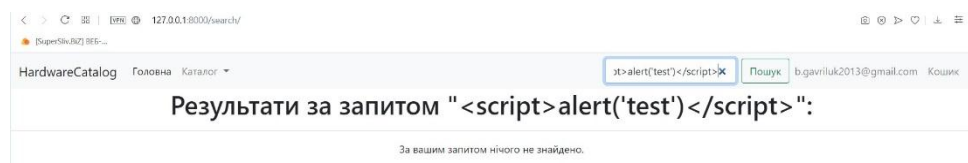


Рисунок 3.10 - Результат атаки типу XSS.

Також реалізуємо атаку XPath-інєкція .Результат наведений на (рис 3.11).

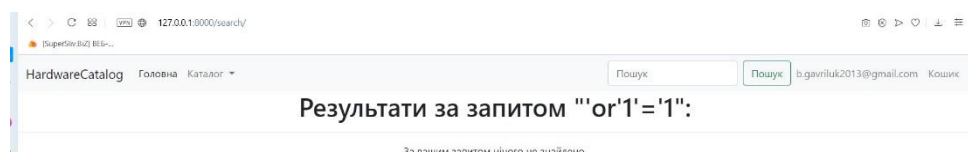


Рисунок 3.11 - Результат атаки XPath.

Спробуємо перехопити get запит за допомогою програми Burp Suite Community на сервер при авторизації та при відправці коду підтвердження, результати перехопленого запиту наведений на (рис 3.12).

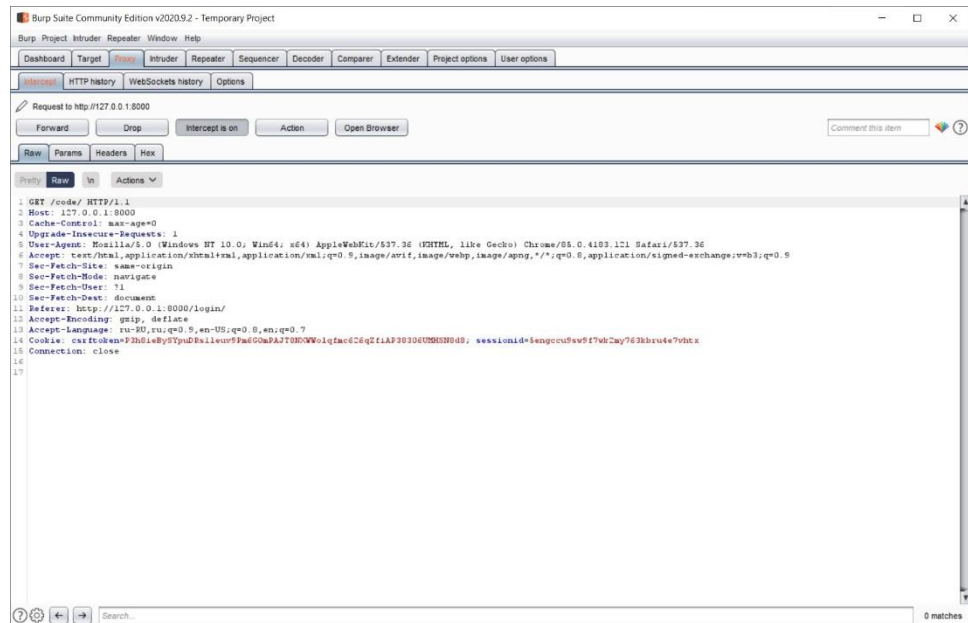


Рисунок 3.12 - Перехопления get запроса на сервер.

3.3. Рекомендації по застосуванню розробленої підсистеми захисту сайту інтернет-магазину

Розроблена система та її подібні аналоги повинні(!) застосовуватись у масштабних веб-ресурсах, наприклад український інтернет-магазин Rozetka.ua успішно застосовує схожу підсистему захисту.

Дана система має декілька очевидних переваг, що покращують захист даних користувачів та даних системи взагалі:

- Покращений захист паролів користувачів, що унеможливляють злом акаунтів зловмисниками.
- Покращений захист даних всієї системи, оскільки паролі адміністраторів зашифровані в тому числі.
- Зловмисник не матиме доступу до акаунту не маючи доступу до пошти користувача.

Розроблена система може бути легко модифікована та застосована у будь-якому веб-додатку. Можливі модифікації системи:

- Збільшення кількості ітерацій алгоритму PBKDF2
- Збільшення довжини та псевдовипадковості коду підтвердження
- Використання більш надійної робочої пошти

Висновки по 3 розділу

В даному розділі наведена перевірка працездатності розробленої підсистеми захисту. Створено інструкцію розгортання розробленого сайту інтернет-магазину та підсистеми його захисту, наведено приклад розгортання. Проведено дослідження та порівняння налаштувань безпеки сайту при різних властивостях, а саме – перевірено системи двофакторної автентифікації та шифрування паролів алгоритмом PBKDF2.

Наведено практичні рекомендації щодо впровадження розробленої підсистеми захисту.

Перевірка показала захищеність додатку до атаки грубої сили, XSS - атаки, XPath, та перехоплення get – запиту.

ВИСНОВКИ

У випускній кваліфікаційній роботі було виконане актуальне завдання по розробці підсистеми захисту сайту інтернет-магазину.

У першому розділі проведено аналітичну роботу над доведенням актуальності теми роботи, а саме розробки підсистеми захисту сайту інтернет-магазину. Виділено види загроз, що існують у сучасності, приведено приклади та статистику жертв одного з найпоширенішого виду атак – DDoS.

Виділено методи захисту веб-сайтів, розглянуті переваги та недоліки таких методів, як:

- Web Application Firewall (WAF).
- Засоби аналізу веб-сайтів на наявність вірусів.
- Балансувальник навантаження на веб-додатки.
- Засоби захисту від DDoS.
- Сканери захищеності веб-додатків (WASS).

Проведено аналіз існуючих систем захисту веб-сайтів, наведено конкретні приклади.

У другому розділі проведено роботу над розробкою сайту інтернет-магазину та підсистемою його захисту. Проведено порівняльний аналіз патернів проектування.

Проведено аналіз існуючих СУБД, приведені переваги та недоліки таких СУБД як PostgreSQL, MySQL, SQLITE, MsSQL Server.

В останньому розділі наведена перевірка працездатності розробленої підсистеми захисту. Створено інструкцію розгортання розробленого сайту інтернет-магазину та підсистеми його захисту, наведено приклад розгортання. Проведено дослідження та порівняння налаштувань безпеки сайту при різних властивостях, а саме – перевірено системи двофакторної автентифікації та шифрування паролів алгоритмом PBKDF2.

Наведено практичні рекомендації щодо впровадження розробленої підсистеми захисту.

Перевірка показала захищеність додатку до атаки грубої сили, XSS - атаки, XPath, та перехоплення get – запиту.

При створенні проекту виділено патерн MVC, прийнято рішення дотримуватись саме цього патерну, як використовуваної СУБД вибір зроблений на користь MySQL, а в якості мови програмування для розробки програми доступу до даних обраний Python, фреймворк Django. Це пов'язано з тим, що SQLite дозволяє ефективно обробляти невеликий обсяг даних.

При розробці інформаційної системи, був використаний для користувача інтерфейс за стандартом GUI. Інформаційна система є відкритою і має великий потенціал свого розвитку. Вона може змінюватися при зміні вимог, що пред'являються до предметної області. Розглянуто питання безпеки проекту. Розглянутий двох факторний захист користувачів, а паролі захищаються алгоритмом PBKDF2 з хешем SHA256.

Розроблена інформаційна система не претендує на комерційний повноцінний додаток, а створена у якості наукового дослідження. Розроблена система підлягає модифікації, видозміні та дозволяється до використання у подальших розробках та дослідженнях.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бабаш, А.В. Информационная безопасность: Лабораторный практикум / А.В. Бабаш, Е.К. Баранова, Ю.Н. Мельников. - М.: КноРус, 2019. - 432 с.
2. Кузин, А.В., Демин, В.М. Компьютерные сети: Учебное пособие. - М.: Форум: ИНФРА-М, 2005. - 192 с.
3. Запечников, С.В. Информационная безопасность открытых систем. В 2-х т. Т.1 - Угрозы, уязвимости, атаки и подходы к защите / С.В. Запечников, Н.Г Милославская. - М.: ГЛТ, 2006. - 536 с.
4. NETSCOUT. [Электронный ресурс]. URL: <https://www.netscout.com/threatreport>
5. The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. [Электронный ресурс]. URL: <https://owasp.org/>
6. Партыка, Т.Л. Информационная безопасность: Учебное пособие / Т.Л. Партыка, И.И. Попов. - М.: Форум, 2018. - 88 с.
7. Фостер, Дж.С. Защита от взлома: сокет, эксплойты, shell-код: выявление уязвимостей операционных систем и прикладных программ к атакам хакеров / Дж.С. Фостер. - М.: ДМК, 2006. - 784 с.
8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. -- СПб: Питер, 2001. -- 368 с.: ил.
9. Дейт, К.Дж. Введение в системы баз данных / К.Дж. Дейт. - К.: Диалектика; Издание 6-е, **2015**. - 784 с.
10. Аллен, Г. Тейлор SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, Вильямс, **2015**. - 416 с.
11. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. -- СПб: Питер, 2001. -- 368 с.: ил.

12. Г.С. Иванова Объектно-ориентированное программирование / Г.С. Иванова. - М.: Московский Государственный Технический Университет (МГТУ) имени Н.Э. Баумана, 2014. – 149.
13. Лутц М. **Изучаем Python**, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
14. ORM. Object-Relational Mapping Wiki – веб-ресурс. URL: <https://ru.wikipedia.org/wiki/ORM>
15. Дзямулич Н. Використання хмарних сервісів – новий етап у розвитку освітніх інформаційно-комунікаційних технологій / Н. Дзямулич – 2014. – № 10 (Ч. 1). – С. 120–124.

ДОДАТКИ

Додаток А

Settings.py

```
import datetime
import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'h1rh1lqm+rdhaw&j^gkawv1--c@((omvpjn!jsfsgghwa^@u(bc'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['127.0.0.1']

# Application definition

INSTALLED_APPS = [
    'grappelli',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'hardware',
    'catalog',
    'autoslug',
    'django_template_maths',
    'crispy_forms',
    'axes',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'axes.middleware.AxesMiddleware',
]

ROOT_URLCONF = 'DjangoShop.urls'
```

```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_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 = 'DjangoShop.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators
AUTHENTICATION_BACKENDS = [
    # AxesBackend should be the first backend in the AUTHENTICATION_BACKENDS list.
    'axes.backends.AxesBackend',

    # Django ModelBackend is the default authentication backend.
    'django.contrib.auth.backends.ModelBackend',
]

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
        'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```

# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images) and media files
# https://docs.djangoproject.com/en/3.1/howto/static-files/
CRISPY_TEMPLATE_PACK = 'bootstrap4'
STATIC_URL = '/static/'

STATIC_ROOT = os.path.join(BASE_DIR, 'static')

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
LOGOUT_REDIRECT_URL = '/'

CURRENT_YEAR = datetime.datetime.now().year

```

Додаток Б

Models.py

```

from django.contrib.auth.models import User
from django.db import models
from django.db.models import Q

from hardware.models import CPU, GPU, RAM, Notebook, PC, AbstractProduct

class Comment(models.Model):
    product = models.ForeignKey(CPU, on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField(max_length=500)
    pubdate = models.DateTimeField(auto_created=True, auto_now_add=True, null=True)

    class Meta:
        abstract = True

class CPUComment(Comment):
    product = models.ForeignKey(CPU, on_delete=models.CASCADE)

    class Meta:
        verbose_name = 'CPU коментар'

```

					КБ.КВ.Б – 125 – 04– ПЗ	
						67

```

        verbose_name_plural = 'CPU коментарі'

class GPUComment(Comment):
    product = models.ForeignKey(GPU, on_delete=models.CASCADE)

    class Meta:
        verbose_name = 'GPU коментар'
        verbose_name_plural = 'GPU коментарі'

class RAMComment(Comment):
    product = models.ForeignKey(RAM, on_delete=models.CASCADE)

    class Meta:
        verbose_name = 'RAM коментар'
        verbose_name_plural = 'RAM коментарі'

class PCComment(Comment):
    product = models.ForeignKey(PC, on_delete=models.CASCADE)

    class Meta:
        verbose_name = 'PC коментар'
        verbose_name_plural = 'PC коментарі'

class NotebookComment(Comment):
    product = models.ForeignKey(Notebook, on_delete=models.CASCADE)

    class Meta:
        verbose_name = 'Notebook коментар'
        verbose_name_plural = 'Notebook коментарі'

class Code(models.Model):
    user = models.CharField(max_length=200)
    code = models.IntegerField(default=0)

class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    pc_cart = models.ManyToManyField(PC)
    ram_cart = models.ManyToManyField(RAM)
    cpu_cart = models.ManyToManyField(CPU)
    gpu_cart = models.ManyToManyField(GPU)
    notebook_cart = models.ManyToManyField(Notebook)

    def get_all_products(self):
        products = [self.pc_cart.get_queryset(), self.ram_cart.get_queryset(),
self.cpu_cart.get_queryset(),
                    self.gpu_cart.get_queryset(), self.notebook_cart.get_queryset()]
        return products

```

```

from random import randint

from django.contrib import auth
from django.contrib.auth.models import User
from django.http import JsonResponse

from django.shortcuts import get_object_or_404, redirect
from django.contrib.auth.views import LoginView, LogoutView
from django.contrib.auth.forms import UserCreationForm
from django.urls import reverse_lazy, reverse
from django.views import generic
from django.views.generic import TemplateView, FormView

from .forms import CPUCommentForm, RAMCommentForm, GPUCommentForm, PCCCommentForm,
NotebookCommentForm, EmailConfirm
from catalog.models import CPUComment, GPUComment, RAMComment, PCCComment,
NotebookComment, Code, Cart
from hardware.models import CPU, GPU, RAM, PC, Notebook
from .email_sending import send_mail

class SignupView(generic.CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('catalog:login')
    template_name = 'login/signup.html'

class SigninView(LoginView):
    template_name = 'login/login.html'
    username: str

    def form_valid(self, form):
        if not Code.objects.filter(user=form.cleaned_data['username']):
            code = randint(1000000, 9999999)
            Code.objects.create(user=form.cleaned_data['username'], code=code)
            send_mail(form.cleaned_data['username'], code=code)
        return super(SigninView, self).form_valid(form)

    def get_success_url(self):
        return reverse('catalog:signin_code')

class SigninCodeView(FormView):
    template_name = 'catalog/code.html'
    form_class = EmailConfirm
    success_url = reverse_lazy('catalog:index')

    def form_valid(self, form):
        if form.cleaned_data['code'] == Code.objects.get(user=self.request.user).code:
            Code.objects.get(user=self.request.user).delete()
            return super(SigninCodeView, self).form_valid(form)
        else:
            auth.logout(self.request)

```

```

        return reverse_lazy('catalog:login')

class Logout(LogoutView):
    template_name = 'login/logout.html'

class CPUListView(generic.ListView):
    template_name = 'catalog/cpu_list.html'
    queryset = CPU.objects.all()
    context_object_name = 'cpus'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(CPUListView, self).get_context_data()
        return context

class SearchView(generic.ListView):
    template_name = 'catalog/search.html'
    context_object_name = 'products'

    def get_queryset(self):
        title = self.request.GET.get('title')
        products = []
        if title:
            if CPU.objects.filter(title__contains=title):
                products.append(CPU.objects.filter(title__contains=title))
            if GPU.objects.filter(title__contains=title):
                products.append(GPU.objects.filter(title__contains=title))
            if RAM.objects.filter(title__contains=title):
                products.append(RAM.objects.filter(title__contains=title))
            if PC.objects.filter(title__contains=title):
                products.append(PC.objects.filter(title__contains=title))
            if Notebook.objects.filter(title__contains=title):
                products.append(Notebook.objects.filter(title__contains=title))
        return products

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(SearchView, self).get_context_data()
        context['title'] = self.request.GET.get('title', None)
        return context

class GPUListView(generic.ListView):
    template_name = 'catalog/gpu_list.html'
    queryset = GPU.objects.all()
    context_object_name = 'gpus'

class RAMListView(generic.ListView):
    template_name = 'catalog/ram_list.html'
    queryset = RAM.objects.all()
    context_object_name = 'rams'

class PCListView(generic.ListView):
    template_name = 'catalog/pc_list.html'

```

```

        queryset = PC.objects.all()
        context_object_name = 'pcs'

class NotebookListView(generic.ListView):
    template_name = 'catalog/notebook_list.html'
    queryset = Notebook.objects.all()
    context_object_name = 'notebooks'

class CPUDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = CPUCommentForm
    template_name = 'catalog/cpu_detail.html'
    context_object_name = 'cpu'

    def get_object(self, queryset=None):
        return get_object_or_404(CPU, slug=self.kwargs.get('slug', None))

    def get_context_data(self, **kwargs):
        context = super(CPUDetailView, self).get_context_data()
        context['comments'] = CPUComment.objects.filter(product=self.object)
        return context

    def get_success_url(self):
        return reverse_lazy('catalog:cpu', kwargs={'slug': self.kwargs.get('slug',
None)})

    def post(self, request, *args, **kwargs):
        form = self.get_form()
        if form.is_valid():
            return self.form_valid(form)
        else:
            return self.form_invalid(form)

    def form_valid(self, form):
        CPUComment.objects.create(author=self.request.user,
text=form.cleaned_data['text'], product=self.get_object())
        return super(CPUDetailView, self).form_valid(form)

class GPUDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = GPUCommentForm
    template_name = 'catalog/gpu_detail.html'
    context_object_name = 'gpu'

    def get_object(self, queryset=None):
        return get_object_or_404(GPU, kwargs={'slug': self.kwargs.get('slug', None)})

    def get_context_data(self, **kwargs):
        context = super(GPUDetailView, self).get_context_data()
        context['comments'] = GPUComment.objects.filter(product=self.object)
        return context

    def post(self, request, *args, **kwargs):
        form = self.get_form()
        if form.is_valid():

```

```

        return self.form_valid(form)
    else:
        return self.form_invalid(form)

    def get_success_url(self):
        return reverse_lazy('catalog:gpu', kwargs={'slug': self.kwargs.get('slug',
None)})

    def form_valid(self, form):
        GPUComment.objects.create(author=self.request.user,
text=form.cleaned_data['text'], product=self.object)
        return super(GPUDetailView, self).form_valid(form)

class RAMDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = RAMCommentForm
    template_name = 'catalog/ram_detail.html'
    context_object_name = 'ram'

    def get_object(self, queryset=None):
        return get_object_or_404(RAM, kwargs={'slug': self.kwargs.get('slug', None)})

    def get_context_data(self, **kwargs):
        context = super(RAMDetailView, self).get_context_data()
        context['comments'] = RAMComment.objects.filter(product=self.object)
        return context

    def post(self, request, *args, **kwargs):
        form = self.get_form()
        if form.is_valid():
            return self.form_valid(form)
        else:
            return self.form_invalid(form)

    def get_success_url(self):
        return reverse_lazy('catalog:ram', kwargs={'slug': self.kwargs.get('slug',
None)})

    def form_valid(self, form):
        RAMComment.objects.create(author=self.request.user,
text=form.cleaned_data['text'], product=self.object)
        return super(RAMDetailView, self).form_valid(form)

class PCDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = PCCommentForm
    template_name = 'catalog/pc_detail.html'
    context_object_name = 'pc'

    def get_object(self, queryset=None):
        return get_object_or_404(PC, slug=self.kwargs.get('slug', None))

    def get_context_data(self, **kwargs):
        context = super(PCDetailView, self).get_context_data()
        context['comments'] = PCComment.objects.filter(product=self.object)
        return context

```

```

def post(self, request, *args, **kwargs):
    form = self.get_form()
    if form.is_valid():
        return self.form_valid(form)
    else:
        return self.form_invalid(form)

def get_success_url(self):
    return reverse_lazy('catalog:pc', kwargs={'slug': self.kwargs.get('slug',
None)})

def form_valid(self, form):
    PCComment.objects.create(author=self.request.user,
text=form.cleaned_data['text'], product=self.object)
    return super(PCDetailView, self).form_valid(form)

class NotebookDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = NotebookCommentForm
    template_name = 'catalog/notebook_detail.html'
    context_object_name = 'notebook'

def get_object(self, queryset=None):
    return get_object_or_404(Notebook, slug=self.kwargs.get('slug', None))

def get_context_data(self, **kwargs):
    context = super(NotebookDetailView, self).get_context_data()
    context['comments'] = NotebookComment.objects.filter(product=self.object)
    return context

def post(self, request, *args, **kwargs):
    form = self.get_form()
    if form.is_valid():
        return self.form_valid(form)
    else:
        return self.form_invalid(form)

def get_success_url(self):
    return reverse_lazy('catalog:notebook', kwargs={'slug': self.kwargs.get('slug',
None)})

def form_valid(self, form):
    NotebookComment.objects.create(author=self.request.user,
text=form.cleaned_data['text'], product=self.object)
    return super(NotebookDetailView, self).form_valid(form)

class IndexView(generic.ListView):
    template_name = 'catalog/index.html'
    queryset = Notebook.objects.all()
    context_object_name = 'notebooks'

class CartView(generic.DetailView):

```

```

template_name = 'catalog/cart.html'
model = Cart

def get_object(self, queryset=None):
    if self.model.objects.filter(user=self.request.user):
        return self.model.objects.get(user=self.request.user)
    self.model.objects.create(user=self.request.user)
    return self.model.objects.get(user=self.request.user)

def cart_add(request):
    product = request.GET.get('product_title', None)
    if Cart.objects.filter(user=request.user):
        cart = Cart.objects.get(user=request.user)
    else:
        cart = Cart.objects.create(user=request.user)
    if product:
        if PC.objects.filter(title=product):
            product = PC.objects.get(title=product)
            cart.pc_cart.add(product)
        elif Notebook.objects.filter(title=product):
            product = Notebook.objects.get(title=product)
            cart.notebook_cart.add(product)
        elif RAM.objects.filter(title=product):
            product = RAM.objects.get(title=product)
            cart.ram_cart.add(product)
        elif CPU.objects.filter(title=product):
            product = CPU.objects.get(title=product)
            cart.cpu_cart.add(product)
        elif GPU.objects.filter(title=product):
            product = GPU.objects.get(title=product)
            cart.gpu_cart.add(product)
    return JsonResponse({'message': 'Товар додано до кошику'})

```