

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 готелю»

ЗАВДАННЯ

на випускну кваліфікаційну роботу

Студент: Попадюк Юрій Святославович

Тема роботи: «Проект підсистеми захисту Web-сайту готелю»

затверджена Наказом університету від 1 березня 2022 р. № 123/с

Термін здачі студентом закінченої роботи 10 червня 2022 р.

Виконав студент 4-го курсу, групи ЗКБ-18
спеціальності 125 «Кібербезпека»

Вихідні дані роботи: Вчений дослідження Продумані дослідження Висновок
_____ Попадюк Ю.С.

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

Консультанти з випускної кваліфікаційної роботи: _____ Котенко В.М.

їх стосуються

Рецензент PhD, в.о.зав.каф.КН

Розділ	Консультант	Підпис	Дата
1	Котенко В.М.	_____	01.03.2022
2	Котенко В.М.	_____	01.03.2022
3	Котенко В.М.	_____	01.03.2022

Житомир – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

(назва прописними)

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ

(назва прописними)

СПЕЦІАЛЬНІСТЬ КІБЕРБЕЗПЕКА

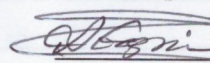
(назва прописними)

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної

інженерії та кібербезпеки

 Андрій Єфіменко

1 березня 2022 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу

Студент: Попадюк Юрій Святославович

Тема роботи: «Проект підсистеми захисту Web-сайту готелю»

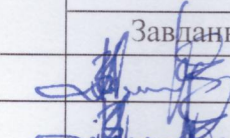
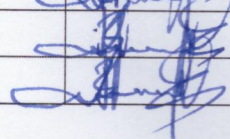
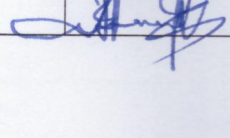
затверджена Наказом університету від 1 березня 2022 р. № 128/с

Термін здачі студентом закінченої роботи 10 червня 2022 р.

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

Об'єкт дослідження - процес захисту Web-сайту готелю
Предмет дослідження - підсистеми захисту Web-сайту готелю

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Котенко В.М.	 1.03.22	 01.03.2022
2	Котенко В.М.	 1.03.22	 01.03.2022
3	Котенко В.М.	 1.03.22	 01.03.2022

Керівник

(підпис)

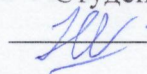
СТУДЕНТ  Попадюк Ю.С.

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	Проектування структури системи		
3	Написання програмного коду		
4	Тестування		
5	Захист		

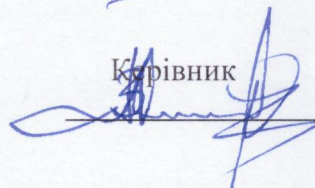
КЛЮЧОВІ СЛОВА: Django, Python, Web-

Студент



Романчук Ю.С.
(підпис)

Керівник



(підпис)

АНОТАЦІЯ

Випускна кваліфікаційна робота бакалавра складається з проекту підсистеми захисту Web-сайту готелю та пояснювальної записки.

Мета кваліфікаційної роботи – підняти рівень захищеності Web-сайту готелю шляхом розробки підсистеми захисту.

Об'єкт дослідження – процес захищеності Web-сайту готелю.

Предмет дослідження – підсистеми захисту Web-сайту готелю.

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.

					Державний університет «Житомирська Політехніка»			
					125.7.000			
Зм.	Арк.	№ докум.	Підпис	Дата	Проект підсистеми захисту Web-сайту готелю	Літ.	Арк.	Аркушів
Розроб.		Попадюк Ю.С.		25.06				
Керівник		Котенко В.М.		25.06			4	68
Реценз.		Граф М.С.		25.06		Житомирська політехніка, група ЗКБ-18		
Зав. каф.		Єфіменко А.А.		25.06				

ЗМІСТ

РЕФЕРАТ	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ	
АКТУАЛЬНОСТІ ТЕМИ РОБОТИ	7
1.1. ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ	7
1.2. АНАЛІЗ МЕТОДІВ ЗАХИСТУ WEB-САЙТУ ГОТЕЛЮ	10
1.3. АНАЛІЗ ІСНУЮЧИХ ПІДСИСТЕМ ЗАХИСТУ WEB-САЙТУ ГОТЕЛЮ	17
1.4. ПОСТАНОВКА ЗАВДАННЯ	24
Висновки по 1 розділу	25
РОЗДІЛ 2. РОЗРОБКА WEB-САЙТУ ГОТЕЛЮ ТА ПІДСИСТЕМИ ЙОГО	
ЗАХИСТУ	26
2.1. РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ПІДСИСТЕМИ ЗАХИСТУ WEB-САЙТУ ГОТЕЛЮ	26
2.2. РОЗРОБКА WEB-САЙТУ ГОТЕЛЮ	35
2.3. РОЗРОБКА ДВОФАКТОРНОЇ СИСТЕМИ ЗАХИСТУ	45
2.4. АЛГОРИТМ PBKDF2 З ХЕШЕМ SHA256	46
Висновки по 2 розділу	47
РОЗДІЛ 3. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ	
ПІДСИСТЕМИ ЗАХИСТУ WEB-САЙТУ ГОТЕЛЮ	48
3.1. РОЗГОРТАННЯ МОДЕЛІ ЕКСПЕРИМЕНТУ	48

3.2. Дослідження та порівняння налаштувань безпеки сайту при різних вразливостях	51
3.3. Рекомендації по застосуванню розробленої підсистеми захисту веб-сайту готелю	54
Висновки по 3 розділу	55
Висновки	56
Список використаної літератури	57
Додатки	59
Додаток А	59
Додаток Б	61
Додаток В	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

БД - Бази даних.

ІТ - Інформаційні технології.

ООП - Об'єктно-орієнтоване програмування

СУБД - Система управління базами даних

ЕОМ - Електронна обчислювальна машина

API - Application Programming Interface - інтерфейс програмного додатку

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

					ЗКБ-18.125.000	5

ВСТУП

Актуальність роботи. В якості одного з найголовніших вимог до інформаційних систем на сьогоднішній день виступає забезпечення інформаційної безпеки[1]. Основною передумовою для цього є наявність нерозривного зв'язку між інформаційними технологіями та основними бізнес-процесами в усіх організаціях різних сфер економіки.

Надійний захист внутрішньої інформаційної безпеки – загальносвітова проблема. На початкових етапах запровадження корпоративних локальних мереж основну проблему компанії становив несанкціонований доступ до конфіденційних даних шляхом зовнішнього злому (хакерська атака), на даний час з цією проблемою успішно справляються. У всіх ІТ-відділах розвинутих компаній, як правило, є велике число антивірусів, які успішно захищають програми (файрволи) та використовують засоби боротьби з зовнішніми атаками.

Враховуючи, що одним з головних факторів успішної діяльності комерційної структури є забезпечення інформаційної безпеки, нами було вирішено розробити підсистему захисту Web-сайту готелю.

Мета кваліфікаційної роботи – підвищення рівня захищеності Web-сайту готелю шляхом розробки підсистеми захисту.

Об'єкт дослідження – процес захищеності Web-сайту готелю.

Предмет дослідження – підсистеми захисту Web-сайту готелю.

Було поставлено такі **завдання**:

- аналіз існуючих інформаційних систем;
- аналіз і класифікація потенційних джерел загроз інформації;
- аналіз ефективності підсистем захисту;
- розробка підсистеми захисту Web-сайту готелю;
- тестування на працездатність розробленої підсистеми захисту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1. Обґрунтування актуальності теми роботи

Переважна кількість сучасних інформаційних систем створюються у вигляді Web-сайтів, тому особлива увага повинна бути приділена безпеці інформації. Перша ж поява пристрою з операційною системою і програмним додатком, провокує інтерес злоумисників.

Розробники Web-сайтів 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 врахувати те, що сайти можуть бути різноманітні, а саме: соціальні мережі, електронні бібліотеки, Web-сайти різних навчальних закладів, онлайн-магазини, офіційні сайти різних організацій, банків тощо.

Передусім, важливо звернути увагу на безпеку веб-сервера, оскільки саме він відповідає за прийом і обробку HTTP-запитів[2] від клієнтів до веб-сайту. Саме веб-сервер забезпечує функціонування численних веб-сайтів по всьому світі, й так само відповідає за базові послуги і збереження персональних даних користувачів і відвідувачів сайту. Томк забезпечення надійного захисту серверів - це одне з найважливіших завдань будь-якої організації.

Вже під час появи перших Web-сайтів зловмисники здійснювали атаки на веб-сайти важливих організацій, наприклад, такий як, американський Сітібанк (1994р.), внаслідок чого було вкрадено 12 млн. доларів, жертвою так само стали і сайти НАТО, ЦРУ, і Мін'юсту США, крім того, на початку 2022р. було здійснено ряд хакерських атак на системи зокрема державних органів України.

Слід зауважити, що географічне розташування сервера, жодним чином не впливає на його захист. Провести атаку на нього можливо з будь-якої точки доступу. Це пов'язано з тим, що Web-сервери завдяки своїй відкритості розраховані на передачу інформації між необмеженою кількістю користувачів і саме тому мають безліч вразливостей. Наприклад, зловмисник може спробувати

[illegible]

внести якусь зміну (модифікацію) в код HTTP сервера, самих сторінок веб-сайту або сервера бази даних, помінявши тим самим його початкову функціональність.

До найбільш поширених дії правопорушників внаслідок кібератак над зламаними сайтами, можна віднести:

публікацію недостовірної інформації на сайті;

отримання несанкціонованого доступу до сервера, де були зібрані дуже важливі відомості;

атака баз даних про власників емітованих банками пластикових карток;

викрадення конфіденційних даних, що стосуються власників банківських карток, їх прізвищ, імен, та іншої контактної інформації, а потім заволодіння їх коштами;

створення вірусів, які легко підбирають та зламують паролі, а потім за допомогою вже заражених машин здійснюють розсилки спаму чи використовують як бази для зберігання викраденої інформації;

здійснення атак на веб-сайти урядових установ різних країн з метою їх блокування чи поширення на них недостовірної інформації;

поширення вірусів, які призводять до значного уповільнення швидкості роботи інтернету (непоодинокими були випадки, коли в результаті таких дій деякі регіони виявилися взагалі відрізнаними від мережі);

Dos-атаки (Denial of Service- "відмова у обслуговуванні") на мережу для переповнення її додатковими запитами настільки, що корисний трафік або знижувався до мінімуму, або звагалі переривався.

На даний час існують різні програми, а також спеціальне Web-тестування, які мають змогу визначати приблизний коефіцієнт безпеки та надійності сайту. Завдяки такому тестуванню можливо наперед оцінити безпеку сайту і його вразливість.

Прийнято вважати, що практично кожен третій сайт відстежується сторонніми людьми. Це стає неприємним і насторожуючим фактом для Web-розробників. Адже, володіючи необхідною інформацією про уразливість сайту,

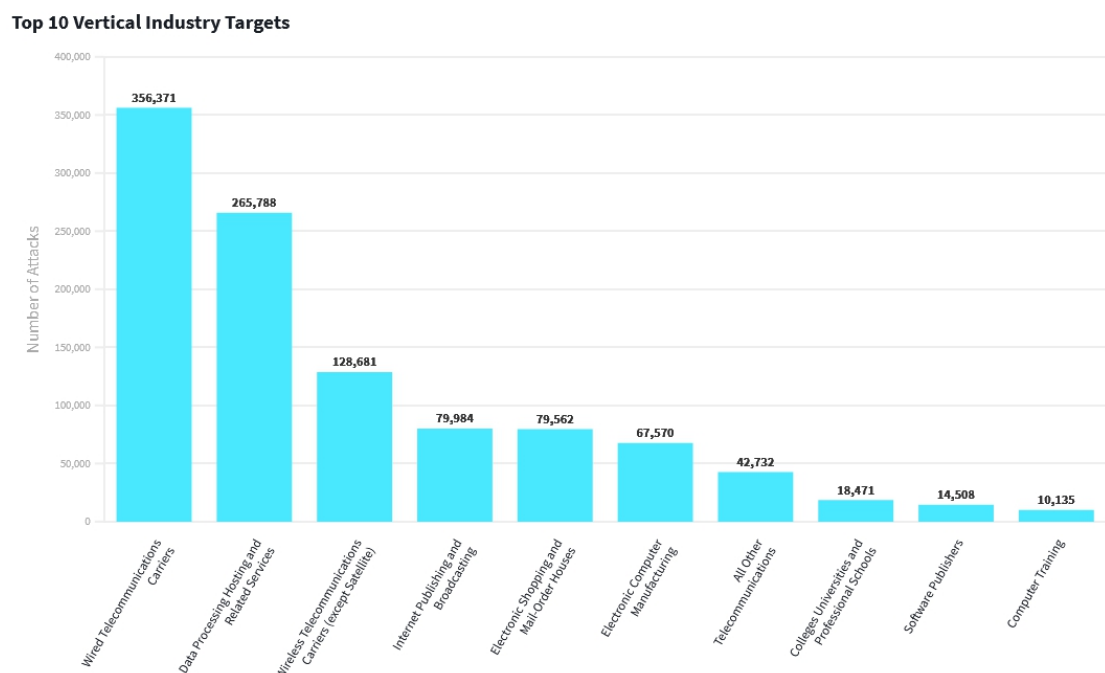
					ЗКБ-18.125.000	8

його можна з легкістю зламати, тому головним завданням розробників в питанні про надійність і захист сайту стає запобігання якнайбільшій кількості цих вразливостей чи необхідність їх заховати як слід.

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

Великі підприємства підходять до питання інформаційної безпеки[3] з набагато більшим ступенем відповідальності, ніж компанії малого та середнього бізнесу (СМБ), які страждають найбільше. За статистикою, саме у останніх виникає більше інцидентів в плані інформаційної безпеки.

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



Мал. 1.1 – Статистика жертв DDoS-атак

Сучасні засоби захисту, безперечно, є невід'ємною частиною заходів щодо забезпечення інформаційної безпеки. Тому захист свого сайту є одним з найважливіших завдань Web-розробників. Якщо ж не надати цьому належної уваги – можна отримати або ж створити значні збитки, як у матеріальній, так і в політичній сферах. Враховуючи, що інтерес зловмисників до атак на Web-сайти не вщухає навіть зараз (не дивлячись на те, що всі зловмисні діяння караються

законом), розробникам Web-сайтів варто, передусім, задуматися про його безпеку. Є безліч засобів захисту Web-сайтів, проте не дивлячись на їх ефективність, все одно потрібно час від часу тестувати Web-сайти на вразливості, якими в будь-який момент може скористатися зловмисник.

Таким чином, питання створення систем захисту Web-сайтів і надалі залишається актуальним.

1.2. Аналіз методів захисту Web-сайту готелю

Засобами захисту веб-сайтів є програмні або апаратно-програмні комплекси, створені забезпечення захисту веб-додатків і різних веб-сервісів.

До основних засобів захисту веб-сайтів слід віднести наступні:

Web Application Firewall (WAF);

засоби аналізу веб-сайтів щодо наявності вірусів;

балансувальник навантаження на Web-додатки;

засоби захисту від DDoS;

сканери захищеності Web -додатків (WASS).

Перелічені засоби захисту веб-сайтів забезпечують захист від таких типів загроз:

загроза зміни контенту Web-сайту. Дана дія дозволяє зловмисникові розмістити на Web-сайті будь-яку інформацію, у тому числі тексти сторінок, а також дані платіжних реквізитів;

загроза видалення даних. Дана дія дозволяє зловмисникові видалити усі дані, включаючи інформацію про паролі, бази даних клієнтів, постачальників, матеріальних ресурсів тощо;

загроза застосування шкідливих програм. В результаті такої дії власники і користувачі сайту можуть не одразу побачити будь-які зміни. Однак шкідлива програма зможе вчиняти шкідливі дії, включаючи переадресацію на шкідливий чи шахрайський сайт, крадіжку персональних даних користувачів, зараження відвідувачів Web -сайту вірусами і т.п.;

					ЗКБ-18.125.000	
						10

загроза розсилки спаму. В результаті даної дії Web-додаток буде використовуватиметься зловмисниками для розсилки небажаних повідомлень. Це може призвести до занесення веб-сайту в спам-листи, що не дозволить надалі відправляти санкціоновані повідомлення;

загроза збільшення навантаження на ресурси, або ж DDoS-атака. Під цим слід розуміти відправку на адресу Web-сервера спеціальних запитів, які призводять до ускладнення доступу користувачів до сайту чи падіння прикладного і системного програмного забезпечення Web-сервера.

Розглянемо більш детально кожен метод захисту.

WAF (Web Application Firewall) - міжмережевий екран для Web-додатків. Це інструмент для фільтрації трафіку, який працює на прикладному рівні та захищає Web-додатки методом аналізу трафіку HTTP / HTTPS і семантики XML / SOAP. WAF може встановлюватися на фізичний чи віртуальний сервер і виявляє різноманітні види атак. Діє міжмережевий екран як проксі-сервер, однак з огляду на можливості вивчати HTTPS-трафік методом перевірки сертифіката конкретного сервера, WAF розрахований також 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 відбувається в режимі реального часу за допомогою порту SPAN.

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

Negative. Певний «чорний список», який забороняє прийом конкретної інформації, прописаної у налаштуваннях. Захищає веб-додатки на прикладному рівні (аналог IPS), у той же час вмє оцінювати потенційні загрози ьльш детально і частіше застосовується для забезпечення захисту від «популярних», а також специфічних типів атак. Аналізує вразливі місця конкретних веб-додатків.

					ЗКБ-18.125.000	11

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

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

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

Можливості WAF:

Оперативно реагувати на будь-які види атак на Web-додатки, що входять в OWASP Top 10 (Open Web Application Security Project - відкритий проект забезпечення безпеки веб-ресурсів)[5]. Захист, при цьому, забезпечується заданими активними правилами.

Перевіряти, чи надходять на додаток трафік HTTP / HTTPS також інші запити, після чого приймати рішення на підставі заданих правил та політики (дозволити, блокувати, відправити повідомлення).

Підтримувати стабільну роботу моделей безпеки Negative та Positive, а також дотримуватися усіх заданих в рамках моделей безпеки правил.

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

Запобігати витоку інформації, перевіряючи, який виходить від Web-додатків трафік HTTP / HTTPS, і вживати відповідні заходи на підставі заданих активних правил.

Постійно вести журнал подій та записувати нього усі виконані операції, аналітичну інформацію та інші події, які відбулися.

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

Перевіряти усі вхідні дані, які застосовуються для надсилання чи отримання інформації від Web-додатків.

					ЗКБ-18.125.000	
						12

Захищати від атак, спрямованих саме на Web Application Firewall.

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

Головна відмінність брандмауера від інших методів захисту веб-додатків полягає у глибокому аналізі трафіку протоколів прикладного рівня.

Можна виділити такі засоби аналізу Web-сайтів на наявність вірусів.

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

після завершення завантаження сторінки у неї додається javascript, який виконує drive-by download атак;

користувач йде зі сторінки, а в цей момент підвантажується код і відкриває popunder з контентом "для дорослих";

відвідувач сайту знаходиться на сторінці ще кілька секунд і тільки після цього його перенаправляють на якусь платну підписку за смс.

Коли заздалегідь невідомо, який саме код провокує такі несанкціоновані дії, то виявити його статичним аналізом надзвичайно складно. Проте, є також аналіз динамічний, або іноді його ще називають "поведінковим". Якщо веб-сканер сучасний, він буде не тільки аналізувати вихідний код сторінки або файлів, але ще й намагатиметься виконувати якісь операції, емулюючи дії реального відвідувача. Після кожної дії чи за певних умов роботи сканер аналізує зміни і накопичує дані для підсумкового звіту: завантажує сторінку в декількох браузерах (причому, не просто з різних User-Agent'ів, а й з різними значеннями об'єкта navigator в javascript, різними document.referrer і т.п.),

					ЗКБ-18.125.000	13

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 "шкідники" динамічно додають або приховують об'єкти на javascript, а також виконують фонові завантаження всередині динамічних фреймів. Наприклад, код зараженого віджета може через 3 секунди чи по руху миші завантажити скрипт, який в подальшому скерує на сторінку javascript з перенаправленням на завантаження небезпечного .apk файлу. Зрозуміло, 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, міжмережеві 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:

Мережевий. Такий метод має на увазі використання однієї 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.

					ЗКБ-18.125.000	14

Транспортний. Такий спосіб передбачає централізований розподіл запитів до сервера між різними пристроями. При такому балансуванні навантаження можуть використовуватися за допомогою різних алгоритмів (наприклад, запит буде направлятися найменш завантаженому серверу чи тому серверу, який знаходиться наступним в черзі, т.з. «круговий перебір»).

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

Балансування навантаження може використовуватися для реалізації таких цілей:

- виключення можливості виділення ресурсів сервера на єдиний запит, тоді як інші запити будуть відхилятися;

- ефективного використання ресурсів, коли не допускається простоювання сервера, у той час, як інші завантажені;

- збільшення продуктивності чи зменшення часу, необхідного для виконання запиту;

- зменшення часу, який необхідний для надання відповіді на запит клієнта.

Розробляючи варіанти балансування навантаження, адміністратор має повністю розуміти, як саме буде відпрацьовуватися балансування навантаження, яким чином можна буде в подальшому масштабувати архітектуру. Балансування навантаження забезпечує:

- управління трафіком;

- збільшення швидкості роботи додатків, захист додатків й збереження інформації;

- моніторинг трафіку.

Балансування навантаження широко використовується у корпоративних системах захисту, забезпечуючи рівномірність навантаження на сервери, для захисту Web-сервісів і сайтів, систем фільтрації Web -контенту і т.п. Також існують різні онлайн-сервіси з можливістю організації балансування за певним методом.

Засоби захисту від DDoS-атак[7] - це спеціалізовані програмно-апаратні та програмні засоби, призначені для захисту Web-серверів (Web-сайтів) організації від розподілених атак типу «Відмова в обслуговуванні». Такий тип атак спрямований на порушення доступності (повне або часткове) різних публічних сервісів, в тому числі Web -сайтів, баз даних і серверів додатків, при якому легітимні користувачі позбавляються можливості отримання доступу.

Принцип атаки полягає у генерації великої кількості паразитного трафіку, на обробку якого буде виділятися величезний об'єм обчислювальних ресурсів. В результаті атаки, власники компаній можуть понести великі фінансові втрати через те, що їх клієнти не зможуть скористатися сервісами компанії.

Загалом, захист від DDoS можна організувати двома способами:

Скориставшись послугами зовнішніх компаній за моделлю SaaS (Security as a Service). Таке рішення дозволить не вникати у технічну складову питання захисту від DDoS-атак, а також зменшить витрати на інформаційну безпеку;

Шляхом впровадження власних засобів захисту від атак типу «Відмова в обслуговуванні», а також шляхом застосування організаційно-технічних заходів, які спрямовані на збільшення пропускної спроможності каналів зв'язку і використання механізмів балансування навантаження.

Засоби захисту від DDoS-атак on-premise, як правило, розташовуються на кордоні мережі, в розриві, для своєчасного виявлення та фільтрації шкідливого чи паразитного трафіку. Зазвичай є відповідне окреме обладнання, яке розташовується на віртуальній машині. Такі засоби вимагають постійного оновлення, оскільки сценарії DDoS-атак постійно змінюються. Ціна на спеціалізоване обладнання, зазвичай, залежить від заявленої швидкості обробки

					ЗКБ-18.125.000	
						16

трафіку та пропускної здатності. У той же час, з огляду на високу вартість засобів захисту, оптимальному комплексу може бути не під силу обробка дуже великої DDoS-атаки.

1.3. Аналіз існуючих підсистем захисту Web-сайту готелю

Сучасний готельний бізнес вже освоїв такий канал залучення і обслуговування клієнтів, як Інтернет. Ефективність цього каналу оцінили вже навіть найбільш консервативні власники готелів. З примітивної базової програми електронне обслуговування перетворилося на бізнес з багатомільйонними оборотами.

Останнім часом прискорюється зростання електронної комерції – резервування через Інтернет дозволяє власникам здійснювати прогноз щодо напливу клієнтів у курортні періоди, а також заощадити на персоналі.

Однак, як і завжди в бізнесі, нові можливості передбачають і нові ризики. Так, Web-сайт з можливістю зарезервувати номери, оплатити замовлення, приносить прибуток і знижує витрати, однак власникам доводиться дбати про те, щоб він нормально функціонував, не допускав шахрайських операцій, не допускав витоків призначених для користувача даних чи інших інцидентів, які впливають на обсяг замовлень, прибуток, лояльність клієнтів і санкції від регуляторів ринку.

Перший елемент захисту - це постійна доступність сайту. Немає жодного сенсу проводити рекламні компанії та оптимізувати резервування номерів, якщо користувачі, перейшовши по рекламованій адресі, не отримують доступу до сайту.

Водночас і сайт, який зберігає свою доступність, також може нести в собі загрозу. На сайті може зберігатися конфіденційна інформація - дані облікових записів (особистих кабінетів) клієнтів сайту, які можуть містити окрім персональних даних їх- прізвища, імена, по батькові, номер мобільного телефону, адресу, також і дані, прив'язаних до особистого кабінету платіжних карт, які

					ЗКБ-18.125.000	
						17

зловмисники можуть використовувати у власних інтересах, для оплати товарів чи послуг.

Чим більш відомий готель, тим більше у нього зареєстрованих користувачів, а, отже, більший інтерес представляє база облікових записів клієнтів, яка зберігається на сайті. Така інформація має значну вартість тому, що з її допомогою можна отримати пряму вигоду - використовувати дані кредитної картки для оплати своїх покупок чи оплачувати товари від імені законних користувачів, але вказувати іншу адресу доставки.

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

Також важливо, щоб інформація на сайті відповідала тому, що саме хотіли розмістити його розробники. Це особливо важливо для комерційних сайтів – викривлення (спотворення) інформації готелем може привести до відмови клієнта від бажання скористатися його послугами. Трапляються також і більш складні атаки, ніж підміна інформації про готель: наприклад, зловмисники підміняють телефон «гарячої лінії» чи посилання з кнопки «резервувати» на інші вірусні сайти, перенаправляючи на нього клієнтів, які мають наміри скористатися послугами після ознайомлення.

Компанії, які довіряють програмістам реалізовувати бізнес-процеси, часто не перевіряють розроблений додаток, який запускається, на наявність так званих «декларованих можливостей» – функціоналу, який не був передбачений в технічному завданні. Це можуть бути як службові функції, наприклад, налагодження гілки, які спрощують деякі операції в процесі тестового функціонування сервісу, так і навмисні «закладки», які реалізують можливості зловмисних операцій.

Є різні системи захисту Web-додатків, як інтегрованих, так і спеціалізованих: різного роду сканери, анти-DDoS продукти й сервіси,

					ЗКБ-18.125.000	18

міжмережеві екрани прикладного рівня (WAF-web application firewall). Існуюча проблема зовнішніх (на сленгу - "навісних") систем захисту в тому, що практично всі вони не встигають перебудовуватися так швидко, як змінюється об'єкт захисту.

Основні способи зміни налаштувань систем захисту - це зміни вручну і самонавчання. При цьому, жоден з цих способів не працює при частоті змін навіть сто змін на добу. За експлуатаційної документації на WAF, наприклад, самонавчання такого продукту може займати від однієї до трьох годин, тобто якщо об'єкт захисту буде змінюватися 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, потрібно мати солідну професійну команду, яка працюватиме в цілодобовому режимі.

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

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

Перевагою такого підходу є можливість використання паралельно різних систем захисту для визначення атаки. Недоліком же є делегування прийняття рішення операторам системи контролю. Це вимагає утримувати дорогу команду інформаційної безпеки, що при інтенсивному трафіку не виключає наявності інцидентів – буває, що рішення операторами приймаються тоді, коли вже пізно - атака пройшла.

					ЗКБ-18.125.000	
						19

Активна стратегія полягає у виборі рішень, які здатні без помилок працювати в режимі блокування, і інтеграції їх таким чином, щоб налаштування змінювалися в автоматичному режимі. Будь-яка зміна на захищеному сайті повинна негайно досліджуватися на предмет наявності уразливостей, знайдені уразливості повинні відправлятися на виправлення особам, відповідальним за цей функціонал – адміністраторам, інженерам чи програмістам – а до виправлення можливість зловмисного використання функціоналу має бути заблоковане «навісними» системами інформаційної безпеки.

Найбільш часте поєднання активних систем безпеки – статичний сканер досліджує код додатка, знаходить в ньому вразливості, які аналізуються динамічним сканером для відкидання загрози помилкових спрацьовувань, далі знайдені і підтверджені можливістю експлуатації вразливості передаються WAF і анти-DDoS, які блокують небезпечні звернення до цих уразливостей. Паралельно інформація про вразливість до рекомендацій щодо виправлення передається інженерам або програмістам, в залежності від того, що саме треба міняти - вміст сайту, сам функціонал додатка, чи налаштування інфраструктури.

Перевагою такого рішення є активність, інтегрованість та незалежність від людини – рішення, налаштоване один раз, працює само собою, без помилкових спрацьовувань відбиваючи атаки. Клієнти таких рішень про атаки дізнаються зі звітів і можуть залучати своїх кваліфікованих фахівців для більш творчої роботи, ніж чергування. Такі рішення стали користуватися особливим попитом в умовах економічної кризи, коли не кожен господарюючий суб'єкт може дозволити собі кілька змін кваліфікованих фахівців інформаційної безпеки для цілодобових чергувань.

Далі розглянемо декілька прикладів систем захисту веб-сайтів.

Усі способи захисту ділять на три великі групи (мал. 1.2):

					ЗКБ-18.125.000	
						20



Мал. 1.2 – Способи захисту веб-сайтів

Захист CMS

Стержнем, основою є CMS (система управління сайтом і всією інформацією на ньому). На нашу думку для забезпечення максимального захисту ресурсу слід застосовувати вказані нижче методи.

SSL-сертифікат

Функцією протоколу SSL є надійний захист даних в інтернеті, завдяки шифруванню інформації, що передається. Наявність SSL-сертифікату - електронного цифрового підпису вашого сайту, забезпечує доступність такого роду безпеки. Вказаний сертифікат містить в собі:

- доменне ім'я;
- відомості про особу, на яку оформляється сертифікат;
- місцезнаходження власника;
- термін дії;
- базові дані про надавача сертифіката.

Електронний цифровий підпис підтверджує, що домен належить діючій компанії, власник якої використовує секретний ключ на законних підставах. Отже, сайт захищений SSL від фішинг атак.

SSL-сертифікат можна отримати двома способами: звернутися до провайдера, що надає підтримку SSL, або в центр сертифікації. При цьому, якщо сайт, розміщений на сервісах Google (Google Мій бізнес, Blogger), а також

у разі співпраці з партнерами (Bluehost, Shopify, Weebly, Wix), сертифікат можна отримати безкоштовно.

У той же час, звернення в спеціалізовані компанії для сертифікації, вимагатиме фінансових витрат – близько 100 \$.

Важливою умовою для безпечного функціонування ресурсу є регулярне оновлення CMS. Незахищені місця, через які легко зламати сайт усуваються кожною новою версією технічної частини.

Існують спеціальні плагіни, які забезпечують додатковий захист ресурсу. Кожен плагін для захисту від різних атак пропонує певне комплексне рішення, перевірку на наявність спроб злому і вірусів, створення резервних копій ресурсу тощо. Вони швидко встановлюються, є простими в налаштуваннях та довели свою ефективність у роботі.

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

Themes Security

Плагін, що пропонує 30 видів надійного захисту від:

SQL-ін'єкцій;

Шелл;

XSS;

злому через FTP або SSH;

злому phpMyAdmin;

злому через сусідні хостинги;

злому через уразливість в механіці;

брутфорс-атак і ботів, які шукають уразливості ресурсу;

захист від ДДОС-атак на сайт.

Themes Security знаходить вразливі місця і підбирає способи їх усунення, відображає будь-які атаки на файлову систему або бази даних, інформуючи про будь-які зміни. Також плагін може змінити IP користувача, адресу адміністратора, входу на сайт, шлях до папки wp-content. З допомогою Themes

					ЗКБ-18.125.000	22

Security створюються бекапи для швидкого (у разі необхідності) відновлення втрачених даних.

Wordfence Автоматично перевіряє на зараження а також спроби злому. Його основними функціями є: розпізнавання шкідливого трафіку, загроз, блокування спроб імплементації вірусного коду і користувачів, які порушують правила взаємодії з сайтом, забезпечення безпечного входу. Під час подальшого сканування надає звіт із виявленими можливими проблемами та результатами боротьби з ними.

Захист сервера

Для того щоб обмежити зловмисникам доступ до серверу, виконуються такі правила технічної безпеки:

- користувачі обмежуються в правах на доступ до бази даних;
- інформація, яку вводить користувач на сайті, відстежується;
- постійно змінюються паролі. Створити складний пароль можна за допомогою онлайн-генераторів;

- створюються бекапи сайту: самостійно за допомогою панелі управління хостингу, плагінів чи звернення до технічної підтримки хостингу.

У більшості хостингів передбачена можливість робити копії сайту. Для цього потрібно зайти в панель управління, обрати цю функцію і дотримуватись інструкцій. Прикладом слугує хостинг ukraine.com.ua.

Створювати бекапи можна і за допомогою плагінів. Вони бувають безкоштовними і платними, з можливістю автоматичного копіювання, роблять повні чи часткові копії, зберігають на комп'ютер або в хмарний сервіс. Наприклад, для WordPress є такі плагіни, як BackUpWordPress, BackWPup, BackupBuddy, Backup by Supsysic та інші. Обирати можна в залежності від ваших завдань, розміру сайту, а також функціональних можливостей.

Для користувачів панелі адміністрування, які мають повний доступ до відповідних даних сайту, та тих, хто має право користуватись сервером, необхідно провести процедуру щодо виявлення вірусів за допомогою антивірусних програм. Збереження інформації та даних веб-сайтів повинно

					ЗКБ-18.125.000	
						23

забезпечуватися постійним антивірусним захистом. При цьому важливо підтримувати актуальну (оновлену) версію програми.

До найнадійніших і популярних антивірусів на ринку програмного забезпечення можна віднести:

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. Постановка завдання

У даній кваліфікаційній роботі розглядається питання розробки підсистеми захисту Web-сайту готелю. В цілому, можна виділити такі завдання для розроблюваної підсистеми захисту Web-сайту:

система захисту інформації повинна працювати як єдине ціле. Цілісність системи полягає в наявності єдиної мети її функціонування, наявності інформаційних зв'язків між її елементами, структуризованості побудови підсистеми управління системою захисту інформації;

система захисту інформації повинна унеможливити злам та доступу до інформації, що захищається;

в системі захисту інформації методи і засоби захисту повинні бути максимально прозорими для законного користувача, не створювати йому великих додаткових незручностей, пов'язаних з доступом до інформації.

Системи захисту інформації повинна виявляти і оцінювати загрози зовнішніх дестабілізуючих факторів і забезпечувати захист від них

					ЗКБ-18.125.000	24

Висновки по 1 розділу

У даному розділі проведено аналітичну роботу над доведенням актуальності теми роботи, а саме розробки підсистеми захисту Web-сайту готелю. Виділено основні види загроз, що є актуальними на даний час, приведено приклади та статистику потерпілих від одного з найпоширенішого виду атак – DdoS.

Виділено методи захисту Web-сайтів, розглянуті переваги та з'ясовано недоліки таких методів, як:

Web Application Firewall (WAF);

Засоби аналізу Web-сайтів на наявність вірусів;

Балансувальник навантаження на Web-додатки;

Засоби захисту від DDoS;

Сканери захищеності Web-додатків (WASS).

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

Виділено перелік ключових вимог до розроблюваної підсистеми захисту Web-сайту готелю.

РОЗДІЛ 2. РОЗРОБКА WEB-САЙТУ ГОТЕЛЮ ТА ПІДСИСТЕМИ ЙОГО ЗАХИСТУ

2.1. Розробка структурної схеми підсистеми захисту Web-сайту готелю

Для планування структурної схеми підсистеми захисту, пердусім варто розглянути структуру самого Web-сайту готелю, тобто його внутрішні складові.

Зразок проектування (патерн проектування) – це архітектурне рішення[8], яке можна повторити. Різні патерни додатками використовуються у різних місцях. Програмісти за допомогою патернів досить швидко вирішують складні завдання. Для розробки інформаційних систем, або для конкретного завдання важливо підібрати патерн, який найбільше підходить.

Model-View-Controller – перший та головний. MVC - це базовий патерн, який дав розвиток новим технологіям, оскільки застосовувався в різних технологіях.

Склад MVC описують два пункти:

View - це проекція моделі;

Controller - це взаємодія користувача із системою.

Controller отримує вхідні дані, а View - вихідні, разом з тим між ними також відбувається багато операцій. Для середніх та незначних проєктів така архітектура підходить.

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

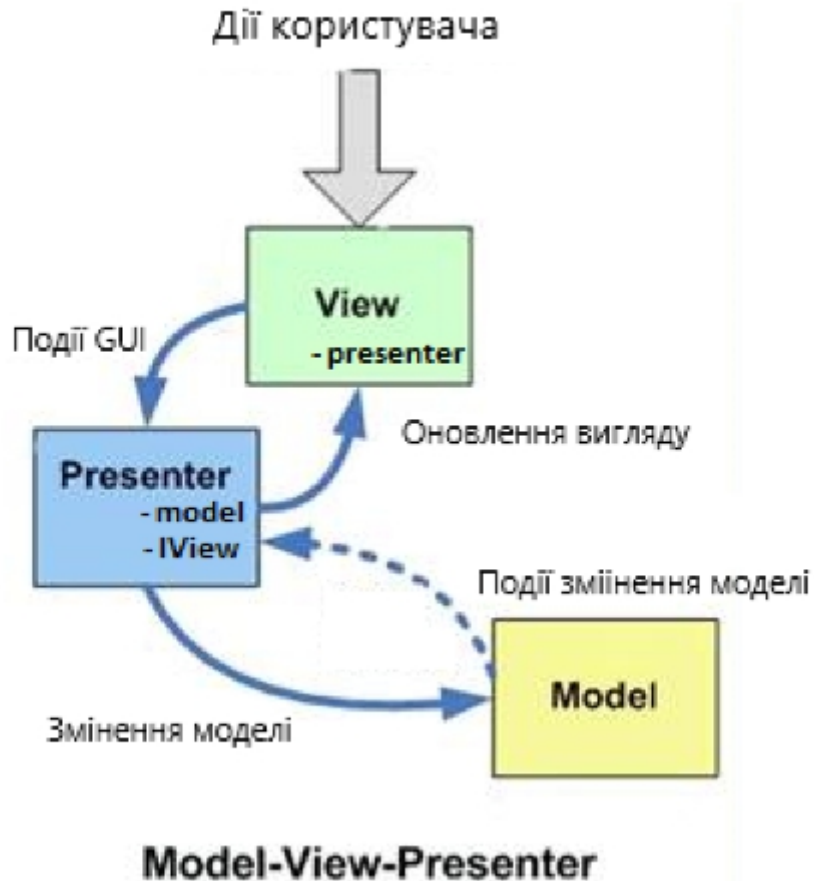
MVP – Model-View-Presenter;

MVVM – Model -View-View Model;

MVC – Model-View-Controller.

Проведемо порівняння кожного.

Model-View-Presenter(див. мал. 2.1)



Мал. 2.1 – Структура Model-View-Presenter

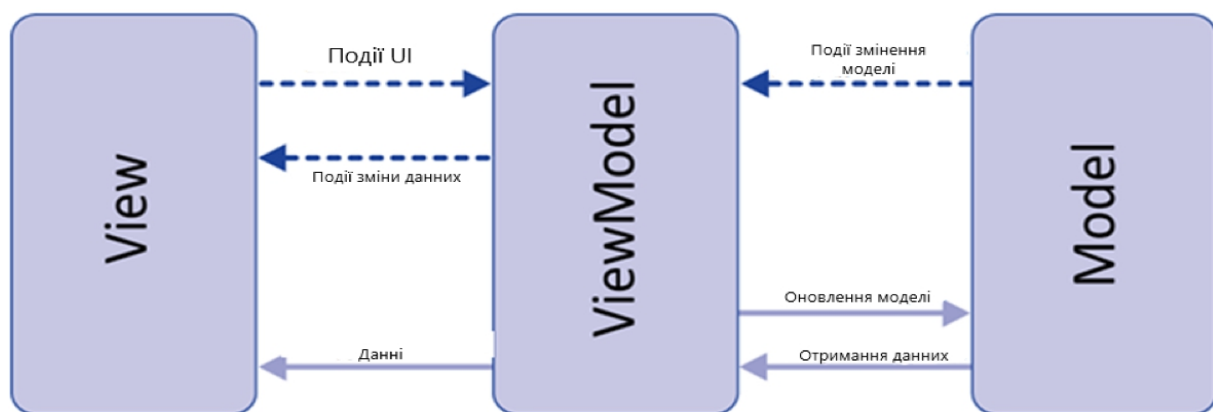
MVP легкий для розробки тестів, полегшує unit-тестування, завдяки View, яка реалізовує кілька інтерфейсів, та може використовуватися багаторазово.

Розглянемо MVP і MVC як основоположні ідеї, а не патерни розробки, для кращого і ефективнішого створення інтерфейсів.

MVP подібно до інших патернів (MVC, MVVM) прискорює розробку і розділяє відповідальність між спеціалістами, є зручним тестуванні та підтримці. Найбільшого використання набув у розробці мобільних додатків, та є досить популярним.

Відмінність MVC і MVP полягає у тому, що в MVC оновлена модель, яка вказує, що потрібно показати інші дані. Якщо необхідно здійснити вибір даних власноруч, то такий патерн називається MVP.

Model-View-View Model(див. мал. 2.2)



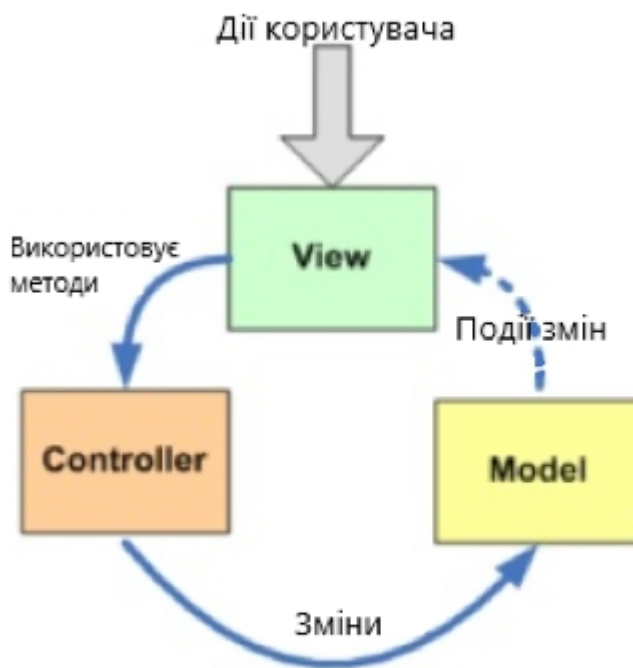
Мал. 2.2 – Структура Model-View-View Model

MVVM – це один з способів біндингу, у якому прив'язуються до параметрів view-моделі елементи view. Наприклад, поля password, email, і розмітка є зв'язаними. Редагування в нашій моделі, призводить до змін в розмітці.

Контакт між ViewModel та View безпосередньо не відбувається. Натомість вона дозволяє контактувати за допомогою властивостей і методів у вигляді команд. З метою отримання інформації з ViewModel та виклику на ній команд (методів) View може приєднатись до цих властивостей. Для зв'язки View і ViewModel, XAML Databinding користується рефлексією, що дає можливість використання будь-якої ViewModel з необхідними якостями для View.

У патерні Model-View-ViewModel відокремлюється логіка додатку від візуальної частини (подання). Даний патерн задає загальну архітектуру програми.

Model-View-Controller(див. мал. 2.3)



Model-View-Controller

Мал. 2.3 – Структура Model-View-Controller

Controller та View є залежними від моделі, тоді як модель не залежить від них.

Властивості Controller:

Controller визначає, яка View має відображатись;

взаємодія з виконуваними операціями впливає лише на Controller. В свою чергу Controller впливає на модель та створює іншу View.

Controller реагує відповідно до закладеної в нього логіки на зовнішню подію та змінює модель, викликаючи відповідний метод. При зміні моделі використовується подія зміни, внаслідок чого View відображає модель за оновленими даними.

Використання MVVM і MVP-патернів, може здаватися простим та схожим. Різниця в тому, що MVVM зв'язування View з View-моделлю здійснюється автоматично, а для MVP – потрібно програмувати. По управлінню поданням MVC, має більше можливостей.

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 збірка програм та мов програмування, які дозволяють створювати бази даних і керувати ними. Використання СУБД є актуальним, оскільки більшість сайтів не зможуть функціонувати без бази даних.

Основні можливості СУБД:

забезпечує керування даними на зовнішніх носіях;

забезпечує керування даними в оперативній пам'яті із використанням дискового кешу;

веде облік змін, здійснює резервне копіювання і відновлення бази даних; підтримує мови БД.

Однією з ознак класифікації є ґрунтування СУБД на будь-якій моделі даних.

Види СУБД:

реляційні;

ієрархічні;

мережеві;

об'єктно-орієнтовані;

об'єктно-реляційні.

Найбільш популярними залишаються реляційні бази даних при створенні веб-додатків і сервісів. Мова SQL (Structured Query Language – структурована мова запитів)[10] використовується для керування базами даних. Удосконалення SQL перетворило її на інструмент розробника з інструменту роботи користувача з базою даних.

В різних рейтингах найпопулярніших СУБД перші місця традиційно займають MySQL, Microsoft SQL Server, PostgreSQL, SQLite.

					ЗКБ-18.125.000	
						30

MySQL.

Найпоширенішою СУБД вважається MySQL - це надійна і, до того ж, безкоштовна система із простим інтерфейсом та можливістю синхронізації з іншими базами даних, реляційна СУБД швидка і гнучка, з відкритим вихідним кодом, забезпечена підтримкою великої кількості різних типів таблицьщо дозволяє використовувати MySQL великим корпораціям та невеликим компаніям.

Microsoft SQL Server

Це фірмова СУБД, розробником якої є Microsoft, оптимальна як для операційних систем сімейства Windows, так і для Linux, синхронізується з іншими продуктами компанії Microsoft, надійно захищає дані та має простий інтерфейс, однак, відрізняється зокрема підвищеним споживанням ресурсів та високою вартістю ліцензії. Проте, оскільки продукти корпорації Microsoft використовуються багатьма компаніями, дана СУБД заслужено зберігає свою популярність.

PostgreSQL

Універсальна СУБД – досить популярна і безкоштовна система. Підходить для роботи з більшістю популярних платформ, знайшла найбільше застосування для управління БД різних сервісів і веб-сайтів. Її перевагою серед безкоштовних СУБД, що в більшості є реляційними, є те, що вона об'єктно-реляційна.

SQLite

Програмна бібліотека SQLite забезпечує безсерверний, автономний, транзакційний механізм бази даних SQL з нульовою конфігурацією, є одним з найбільш швидкозростаючих механізмів баз даних з точки зору популярності, а не розміру.

Таблиця 2.1 – Плюси та мінуси СУБД

					ЗКБ-18.125.000	
						31

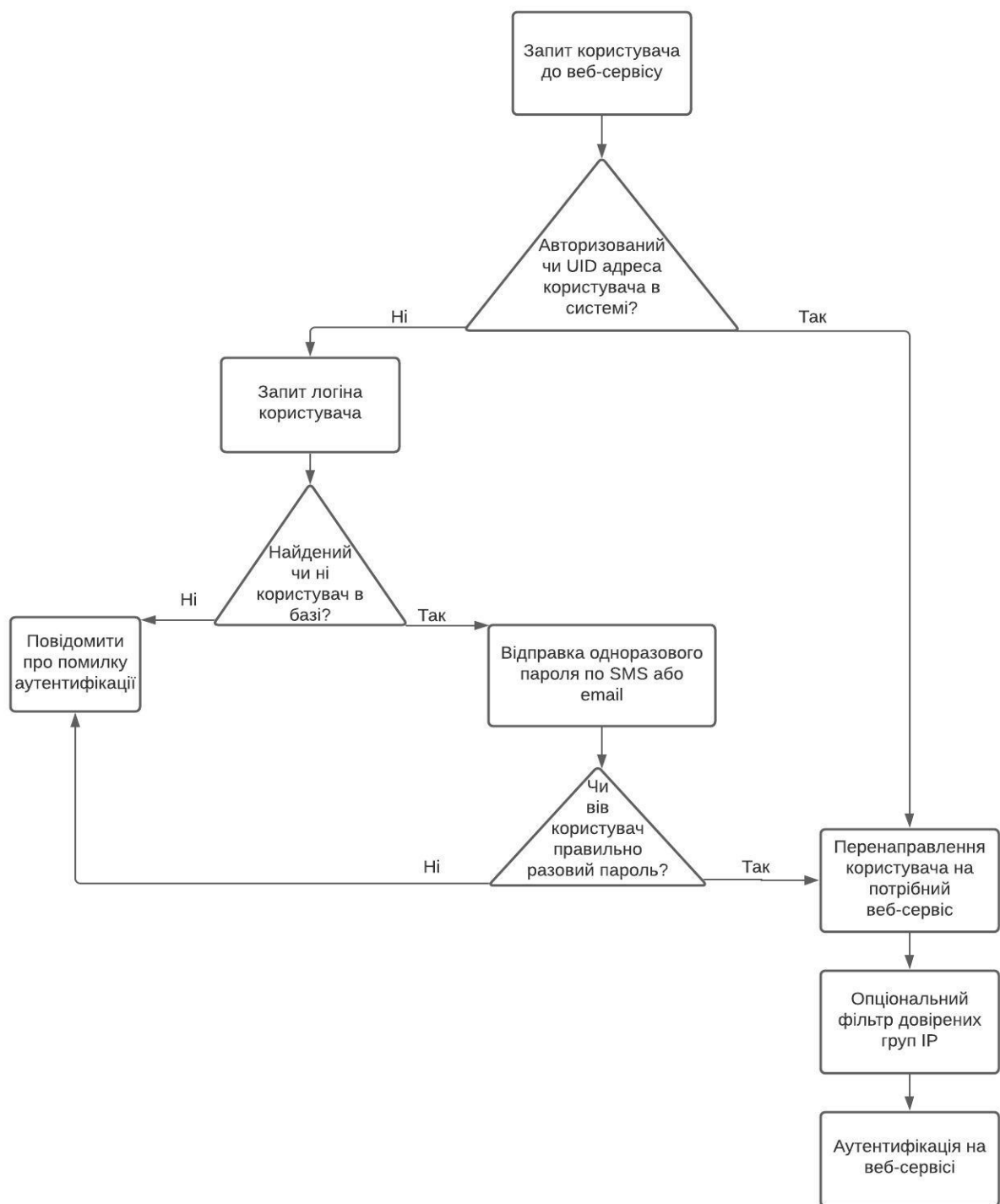
СУБД	Плюси	Мінуси
Postgre	Високофункціональна з відкритим кодом і безкоштовна СУБД, має підтримку розробників	малопопулярний продукт з низькою продуктивністю під час обробки великих обсягів інформації,
MS SQL	Високофункціональна, зручна СУБД в середовищі Windows	Небезкоштовна, складна у використанні
MySQL	Безкоштовна СУБД, з відкритим код, не містка, легка у використанні mysql з набором необхідних функцій, має підтримку розробників, використовується багатьма платформами, має високий рівень захисту	Посередня ефективність обробки даних
SQLite	Зберігання БД в одиночному файлі на диску. Підтримка мови запитів версії SQL92	Не підходить для великих баз даних, тому що важко масштабується

MySQL – це серверна БД, яка надає відмінний набір інструментів для більш простої та захищеної обробки будь-яких видів даних, вбудована в додаток, яким використовується. При використанні MySQL зв'язок з додатком

проводиться за допомогою прямих і функціональних викликів файлів, що підвищує швидкість і продуктивність операцій. MySQL використовують для максимально ефективної та швидкої розробки підсистеми захисту.

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 стає 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 (мал. 2.4):



Мал. 2.4 – Структура двофакторної підсистеми захисту

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

2.2. Розробка Web-сайту готелю

Мова програмування це основа будь-якого програмного забезпечення. У сучасності існує кілька мов програмування, які відрізняються між собою.

Протягом 60 років за які розвивались ЕОМ були розроблені різні мови програмування, більшість з яких використовуються і зараз, і які можуть бути класифіковані за такими ознаками:

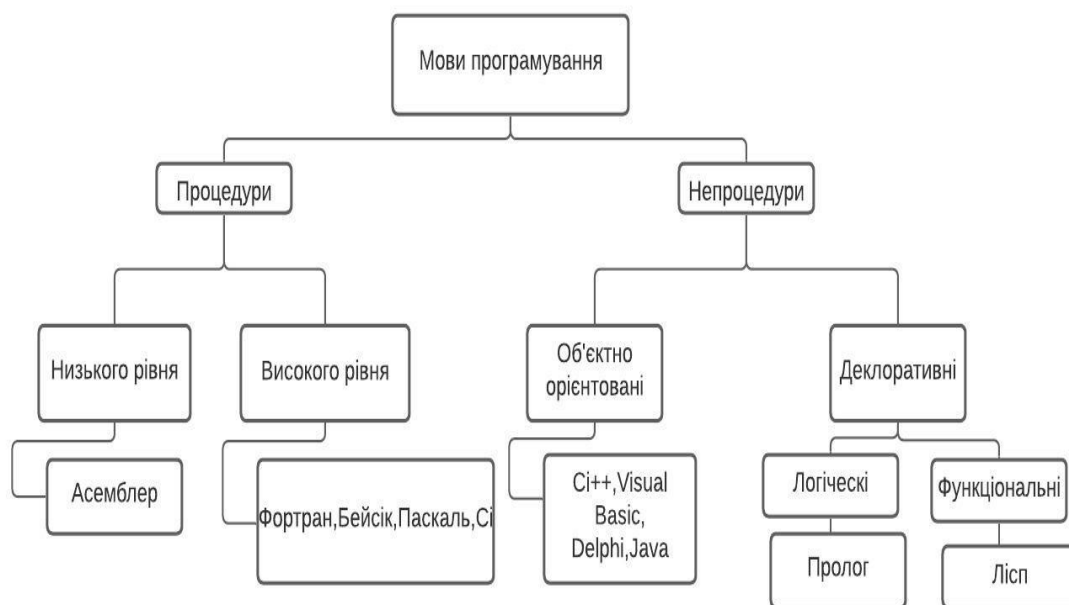
еволюційна - покоління мов;

функціональна – за виконуваними функціями (логічні, математичні описові,);

рівень мови - тобто рівень узагальнення в словах-операторах мови (низького, середнього, високого ...);

область використання - тобто де застосовується мова (системні, мережеві, вбудовані та ін.).

Методи, які використовуються при написанні програм, зумовлюють відмінності мов програмування. (див. мал. 2.5).



Мал.2.5 – Класифікація мов програмування

Мови програмування класифікують за основними функціональними групами:

процедурні;
об'єктно-орієнтовані;
функціональні;
логічні.

Можливість створення в межах програми локальних змінних, функцій і процедур – це процедурне програмування. Будь-яка програма містить код для обробки даних і оперує цими даними. Пріоритет залишається за кодом для обробки даних у мовах, що підтримують парадигму процедурного програмування.

Функціональні та логічні мови програмування належать до декларативних мов. Функціональне програмування – це спосіб складання програм, в яких реалізується єдина дія - виклик функції. Пам'ять, як місце для зберігання даних, у функціональному програмуванні не використовується. Також, не використовуються цикли, оператори присвоювання та проміжні змінні. Вираз це ключове поняття в функціональних мовах. Послідовністю опису функцій і виразів являється програма, написана на функціональній мові.

Об'єктно-орієнтоване програмування (ООП) [11] – це візуалізація всієї програми або її частин об'єктами. В ООП тип об'єкта називається класом. З класів можна створювати екземпляри та оголошувати або успадковувати їх. Екземпляром класу є об'єкт. Об'єкт об'єднує дані і методи для роботи з ним.

Сучасним напрямком системного програмування є непроцедурні мови –, які концентрують увагу розробника не на послідовності дій по їх реалізації, а на описі цілей і правил. Засосування непроцедурної мови забезпечує максимально можливу простоту і зрозумілість, що скорочує терміни розробки і трудомісткість, збільшує надійність ПЗ.

Під час системного аналізу, програми, складені табличною мовою, зручно описують помилки.

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

					ЗКБ-18.125.000	
						36

Парадигма ООП є фундаментальною для багатьох мов сучасності. ГО-мови: C ++, C #, Java, JavaScript, Python[12], PHP, Ruby, Swift та ін. Мови для Android, веб і iOS розробки це топові мови. На мовах з іншими парадигмами можна писати за принципами ООП.

Об'єкти в програмі можуть виступати як абстрактні сутності, або уособлювати об'єкти фізичного середовища. Бізнес-процеси теж можна уявити як взаємодію і стан об'єктів. Зустрічаються такі приклади, як будівництво міста або установи та ін. Навколо ми бачимо об'єкти, які взаємодіють відповідно до субординації, та їх класи об'єктів.

Очевидними плюсами ООП є:

- легкість при модифікації (при грамотному аналізі і проектуванні);
- полегшена розробка завдяки більш легкій розширюваності;
- можливість повернення до попередніх версій;
- зменшений обсяг інформації та скорочена кількість міжмодульних викликів, які передаються між модулями;
- збільшення показника використання коду.

Лідерами серед об'єктно-орієнтованих мов заслужено вважаються PHP, Java, Python.

Спробуємо визначити оптимальний варіант для розробки підсистеми захисту Web-сайту готелю.

Python та Java.

Порівняємо інтерпретований Python із статично типізованою Java. Відмінність Python полягає у тому, що набір інструкцій не потрібно компілювати перед запуском, тоді як написана на Java програма перед інтерпретацією в машинний код, має бути трансльована в байт-код JVM. Тобто Java потребує більше часу для запуску, але менше часу на виконання, тоді як Python швидше запускається і довше виконується.

Філософія і функціональність Python-а дають йому перевагу в нових ІТ-областях перед усіма іншими мовами, тому можна зпрогнозувати збереження такої тенденції і надалі.

розробці сайтів, то можна виділити два найбільш використовуваних фреймворки – Flask та Django.

Flask і Django – обидва великі, розширювані Web-фреймворки, які відрізняються в масштабі відповідальності. Основними відмінностями між ними є:

форми – форми Django інтегруються з ORM і адмінкою сайту; Flask за замовчуванням не підтримує форм;

об'єкт Request - Django за потреби передає запит, а Flask користується локальними потоками;

бази даних – Django, на відміну від Flask, доступний з вбудованою ORM, і системою міграції, яка може керувати базами даних і використовує патерн MVC;

аутентифікація – додаток Django, який реалізує призначене для користувача керування і привілеї, використовуючи шифрування PBKDF2 з хешем SHA256 для захисту даних. Інструмент вашої власної реалізації від Flask – лише безпечні куки;

панель адміністратора – для управління даними додатка Django містить повністю інтегрований адмін-інтерфейс. У Flask такі функції відсутні, але Flask-Admin – досить відомий плагін, який використовується для створення аналогічного адміністративного інструменту.

Django простий та швидкий у створенні веб-додатків, а Flask більш чіткий у налаштуванні додатку, хоча знижує швидкість розробки. В Django є багато додаткових бібліотек, найголовнішою є ORM[13], що полегшує роботу з базою даних.

В арсеналі ORM є інструмент для управління міграціями та автоматичної генерації баз даних.

З Django ORM при певному налаштуванні та розширенні можна отримати чистий SQL, якщо виникає необхідність обійти обмеження.

Вважаємо, що слід використовувати саме веб-фреймворк Django з метою забезпечення найбільш ефективної розробки.

					ЗКБ-18.125.000	
						39

Створення веб-сторінки здійснюється за допомогою верстки – HTML 5, та таблиці каскадних стилів – CSS 3, які будуть використані у розробці веб-орієнтованих систем.

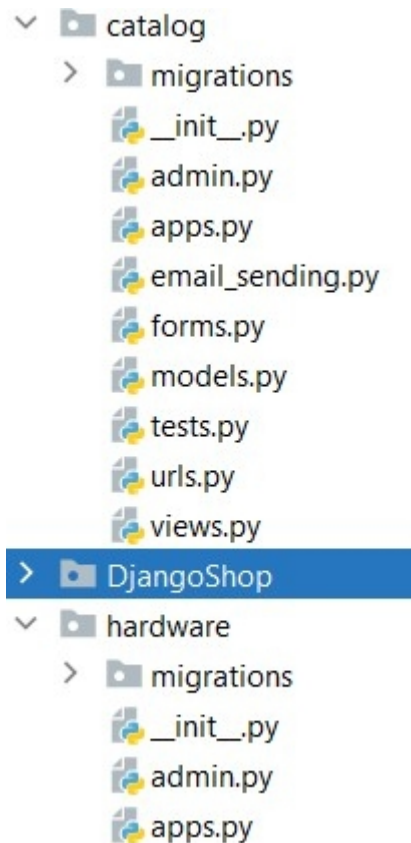
У таблиці 2.2 наведено повний перелік інструментів для реалізації підсистеми захисту Web-сайту готелю.

Таблиця 2.2 – Перелік інструментів для реалізації підсистеми захисту Web-сайту готелю

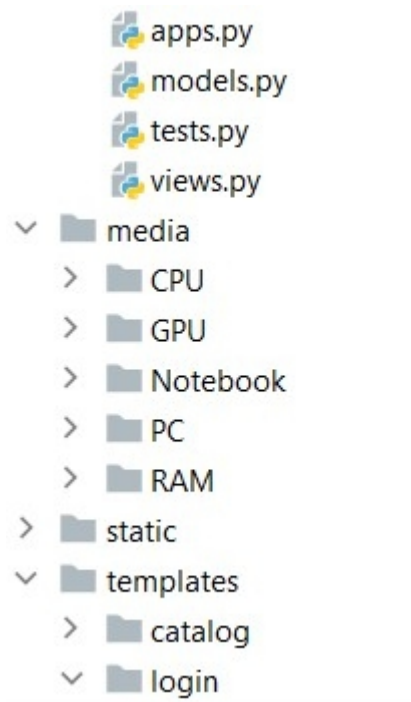
№	Компонент	Обраний інструмент
1	СУБД	SQLite 3
2	Мова програмування	Python(Django)
3	Патерн проектування	MVC
4	Парадигма програмування	ООП
5	Реалізація інтерфейсу сайту	HTML/CSS

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 виглядатиме наступним чином (мал. 2.7-2.8):



Мал.2.7 – Структура розробленого проекту.



Мал.2.8 – Структура розробленого проекту.

При використанні Django, для покращення структуризації системи, необхідно поділяти логічні елементи додатку на apps (додатки), що зроблено і у даному випадку, «catalog» - структурний додаток, що створено для реалізації

моделей, представлень та налаштувань маршрутизації для системи Web-сайту готелю. Така логіка дій необхідна для того, щоб у майбутньому, при спробі розширити проект, мати змогу безпечно створювати нові додатки та не чіпати вже існуючі.

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

Наступним етапом є створення додатку «catalog», який створюється завдяки команді manage.py startapp «назва». Після даних маніпуляцій, структура проекту виглядатиме як на мал. 2.7.

Кожен додаток має власні налаштування маршрутизації, які 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('one_room/', views.OneRoomListView.as_view(), name='one_room_list'),
    path('with_kitchen/', views.WithKitchen ListView.as_view(), name='with_kitchen_list'),
    path('with_sea_views/', views.WithSeaViewsListView.as_view(), name='with_sea_views_list'),
    path('pc/', views.PCListView.as_view(), name='pc_list'),
    path('economy_class/', views.EconomyClassListView.as_view(), name='economy_class_list'),
    path('one_room/<str:slug>/', views.OneRoomDetailView.as_view(), name='one_room'),
    path('with_kitchen/<str:slug>/', views.WithKitchenDetailView.as_view(),
name='with_kitchen'),
    path('with_sea_views/<str:slug>/', views.WithSeaViewsDetailView.as_view(),
name='with_sea_views'),
    path('pc/<str:slug>/', views.PCDetailView.as_view(), name='pc'),
    path('economy_class/<str:slug>/', views.EconomyClassDetailView.as_view(),
name='economy_class'),
    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')
```

					ЗКБ-18.125.000	
						42

```

        template_name = 'login/signup.html'
class WithKitchenListView(generic.ListView):
    template_name = 'catalog/with_kitchen_list.html'
    queryset = WithKitchen.objects.all()
    context_object_name = 'with_kitchens'

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

    def get_queryset(self):
        title = self.request.GET.get('title')
        products = []
        if title:
            if WithKitchen.objects.filter(title__contains=title):
products.append(WithKitchen.objects.filter(title__contains=title))
            if WithSeaViews.objects.filter(title__contains=title):
products.append(WithSeaViews.objects.filter(title__contains=title))
            if OneRoom.objects.filter(title__contains=title):
products.append(WithSeaViews.objects.filter(title__contains=title))
            if PC.objects.filter(title__contains=title):
                products.append(PC.objects.filter(title__contains=title))
            if EconomyClass.objects.filter(title__contains=title):
                products.append(EconomyClass.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, які видають шаблону запитувані користувачем динамічні дані, завдяки Django ORM.

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

```
class AbstractProduct(Model):
```

					ЗКБ-18.125.000	
						43

```

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 за допомогою HTML документів, повний перелік яких представлено у додатку Б.

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

```

@admin.register(WithSeaViewsBus)
class WithSeaViewsBusAdmin(admin.ModelAdmin):
    pass

```

```

@admin.register(PC)

```

					ЗКБ-18.125.000	
						44

```
class PAdmin(admin.ModelAdmin):
    pass
```

2.3. Розробка двофакторної системи захисту

Впровадження двофакторної системи захисту передбачає модифікацію LoginView та SignupView, які забезпечують реєстрацію та авторизацію.

В алгоритмі двофакторної системи автентифікації можна виділити такі складові:

користувач вводить дані для автентифікації;

проводиться перевірка валідності введених даних;

якщо дані валідні, користувач отримує на пошту лист, що містить унікальний ключ, який користувач має ввести на сторінці підтвердження другого етапу автентифікації;

якщо код введено вірно – користувач авторизований;

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
```

					ЗКБ-18.125.000	45

Висновки по 2 розділу

У даному розділі проведено роботу над розробкою Web-сайту готелю та підсистемою його захисту. Здійснено порівняння патернів проектування, виділено плюси та мінуси кожного з них, виділено патерн MVC, прийнято рішення дотримуватись саме цього патерну.

Проведено аналіз існуючих СУБД, приведені плюси та мінуси таких СУБД як PostgreSQL, MySQL, SQLITE, MsSQL Server.

Вирішено використовувати MySQL, оскільки дана СУБД вважається найбезпечнішою.

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

Розроблено додаток Web-сайту готелю за допомогою виділених інструментів.

Проведено аналіз двофакторної системи захисту, алгоритму PBKDF2, впроваджено дані системи у додаток Web-сайту готелю.

РОЗДІЛ 3. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ПІДСИСТЕМИ ЗАХИСТУ WEB-САЙТУ ГОТЕЛЮ

3.1. Розгортання моделі експерименту

Перед тим як розпочати роботу з Django, перш за все необхідно встановити інтерпретатор Python версії 3.8 або вище. Для його встановлення слід 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.

На цьому етапі можна задати шлях, по якому буде встановлений інтерпретатор. Можна залишити за замовчуванням C: \ Users \ [ім'я_користувача] \ AppData \ Local \ Programs \ Python \ Python36 \. Крім того,

					ЗКБ-18.125.000	47

у самому низу потрібно зробити позначку "Додати Python 3.9 до PATH", щоб додати шлях до інтерпретатора в змінних середовища.

Є різні способи встановлення Django. Можемо порекомендувати наступний. Для установки буде потрібен пакетний менеджер `pip`. Зазначений менеджер 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 його рекомендується використовувати, оскільки воно дозволяє створити безліч віртуальних середовищ Python на одній операційній системі. Завдяки віртуальному середовищу додаток може запускатися незалежно від інших додатків на Python.

Варто зазначити, що запускати додатки на Django можна і без віртуального середовища. В такому разі всі пакети Django встановлюються глобально.

Першим кроком для роботи з віртуальним середовищем є встановлення пакету `virtualenv` за допомогою наступної команди:

```
-pip install virtualenv
```

Надалі створимо віртуальне середовище. 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'язане з ним віртуальне середовище треба активувати. Наприклад, активуємо зазначене вище створене середовище, яке 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 здійснюватися 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 в такий же спосіб запустити його.

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

```
-pip install Django.
```

Дана команда є завершальним кроком для встановлення Django.

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

```
-python manage.py runserver.
```

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

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



The image shows a 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'. The form itself is light gray and contains two input fields: 'Username' and 'Password'. Below the 'Password' field is a blue button with the text 'Log in'.

Мал.3.1 – Сторінка авторизації

Оскільки жодного користувача ще не створено, потрібно виконати команду

-python manage.py createsuperuser.

Ця команда викликає скрипт створення супер-користувача (адміністратора), що виглядає наступним чином (мал 3.3):

```
C:\Users\Podja\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\Podja\Desktop\4 курс\DjangoShop-master>
```

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

Після даного кроку потрібно авторизуватись. На цьому етапі система захисту Web-сайту готелю та сам Web-сайт готові до використання.

3.2. Дослідження та порівняння налаштувань безпеки сайту при різних вразливостях

Спочатку необхідно перевірити працездатність алгоритму шифрування паролів PBKDF2 з хешем SHA256, що автоматично впроваджено Django.

Переглядаючи облікові записи у адміністративній панелі зрозуміло, що паролі зашифровані(мал. 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 .

Мал.3.3 – Алгоритм PBKDF2 у системі Django

Перевіримо роботу двофакторної системи захисту.

Перш за все, зареєструємо користувача(мал. 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

Мал.3.4 – Процес реєстрації користувача

Зареєстровано користувача з логіном yurik17021@gmail.com, після натискання на кнопку «Реєстрація», при успішному введенні даних відбувається переадресація користувача до сторінки авторизації(мал. 3.5):

Username:

Password:

Мал.3.5 – Авторизація користувача(Перший етап)

Після коректного введення даних користувач отримує на електронну пошту лист з кодом, який потрібно ввести на наступній сторінці (мал. 3.6):

Введіть код з листа на пошті:

Введіть код:

Мал.3.6 – Авторизація користувача (Другий етап)

Якщо код введено вірно, користувача буде переадресовано на головну сторінку.

Надалі 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. Спочатку спробуємо декілька раз авторизуватися з неправильним паролем, імітуючи атаку брутфорс (рис 3.7)

Burp Suite Community Edition v2020.9.2 - Temporal Project

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Intercept HTTP history WebSockets history Options

Request to http://127.0.0.1:8000

Forward Drop Intercept is on Action Open Browser

Comment this item

Raw Params Headers Hex

Proxy Raw In Actions

```

1 GET /code/ HTTP/1.1
2 Host: 127.0.0.1:8000
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4103.121 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 Referer: http://127.0.0.1:8000/login/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.8,en;q=0.7
14 Cookie: csrfToken=73h1sdy7p0b5allew7Pa6G0m8J70B0Ww0qfacE26q2i1AP30304U0B0H0d0; sessionId=5engccu5v9f7Vb2m763kbu647vbnx
15 Connection: close
16
17

```

0 matches

ЗЛОВМИСНИК НЕ МАТИМЕ ДОСТУПУ ДО АКАУНТУ, НЕ МАЮЧИ ДОСТУПУ ДО ПОШТИ КОРИСТУВАЧА.

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

- збільшення довжини та псевдовипадковості коду підтвердження;
- збільшення кількості ітерацій алгоритму PBKDF2;
- використання більш надійної робочої пошти.

Висновки по 3 розділу

В даному розділі наведена перевірка працездатності розробленої підсистеми захисту. Створено інструкцію розгортання розробленого Web-сайту готелю 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.

ВИСНОВКИ

У випускній кваліфікаційній роботі виконано актуальне завдання з розробки підсистеми захисту сайту готелю. Дослідження ринку програмного забезпечення показало, що в сучасних умовах існує велика кількість розробок у сфері безпеки веб-ресурсів. При виборі управління даними були досліджені, проаналізовані та виявлені переваги і недоліки таких баз даних, як MySQL, SQL Server, PostgreSQL, SQLite. В якості СУБД було акцентовано увагу на MySQL, а в якості мови програмування для розробки програми, був обраний Python, а Django в якості фреймворку.

Інтерфейс користувача при розробці інформаційної системи був використаний за стандартом GUI. Його можна змінити, коли змінюються вимоги до певної області. Розглянутий тип безпеки проекту. Розглянуто двохетапний захист користувачів, а данні захищені алгоритмом PBKDF2 з хешем SHA256. Розроблена інформаційна система не являється повноцінним комерційним додатком, а створена як наукове дослідження. Розроблена система підлягає модифікації та може бути використана в подальших розробках і дослідженнях.

					ЗКБ-18.125.000	
						56

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Безпека інформаційних систем. Лабораторний практикум / Ільїн, Костянтин Іванович, Стьопчкіна Ірина Валеріївна 12-гру-2020 – 60с.
2. Основні принципи інформаційної безпеки відкритих систем. Частина 1. Міри інформації та властивості інформаційних процесів відкритих систем / Кононович, ВолодимирТардаскіна, Тетяна 2006р.
3. NETSCOUT. [Електронний ресурс]. URL: <https://www.netscout.com/threatreport>
4. The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. [Електронний ресурс]. URL: <https://owasp.org/>
5. Інформаційна безпека веб-ресурс https://uk.wikipedia.org/wiki/Інформаційна_безпека
6. ВВЕДЕННЯ В СУЧАСНІ БАЗИ ДАНИХ Навчальний посібник / Демиденко М.А., 2020. - 42 с.
7. SQL підручник – веб сервіс <https://w3schoolsua.github.io/sql/index.html>.
8. Types of Design Patterns - Sameeha Rahman веб ресурс <https://www.freecodecamp.org/news/the-basic-design-patterns-all-developers-need-to-know/>.
9. Let's Learn Python - Basics Trevor Payne відео навчання на сервісі Ютуб - <https://www.youtube.com/watch?v=D48iCw3WWpI>.
10. ORM. Object-Relational Mapping Wiki – веб-ресурс. URL: <https://uk.wikipedia.org/wiki/ORM>
11. Object-Oriented Design Patterns веб-сервіс - <https://pl.cs.jhu.edu/oose/lectures/design-patterns.shtml>
12. ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ(ООП). - Комлев Н., 2014. – 297с.
13. Вивчаємо Python. Програмування ігор візуалізація даних веб-додатки –Ерік Метиз 2016. – 496 с.

					ЗКБ-18.125.000	
						57

14. ORM. Object-Relational Mapping Wiki – веб-ресурс. URL:
<https://uk.wikipedia.org/wiki/ORM>

					ЗКБ-18.125.000	
						58

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 = 'h1rhlqm+rdhaw&j^gkawv1--c@((omvpjn!jsfghwa^@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 = [
```

					ЗКБ-18.125.000	
						59

```
{
    '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/>

					ЗКБ-18.125.000	
						60

```
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 With Kitchen, With sea views, One Room, Economy Class, PC, AbstractProduct
```

```
class Comment(models.Model):
```

```
    product = models.ForeignKey(With Kitchen, 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 WithKitchenComment(Comment):
```

```
    product = models.ForeignKey(With Kitchen, on_delete=models.CASCADE)
```

```
class Meta:
```

```
    verbose_name = 'With Kitchen коментар'
```

```
    verbose_name_plural = 'With Kitchen коментари'
```

					ЗКБ-18.125.000	
						61

```
class WithSeaViewsComment(Comment):
    product = models.ForeignKey(With sea views, on_delete=models.CASCADE)
```

```
class Meta:
    verbose_name = ' With sea views коментар'
    verbose_name_plural = ' With sea views коментари'
```

```
class OneRoomComment(Comment):
    product = models.ForeignKey(One Room, on_delete=models.CASCADE)
```

```
class Meta:
    verbose_name = 'One room коментар'
    verbose_name_plural = 'One room коментари'
```

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

```
class Meta:
    verbose_name = 'PC коментар'
    verbose_name_plural = 'PC коментари'
```

```
class EconomyClassComment(Comment):
    product = models.ForeignKey(Economy Class, on_delete=models.CASCADE)
```

```
class Meta:
    verbose_name = 'Economy class коментар'
    verbose_name_plural = 'Economy class коментари'
```

```
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)
    one_room_cart = models.ManyToManyField(One Room)
    with_kitchen_cart = models.ManyToManyField(With Kitchen)
    with_sea_views_cart = models.ManyToManyField(With sea views)
    economy_class_cart = models.ManyToManyField(Economy Class)
```

```
def get_all_products(self):
    products = [self.pc_cart.get_queryset(), self.one_room_cart.get_queryset(), self.with_kitchen_cart.get_queryset(),
                self.with_sea_views_cart.get_queryset(), self.economy_class_cart.get_queryset()]
    return products
```

Додаток В
Views.py

```
from random import randint
```

```
from django.contrib import auth
from django.contrib.auth.models import User
```

					ЗКБ-18.125.000	
						62

```
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 WithKitchenCommentForm, OneRoomCommentForm, WithSeaViewsCommentForm, PCCommentForm,
EconomyClassCommentForm, EmailConfirm
from catalog.models import WithKitchenComment, WithSeaViewsComment, OneRoomComment, PCComment,
EconomyClassComment, Code, Cart
from hardware.models import WithKitchen, WithSeaViews, OneRoom, PC, EconomyClass
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'
```

					ЗКБ-18.125.000	
						63

```

class WithKitchenListView(generic.ListView):
    template_name = 'catalog/with_kitchen_list.html'
    queryset = WithKitchen.objects.all()
    context_object_name = 'with_kinchens'

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(WithKitchenListView, 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 WithKichen.objects.filter(title__contains=title):
                products.append(WithKichen.objects.filter(title__contains=title))
            if WithSeaViews.objects.filter(title__contains=title):
                products.append(WithSeaViews.objects.filter(title__contains=title))
            if OneRoom.objects.filter(title__contains=title):
                products.append(WithSeaViews.objects.filter(title__contains=title))
            if PC.objects.filter(title__contains=title):
                products.append(PC.objects.filter(title__contains=title))
            if EconomyClass.objects.filter(title__contains=title):
                products.append(EconomyClass.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 WithSeaViewsListView(generic.ListView):
    template_name = 'catalog/with_sea_views_list.html'
    queryset = WithSeaViews.objects.all()
    context_object_name = 'with_sea_views'

class OneRoomListView(generic.ListView):
    template_name = 'catalog/one_room_list.html'
    queryset = OneRoom.objects.all()
    context_object_name = 'one_rooms'

class PCListView(generic.ListView):
    template_name = 'catalog/pc_list.html'
    queryset = PC.objects.all()
    context_object_name = 'pcs'

class EconomyClassListView(generic.ListView):
    template_name = 'catalog/economy_class_list.html'
    queryset = EconomyClass.objects.all()
    context_object_name = 'economy_class`s'

```

					ЗКБ-18.125.000	
						64

```

class WithKitchenDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = WithKitchenCommentForm
    template_name = 'catalog/with_kitchen_detail.html'
    context_object_name = 'with_kitchen'

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

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

    def get_success_url(self):
        return reverse_lazy('catalog:with_kitchen', 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):
        WithKitchenComment.objects.create(author=self.request.user, text=form.cleaned_data['text'], product=self.get_object())
        return super(WithKitchenDetailView, self).form_valid(form)

```

```

class WithSeaViewsDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = WithSeaViewsCommentForm
    template_name = 'catalog/with_sea_views_detail.html'
    context_object_name = 'with_sea_views'

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

    def get_context_data(self, **kwargs):
        context = super(WithSeaViewsDetailView, self).get_context_data()
        context['comments'] = WithSeaViewsComment.objects.filter(product=self.object)
        return context
    else:
        return self.form_invalid(form)

    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:with_sea_views', kwargs={'slug': self.kwargs.get('slug', None)})

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

```

					ЗКБ-18.125.000	
						65

```

class OneRoomDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = OneRoomCommentForm
    template_name = 'catalog/one_room_detail.html'
    context_object_name = 'one_room'

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

    def get_context_data(self, **kwargs):
        context = super(OneRoomDetailView, self).get_context_data()
        context['comments'] = OneRoomComment.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:one_room', kwargs={'slug': self.kwargs.get('slug', None)})

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

class PCDetailView(generic.edit.FormMixin, generic.DetailView):
    form_class = PCCCommentForm
    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))
context = super(PCDetailView, self).get_context_data()
context['comments'] = PCCComment.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):
        PCCComment.objects.create(author=self.request.user, text=form.cleaned_data['text'], product=self.object)
        return super(PCDetailView, self).form_valid(form)

class EconomyClassDetailView(generic.edit.FormMixin, generic.DetailView):

```

					ЗКБ-18.125.000	
						66

```

form_class = EconomyClassCommentForm
template_name = 'catalog/economy_class_detail.html'
context_object_name = 'economy_class'

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

def get_context_data(self, **kwargs):
    context = super(EconomyClassDetailView, self).get_context_data()
    context['comments'] = EconomyClassComment.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:economy_class', kwargs={'slug': self.kwargs.get('slug', None)})

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

```

```

class IndexView(generic.ListView):
    template_name = 'catalog/index.html'
    queryset = EconomyClass.objects.all()    model = Cart

```

```

context_object_name = 'economy_class'

```

```

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 EconomyClass.objects.filter(title=product):
            product = EconomyClass.objects.get(title=product)
            cart.economy_class_cart.add(product)

```

					ЗКБ-18.125.000	
						67

```

elif OneRoom.objects.filter(title=product):
    product = OneRoom.objects.get(title=product)
    cart.one_room_cart.add(product)
elif WithKitchen.objects.filter(title=product):
    product = WithKitchen.objects.get(title=product)
    cart.with_kitchen_cart.add(product)
elif WithSeaViews.objects.filter(title=product):
    product = WithSeaViews.objects.get(title=product)
    cart.with_sea_views_cart.add(product)
return JsonResponse({'message': 'Зарезервовано'})

```

					ЗКБ-18.125.000	
						68