МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ ЗАХИЩЕНОЇ КОРПОРАТИВНОЇ ЛОКАЛЬНОЇ МЕРЕЖІ ПІДПРИЄМСТВА»**

Виконала: студентка 2 курсу, групи 8.1228

спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)

освітньої програми комп'ютерні науки
(назва освітньої програми)

Є.О. Кушнір
(ініціали та прізвище)

Керівник завідувач кафедри комп'ютерних наук, доцент, к.т.н Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри програмної інженерії, доцент, к.т.н. Чопоров С.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки

(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук, к.т.н., доцент

_____ Борю С.Ю.
(підпис)

" 29 " травня 2019 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кушнір Єлизаветі Олександрівні
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка та програмна реалізація проекту захищеної корпоративної локальної мережі підприємства

керівник роботи Борю С. Ю. к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » 05 2019 року № 811-с

2. Строк подання студентом роботи 26.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка та програмна реалізація проекту захищеної корпоративної локальної мережі підприємства

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.05.2019

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	29.05.2019	
2.	Збір вихідних даних.	24.06.2019	
3.	Обробка методичних та теоретичних		
	джерел.	15.07.2019	
4.	Розробка першого та другого розділу.	09.09.2019	
5.	Розробка третього розділу.	07.10.2019	
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	16.12.2019	
7.	Захист кваліфікаційної роботи.	14.01.2020	

Студент _____ Є.О. Кушнір
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка та програмна реалізація проекту захищеної корпоративної локальної мережі підприємства»: 63 с., 23 рис., 1 табл., 25 джерел, 4 додатки.

ЗАХИЩЕНА МЕРЕЖА, КЛІЄНТ-СЕРВЕРНИЙ ДОДАТОК, КОНФІГУРАЦІЙНИЙ ФАЙЛ, ЛОКАЛЬНА КОРПОРАТИВНА МЕРЕЖА, ОБЛІК КОРИСТУВАЧІВ, API, VPN СЕРВЕР.

Об’єкт дослідження: спрощення доступу та взаємодії співробітника і локальної мережі підприємства шляхом створення платформи генерації конфігураційних OpenVPN файлів, і обліку користувачів даної мережі.

Предмет дослідження: клієнт-серверний додаток для зберігання і зміни даних про клієнтів, сервісах компанії, конфігураційних файлів і їх взаємодії.

Мета роботи: проектування та розробка програмної системи, що дозволяє спроектувати захищену локальну мережу підприємства та створити клієнт-серверний проект (додаток). У додатку зберігається інформацію про кожного користувача цієї системи та всіх наявних мережевих сервісів, та створює користувачу дозвіл у вигляді конфігураційного файлу до необхідних йому доступах.

Робота присвячена розробці клієнт-серверної системи, що спілкується з сервером за допомогою API. Розроблена програмна система дозволяє спростити функціонал отримання доступу до локальної мережі підприємства ззовні, а так само об'єднати доступи сервісів з декількох офісів одночасно. Автоматизує генерацію індивідуальних конфігураційних файлів для співробітників і полегшує роботу менеджеру і системного адміністратора компанії. У проекті використовується технологія OpenVPN сервісу для настройки мережі, а так само платформа Yii2 для створення безпосередньо системи обліку співробітників і їх доступів.

SUMMARY

Master's qualification thesis “Development and Software Implementation of the Project of a Protected Corporate Local Network of the Enterprise”: 63 pages, 23 figures, 1 table, 25 references, 4 attachments.

SECURED NETWORK, CLIENT-SERVER APPLICATION, CONFIGURATION FILES, CORPORATE LOCAL NETWORK, USER ACCOUNTING, API, VPN SERVER.

Research object: simpler access and interaction between the employee and the enterprise local network by creating a platform to generate configuration OpenVPN files, and users accounting of this network.

Research subject: client-server application for storing and changing data about clients, company services, configuration files, and their interaction.

Purpose of the paper: the design and development of a software system, enabling to design of an enterprise secure local network and create a client-server project (application). The application saved information about each user of this system and all available network services and create permission to user in the file configuration to access him.

This qualifying paper is dedicated to the client-server system development that communicates with the server using the API. The developed software system makes it possible to simplify the functionality of enterprise access to the local network outside and to combine the services access from several offices at the same time. Keeps a record of the enterprise employees and the permission's availability to certain resources for each of them, depending on the position held by the employees. Service technology OpenVPN to configure the network and the Yii2 platform for the direct creation of an employee accounting system and its access are used in the project.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЛКМ	Локальні комп’ютерні мережі
КЛКМ	Корпоративні локальні комп’ютерні мережі
ЗКЛМ	Захищенні корпоративні локальні мережі
ПК	Персональній комп’ютер
ПЗ	Програмне забезпечення
ОС	Операційна система
ГБ	Гігабайти
ГГц	Гігагерци
Гбіт / с	Гігабіт у секунду
БД	База даних
СУБД	Система управління базами даних
ІС	Інформаційна система
СА	Сертифікатна аутентифікація

ВСТУП

Система безпеки була і залишається актуальною та дуже важливою в роботі корпоративних систем. Особливо це важливо компаніям, які не мають постійного офісу і ведуть онлайн роботу.

Для роботи в офісах використовують особисті локальні мережі. Це робиться в першу чергу для забезпечення конфіденційності та безпеки внутрішніх даних і інформації про саму компанію або вироблених нею продуктах. Тому всі пристрої, що використовуються в роботі фірми повинні бути під'єднані до єдиної мережі. Доцільно поєднувати їх з допомогою закритого з'єднання і обмежувати доступ ззовні. При цьому має сенс шифрувати маршрути, для виключення перехоплення даних на кінцевому вузлі сполуки.

Але кожен співробітник також повинен мати доступ до цієї мережі ззовні для віддаленої роботи або термінових рішень робочих проблем, якщо він знаходиться поза офісом, наприклад доступ до GitLab – системі керування репозиторіями програмного коду для Git. Також, компанії все частіше реалізують практику “віддаленого співробітника” або фрілансера.

Для того, щоб співробітникові був дозволений доступ в локальну корпоративну мережу, було прийнято рішення, що повинен існувати спеціальний VPN сервер, який буде забезпечувати маршрутизацію адреси конкретного пристрою в мережу. У ході порівняння був обраний OpenVPN - протокол, який відповідає за підтримання комунікації між клієнтом і сервером шляхом створення захищеного "тунелю" між VPN-клієнтом і VPN-сервером. Щоб сервер розпізнав доступ для пристрою, необхідно, щоб на самому пристрої був конфігураційний файл з налаштуваннями доступів і списком дозволів.

Однак було б не доцільно і не безпечно давати всім співробітникам один конфігураційний файл на всіх. Тоді б, у випадку витоку інформації було б складніше відстежити винного. Також, потрібно враховувати, що у людей можуть бути різні рівні доступів до різних ресурсів компанії, декому можна давати доступ, а декому ні.

Для цього кожен співробітник повинен мати особистий конфігураційний файл з списком дозволених йому доступів.

Зазвичай налаштуванням системи доступів, а також створенням таких конфігураційних файлів на фірмі займаються системні адміністратори. Однак, працюючи у компанії з багатьма співробітниками, це завдання забирає у них багато часу, так як потрібно не тільки створювати такі файли вручну, але й знаходити і змінювати існуючі. Відомості про зміни конфігураційного файлу для певної людини надходять від менеджера, який і передає цю інформацію до системного адміністратора.

Для прискорення і спрощення цього процесу, було запропоновано створити внутрішню систему обліку користувачів з автоматичною генерацією конфігураційних файлів безпосередньо на сервері за допомогою Application Programming Interface (API) – інтерфейсу програмування додатків.

Відповідно, необхідно створити OpenVPN сервер з унікальними налаштуваннями шифрування, доступів і маршрутизації, а так само клієнт-серверний додаток для забезпечення користувачів доступами до певних сервісів у вигляді файлів з конфігураційними налаштуваннями.

1 ОСОБЛИВІСТЬ РОЗРОБКИ ТА ЕКСПЛУАТАЦІЇ КОРПОРАТИВНІЙ ЛОКАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ

1.1 Основні види КЛКМ та їх особливості

У кожному офісі є потреба у правильно налаштованої комп’ютерної локальній мережі (ЛКМ). ЛКМ створюється для об'єднання в робочі групи до декількох десятків, сотень комп'ютерів в рамках однієї, двох або кількох організацій. Зокрема, у всіх компаніях використовуються переваги мереж, що об'єднують комп'ютери різних відділів, які в свою чергу теж знаходяться в локальній мережі в рамках того ж чи іншого офісу. Використання можливостей комп'ютерних мереж, зокрема локальної мережі, викликано практичною потребою швидкого обміну різнорідною інформацією, одночасного використання прикладних програм, спільного використання ресурсів комп'ютерів і периферійного обладнання, підключеного до мережі, і тому подібне.

До недавнього часу найбільшою популярністю користувалися системи Local Area Network (LAN), які об'єднують обмежена кількість ПК. Вони забезпечують максимальну швидкість обміну файлами і абсолютну безпеку інформації, так як її потоки не потрапляють в загальний доступ. Використання структур цього типу є безкоштовним на програмному рівні, але на апаратному рівні треба закуповувати обладнання, і чим далі офіси знаходяться один від одного, тим сильніше збільшується вартість. Вартість побудови мережі зростає при поєднанні різних офісів в межах району чи міста. До мінусів LAN можна віднести не можливість підключення віддалених користувачів.

Гідною альтернативою стали віртуальні мережі - Virtual Private Network (VPN), які будуються поверх глобальних мереж WAN (Wide Area Network), що

охоплюють велику кількість ПК і комп'ютерних систем по всій планеті. До їх безперечних переваг відносяться простота (а відповідно, і невисока вартість) побудови, можливість підключення безлічі абонентів, що знаходяться в різних кінцях світу, і безпеку передачі даних. За результатами досліджень, проведених Forrester Research Inc. і Infonetics Research, витрати на використання та обслуговування VPN набагато нижчі, ніж логістичних структур, побудованих за технологією LAN. Як правило, вибір VPN починається з вибору протоколу тунелювання. Найбільш поширені - L2TP / IPsec, OpenVpn, WireGuard.

Layer 2 Tunneling Protocol (L2TP) - це протокол тунелювання другого рівня. Він є розширенням протоколу тунелювання «точка-точка» (PPTP), використовуваного постачальниками інтернет-послуг (ISP) для забезпечення роботи віртуальної приватної мережі VPN через Інтернет.

L2TP/IPsec має такі особливості:

– він повільніше інших через подвійне інкапсулювання (створюється IPsec-туннель, а дані ходять через L2TP);

– використовує стандартні порти, і тому його легко може заблокувати інтернет-провайдер або системний адміністратор;

– операційні системи мають вбудовану підтримку цієї технології, немає необхідності встановлювати додаткове ПО;

– при його правильному налаштуванні немає інформації про можливості розшифрувати дані.

Цей протокол вже є застарілою технологією и має дуже повільну швидкість.

OpenVPN – безкоштовне рішення з відкритим вихідним кодом, яке, за визнанням більшості фахівців, є найкращим на сьогоднішній день для створення приватної віртуальної мережі (VPN). Це реалізація технології віртуальної приватної мережі для створення зашифрованих каналів типу “точка-точка” або “сервер-клієнти” між комп'ютерами. Вона дозволяє

встановлювати з'єднання між комп'ютерами, що знаходяться за NAT і мережевим екраном, без необхідності зміни їх налаштувань.

Особливості OpenVPN:

– не входить до складу стандартних дистрибутивів сучасних операційних систем, тому вимагає установки додаткового програмного забезпечення;

– при правильному налаштуванні його не зможуть розшифрувати ні спецслужби, ні зловмисники;

– при не стандартних налаштуваннях складно блокується.

OpenVPN вимагає установки додаткового програмного забезпечення, але це перевірене часом програмне забезпечення з відкритим вихідним кодом, встановлення і налаштування якого не створить проблем навіть новачкові. OpenVPN працює на всіх сучасних операційних системах: Windows, macOS, Linux, Android, iOS.

Для розробників ядра Linux представлена нова реалізація інтерфейсів для шифрованих і аутентіфіційованих каналів. WireGuard - новий варіант реалізації VPN, що поєднує простоту реалізації (близько 4 тисяч рядків) з повним функціоналом перевірених криптографічних алгоритмів. На думку автора Джейсона Доненфілда (Jason A. Donenfeld), глави компанії Edge Security і її фахівця з безпеки, його реалізація позбавлена ускладнень, властивих таким проектам, як xfrm і OpenVPN.

Проект WireGuard розвивається кілька років і вже пройшов стадію рецензування криптографії, що дозволяє говорити про його впровадженні в ядро і публічному тестуванні. Після проходження тестування в ядрі Linux передбачається перенести напрацювання в інші ОС.

Також є спосіб SSH тунелювання - це тунель, який створюється за допомогою SSH з'єднання і використовується для шифрування даних. Використовується для того, щоб убезпечити передачу даних в інтернеті. Особливість полягає в тому, що не зашифрований трафік будь-якого протоколу шифрується на одному кінці SSH з'єднання і розшифровується на іншому.

Отже, цей спосіб надійний, захищений, але потрібні знання за налаштуванням і працює він тільки на платформі Linux/Unix, а на серверах Windows необхідно встановлення додаткових програм, та має продуктивність на середньому рівні. Також потрібні кошти для моніторингу і розуміння, що він працює.

Для вибору оптимальної технології для конкретного випадку завдання роботи, було прийнято рішення скласти порівняльну характеристику даних способів з метою визначити найбільш відповідний. Результати порівняння можна побачити в таблиці 1.1.

Таблиця 1.1 – Порівняння технологій при побудові об'єднання офісів або віддаленого користувача до офісу

	Вартість	Актуальність	Кросплатформеність	Безпечність даних	Зручність користувача
LAN	дорого	актуальний	так	ні	висока
Open VPN	дешево	актуальний	так	так	висока
SSH тунель	дешево	актуальний але вимогливий	ні	так	середня
Wireguard	дешево	новий	ні	так	середня

Тому, виходячи з таблиці, у цьому проекті логічно та доцільно використовувати саме технологію OpenVPN серверу, який на даний момент має найбільшу кількість переваг, таких як безпечність, гнучкість, економічність та має дружелюбний інтерфейс.

1.2 Основні способи захисту КЛКМ

Надання безпеки в комп'ютерних мережах - це основна умова захисту конфіденційних даних від різного роду погроз, таких як шпигунство, знищення файлів та інші несанкціоновані дії. Кожен з перерахованих факторів може негативно вплинути на коректне функціонування локальної та глобальної мережі, що, у свою чергу, нерідко призводить до розголошення або втрати конфіденційної інформації.

Однією з поширених мережевих загроз є несанкціонований доступ ззовні, причому не тільки умисний, але і випадковий. Прогресивні методи захисту інформації при використанні комп'ютерних мереж в більшості своїй спрямовані на запобігання всіляких факторів, що ведуть до втрати або крадіжки конфіденційної інформації.

До ефективних засобів захисту можна віднести адміністрування та грамотний розподіл повноважень і обмеження доступів між співробітниками. З метою запобігання несанкціонованому доступу до секретних файлів застосовують криптографічні методи захисту, які передбачають шифрування вмісту файлів за допомогою електронних ключів.

Також, для того, щоб реалізувати безпечний зовнішній доступ користувачеві у локальну мережу, необхідно врахувати, що корпоративна локальна мережа використовує сірі IP адреси для внутрішньої взаємозв'язку, а для виходу в Інтернет використовується технологія NAT та білі IP-address. Відповідно при створенні файлу конфігурації VPN сервер повинен враховувати цю особливість при налаштуваннях маршрутизації.

IP-address V4 поділяються на:

– білі (публічні / глобальні / зовнішні);

– сірі (приватні / локальні / внутрішні).

Публічні (глобальні) IP-адреси маршрутизируются в Інтернеті, на відміну від приватних адрес, тобто у мережі Інтернет використовуються саме білі адреси.

Наявність публічного IP-адреси на роутері або комп'ютері дозволяє організувати власний сервер (VPN, FTP, WEB та ін.), віддалений доступ до комп'ютера, камерам відеоспостереження, і отримати до них доступ з будь-якої точки глобальної мережі. Приклад роботи мережі з білою адресою можна подивитися на рис. 1.1.

У зв'язку з тим що "білих" IP-адрес існує обмежена кількість, а зростання числа користувачів Інтернету збільшується, інтернет-провайдери все частіше використовують приватні ("сірі") IP-адреси, які призначаються абонентам. Приватні внутрішні адреси не маршрутизируются в Інтернеті і на них не можна відправити трафік з Інтернету, вони працюють тільки в межах локальної мережі.

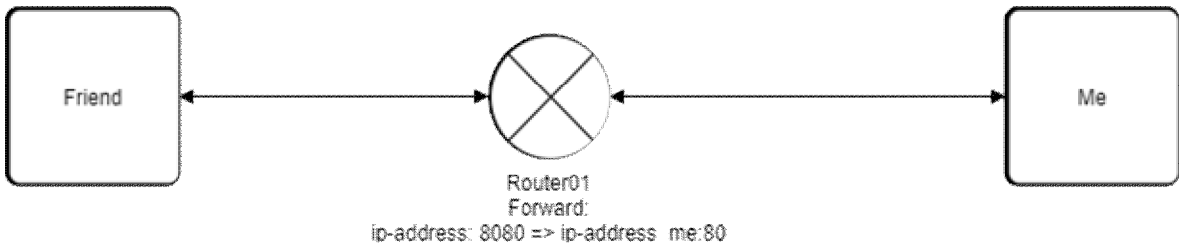

Рисунок 1.1 – Приклад пробросу трафіку якщо у клієнта або офісу є біла адреса

У більшості провайдерів можна замовити білу адресу і за допомогою даного IP-address можна виконати Forward у внутрішню мережу на певний вузол. Наприклад, використовується, щоб прокинути web-port на внутрішній сервер.

Безпосередньо доступ до мережі Інтернет, використовуючи приватний IP-адреса, неможливий. У цьому випадку зв'язок з Інтернетом здійснюється через NAT (трансляція мережевих адрес замінює приватний IP-адреса на публічний). Приватні IP-адреси в межах однієї локальної мережі повинні бути унікальні і не можуть повторюватися.

Що стосується безпеки в Інтернеті, то використання "сірого" IP-адресу більш безпечно, ніж використання "білого" IP-адресу, тому що "Сірі" IP-адреси

не видно безпосередньо в Інтернеті і вони знаходяться за NAT, який також забезпечує безпеку локальної мережі.

Приватні внутрішні адреси не маршрутизируются в Інтернеті і на них не можна відправити трафік з Інтернету, вони вони визначені стандартом використання тільки в локальній мережі і працюють тільки в її межах.

До приватних "сірих" адресами відносяться IP-адреси з наступних підмереж:

– від 10.0.0.0 до 10.255.255.255 з маскою 255.0.0.0 або / 8;

– від 172.16.0.0 до 172.31.255.255 з маскою 255.240.0.0 або / 12;

– від 192.168.0.0 до 192.168.255.255 з маскою 255.255.0.0 або / 16.

Це зарезервовані IP-адреси. Такі адреси призначені для застосування в закритих локальних мережах, розподіл таких адрес ніким не контролюється. На рис. 1.2. показаний неможливості пов'язати мережі офісів без білого адреси.

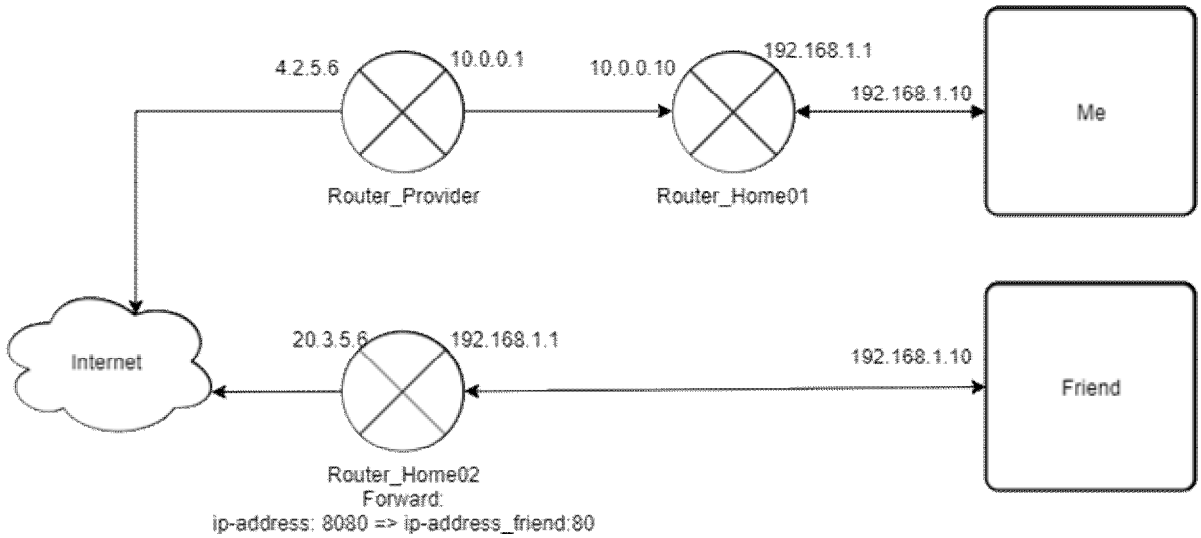

Рисунок 1.2 – Приклад роботи мережі з сірими адресами

Якщо немає можливості придбати у провайдера білий IP-address, то і рис.немає можливості зробити forward портів на внутрішній сервер. Віддалений клієнт не зможе підключиться до нашого адресою, оскільки він є сірим

(локальним) для провайдера, тому клієнт не зможе пройти далі зовнішнього білого адреси.

Тому під’єднатися до корпоративної локальної мережі ззовні можна тільки маючи конфігураційний файл, який належить відповідному OpenVPN серверу, що знаходиться в цій самій мережі.

1.3 Огляд маршрутних потоків та захист при маршрутизації

Локальні мережі підприємств дуже часто підключаються до мережі Інтернет. Для захисту локальних мереж компаній, як правило, застосовуються міжмережеві екрани - брандмауери (firewalls). Екран - це засіб розмежування доступу, яке дозволяє розділити мережу на дві частини (межа проходить між локальною мережею і мережею Інтернет) і сформувати набір правил, що визначають умови проходження пакетів з однієї частини в іншу. Окрім цього, міжмережевий екран також може виконувати й інші функції, пов'язані з фільтрацією трафіку від/до якого-небудь ресурсу мережі Інтернет. Принцип дії Firewall заснований на контролі надходжень трафіку ззовні.

Брандмауер може бути реалізований як апаратними засобами, так і програмними. Конкретна реалізація залежить від масштабу мережі, обсягу трафіку і необхідних завдань. Найбільш поширеним типом брандмауера є програмний мережевий і саме він буде використовуватися для вирішення проектної задачі. Мережевий Firewall встановлюється на шлюзі, що з'єднує кілька підмереж, і виконує фільтрацію потоку даних на мережевому рівні. Мережевий Firewall часто має функцію трансляції адресу (NAT), що дозволяє вести роботу підмереж від одного зовнішнього адреси шлюзу.

В даному проекті Firewall використовується для створення правил доступу віддаленого пристрою до пунктів призначення і доступам. Він представлений утилітою на ОС CentOS під ім'ям firewalld.

1.4 Постанова завдання

Метою роботи є розробка та програмна реалізація проекту захищеної корпоративної локальної мережі підприємства.

Для досягнення поставленої мети в роботі вирішуються наступні завдання:

– автоматизація процесу створення конфігураційних файлів;

– перекладання посадових обов'язків з високооплачуваних технічних фахівців на простих менеджерів;

– можливість давати клієнтам і співробітникам компанії віддалений доступ до сервісів компанії.

– розробка вимог до системи на основі вже існуючої локальної мережі з додаванням нового функціоналу:

– налаштування та приєднання до мережі OpenVPN серверу, який має змогу організувати маршрутизацію портів;

– створення клієнт-серверного додатку для обліку всіх співробітників компанії;

– створення скриптів генерації унікальних конфігураційних файлів для кожного клієнта;

– створити сполучне API для автоматичної відправки даних про користувача і його доступи на сервер для автоматичної генерації .apk файлу;

– додавання можливості скачування конфігураційних файлів і їх розсилки співробітникам.

Для досягнення поставленої мети, кваліфікаційна робота буде виконуватися в декілька етапів в наступній послідовності:

– формування вимог до системи;

– формування задачі;

– проектування інформаційної моделі до системи;

- розробка загальної архітектурі та структури проекту;
- розробка сертифікатів доступу;
- проектування та розробка алгоритмів допоміжного API;
- вибір інструментарію для реалізації ПЗ;
- розробка інтерфейсу та класів системи;
- розробка інструкції користувача та інструкції щодо впровадження.

Результатом виконання проектної дипломної роботи буде клієнт-серверний програмний додаток, що дозволяє вести облік співробітників, дозволених їм доступів до ресурсів офісу, та наявності конфігураційного файлу з їх особливими унікальними ідентифікаторами. Також додатком до проекту буде автоматизований скрипт на головному сервері, що генерує файл з налаштуваннями доступів враховуючи дані співробітника.

2 РОЗРОБКА ТЕХНІЧНОГО РОБОЧОГО ПРОЕКТУ ЗАХИСТУ КЛКМ

Для того щоб клієнт мав змогу отримати конфігураційний файл доступу до сервера, а також системний адміністратор міг відслідковувати який клієнт у який час заходив в локальну мережу, необхідно створити клієнт-серверний додаток для обліку доступів клієнтів.

Клієнт-серверний додаток буде являти собою платформу, в якій будуть зареєстровані всі працівники підприємства, а також список всіх доступів та відношення, який доступ дозволено клієнтові. Також буде інформація, чи має працівник конфігураційний файл взагалі.

2.1 Формування вимог до системи

Інформаційний обмін між користувачем і серверним додатком здійснюється за допомогою локальної мережі компанії.

Система повинна підтримувати декількох користувачів в онлайн режимі. Користувачі здійснюють доступ до системи через локальну мережу.

Інформаційні системи, побудовані на базі комп'ютерних мереж, забезпечують:

– зберігання даних;
– обробку даних;
– організацію доступу користувачів до даних;
– передачу результатів обробки даних користувачеві;
– використання супроводжуючих додатків і ресурсів мережі.

Завдання що виконуються системою

Для даної системи зручно користуватись схемою поділу даних програми, інтерфейсу і керуючої логіки на окремі компоненті з можливістю незалежної модифікації кожного компонента.

Структура запитів і відповідей серверної частини системи має дотримуватись єдиного стандарту.

Інтерфейс системи, документація і будь-яка текстова інформація в програми повинні бути англійською та українською мовами. Також, інтерфейс повинно бути виконано в єдиному графічному дизайні, з однаковим розташуванням основних елементів управління і навігації.

Система повинна забезпечувати виконання таких функцій:

- ідентифікація користувача;
- захист від несанкціонованого доступу до підсистем та даних;
- представлення інформації в зрозумілому для користувача вигляді;
- можливість зручного редагування даних;
- збереження старих даних;
- керування користувачами системи.

Організаційне забезпечення повинно бути достатнім для ефективного та швидкого виконання поставлених завдань при застосуванні функцій системи.

Захист від помилок користувача полягає в перевірці заповнення даних в деяких полях, можливості відновлення даних та зберігання резервних копій, а також розмежування доступу по функціям і повноваженням користувачів.

Вимоги до безпеки

Умови експлуатації, а також види і періодичність обслуговування технічних засобів системи повинні відповідати вимогам по експлуатації, технічного обслуговування, ремонту і зберігання.

Для захисту інформації від несанкціонованого доступу система повинна забезпечувати:

- ідентифікацію користувача;

– автентифікацію користувача;

– авторизацію (перевірку прав і обмежень доступу користувача на рівні функцій і масивів даних при роботі з системою).

Резервне копіювання

Моніторинг та відновлення системи при помилках в роботі апаратних засобів і помилки, пов’язаних з програмним забезпеченням (ОС і драйвери пристроїв), відновлення працездатності покладається на ОС.

Також використовується система транзакцій і у випадку невиконання будь-яких з команд відбуватися безпечний відкат - скасування всіх змін, внесених починаючи з моменту початку транзакції або з якоїсь точки збереження, і відображається помилка користувача в системі збирання помилок - Audit. Yii2 Audit - це модуль, який записує та відображає запити web/cli, зміни бази даних, помилки PHP/JS та пов'язані з ними дані.

Так само повинна бути забезпечена працездатність системи в цілому в разі виникнення збоїв, аварій або відмов окремих робочих станцій. Для захисту апаратури від перебоїв напруги і комутаційних перешкод повинні застосовуватися фільтри та джерела безперебійного живлення. Для збереження цілісності даних рекомендовано налаштувати процес резервного копіювання даних засобами програмного забезпечення та автоматизувати цей процес.

Доцільно передбачити модульну структуру проекту для можливості винести окремі сервіси на іншу машину і розташувати там, де найбільш зручно, наприклад в головному офісі компанії.

Види забезпечення

Система має архітектуру типу “клієнт-сервер” на основі web-технологій, тобто поділяється на дві частини. Основною частиною є серверна, що відповідає за обробку, збереження і передачу даних, та повертає відповідні дані та представлення інформації на запити клієнтської частини.

У комплекс технічних засобів для будування локальної мережі повинні входити такі елементи:

– VPS service;
– Операційна система CentOS;
– OpenVPN service;
– Python 3.6;
– Firewalld;
– встановлений OpenVPN клієнт на призначених для користувача пристроях;
– підтримка OpenVPN протоколу у маршрутизатора.

Мінімальні вимоги до технічного забезпечення серверної частини для зберігання додатку:

– оперативна пам’ять мінімум 2 ГБ;
– двух’ядерний процесор 2.2ГГц;
– жорсткий диск мінімум 30 ГБ місця;
– мережевий адаптер 1 Гбіт / с.

У комплекс програмного забезпечення серверної частини та встановлення клієнт-серверного додатку входять такі елементи:

– HTTP-сервер Apache 2.4 або nginx 1.16.1;
– СУБД MySQL 8.0.0 або вище;
– PHP 7.2;
– налаштування доступу до API.

Також необхідно забезпечити налаштування резервування конфігурації і ключів OpenVPN та резервування БД.

2.2 Формування задачі

Розробка повинна бути проведена в 4 стадії:

– розробка технічного завдання;
– проектування та налаштування локальної мережі;

– проектування клієнт-серверного додатку;

– реалізація та тестування.

На стадії розробки технічного завдання повинен бути виконаний етап розробки, узгодження і затвердження цього технічного завдання.

На стадії проектування та налаштування локальної мережі повинні бути виконані перераховані нижче етапи робіт:

– вибір структури і засобів побудови мережі;

– технічне моделювання мережі;

– настройка і підключення обладнання.

На стадії проектування клієнт-серверного додатку повинні бути виконані:

– вибір використовуваних технологій;

– визначення основних варіантів використання системи для користувачів;

– опис інформаційної моделі;

– проектування основних компонентів і алгоритмів системи у вигляді діаграм;

– проектування структури інтерфейсу;

– погодження та затвердження проектної документації.

На стадії реалізації та тестування повинна бути виконана робота по розробці інформаційної системи на основі технічного завдання, кодування та тестування з налагодженням.

2.3 Проектування інформаційної моделі

Функціональна модель

В інформаційну систему надходять нові дані, дані з БД, користувачі проходять автентифікацію та авторизацію, системи валідації. Потім

відбувається логування та маніпуляція з даними в системі, формуються документи на експорт. Усі дані вводить користувач.

Опис виглядає як “чорний ящик” з входами, виходами, управлінням і механізмом, який поступово деталізується до необхідного рівня.

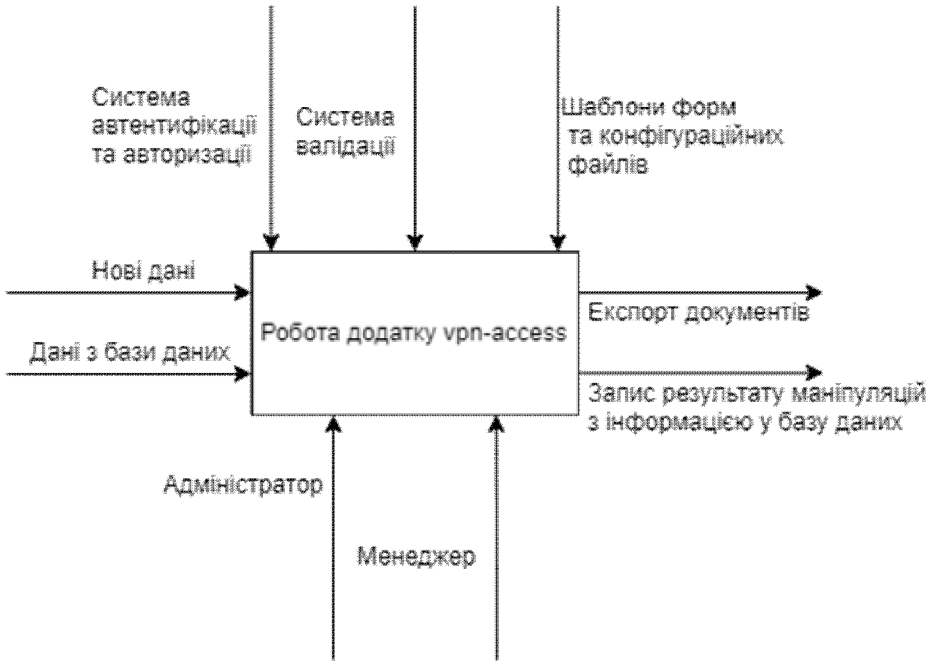

Рисунок 2.1 – Функціональна модель IDEF0

Розглянемо наступну модель IDEF0 представлену на рис. 2.1:

Вхідна інформація: введені дані, дані БД.

Керуюча інформація: система аутентифікації та авторизації, система валідації і шаблони форм та конфігураційних файлів.

Механізм управління: адміністратор, менеджер.

Вихідна інформація: результат дій над даними, формування файлів конфігурації.

Результат роботи ІС - це внесення даних до БД та відправлення запитів на формування конфігураційних файлів.

Модель представлення структури

На рис. 2.2 представлена структура таблиць бази даних і їх зв'язків.

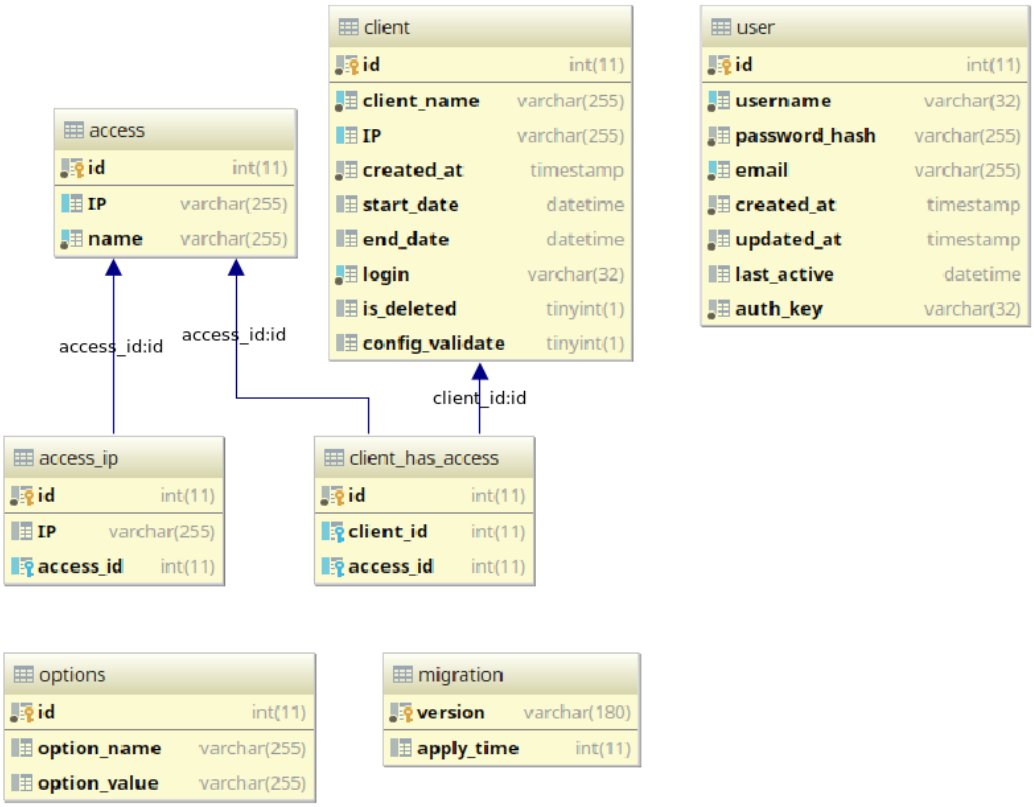

Рисунок 2.2 – Структура бази даних

У базі даних присутні такі таблиці:

– user – таблиця користувачів системи, наприклад адміністратора або менеджера. Користувачі потрібні для управління системою, створення клієнтів, сервісів компанії, дозволів і управлінням правами доступів клієнтів.

– client – таблиця обліку клієнтів, яким будуть видаватися дозволи на доступ до ресурсів компанії у вигляді конфігураційних файлів.

– access – таблиця обліку сервісів компанії до яких можна надавати доступ тим чи іншим користувачам.

– client_has_access – таблиця, яка створює зв'язок належності доступу до будь-яких сервісів компанії того чи іншого користувача.

– access_ip – таблиця, що зберігає ip-адреси для сервісів компанії, оскільки у одного сервісу може бути кілька ip-адрес.

– options – таблиця для створення необхідних статичних параметрів.

– migration – таблиця міграцій (списку змін бази даних), де відслідковуються всі виконані зміни бази даних.

2.4 Розробка загальної структури проекту

Моделювання мережі

Фізична структура мережі передбачає вибір архітектурно-планувальних і технологічних рішень. Прийнявши до уваги умови можливості віддаленого доступу для клієнтів і підключення інших офісів, зупинимося на виборі пристроїв, що входять до складу мережі.

Тип обладнання визначається завданнями, які вона має виконувати в змодельованій мережі. Сервер виконує роль: web-сервера і власник бази даних.

Для реалізації доступу клієнта до сервісів компанії, на прикладі одного клієнта, необхідна наявність такого обладнання:

– клієнтський пристрій (комп'ютер, телефон);
– роутер клієнта;
– VPS сервер з налаштованим на ньому OpenVPN сервером;
– офісний роутер;
– сервер з сервісами компанії.

Як показано на рис. 2.3, VPS сервер займається маршрутизацією до ресурсів. У встановленому на ньому OpenVPN сервері повинні бути заздалегідь налаштовані маршрути до офісів, де встановлені сервіси до яких треба отримувати доступ ззовні.

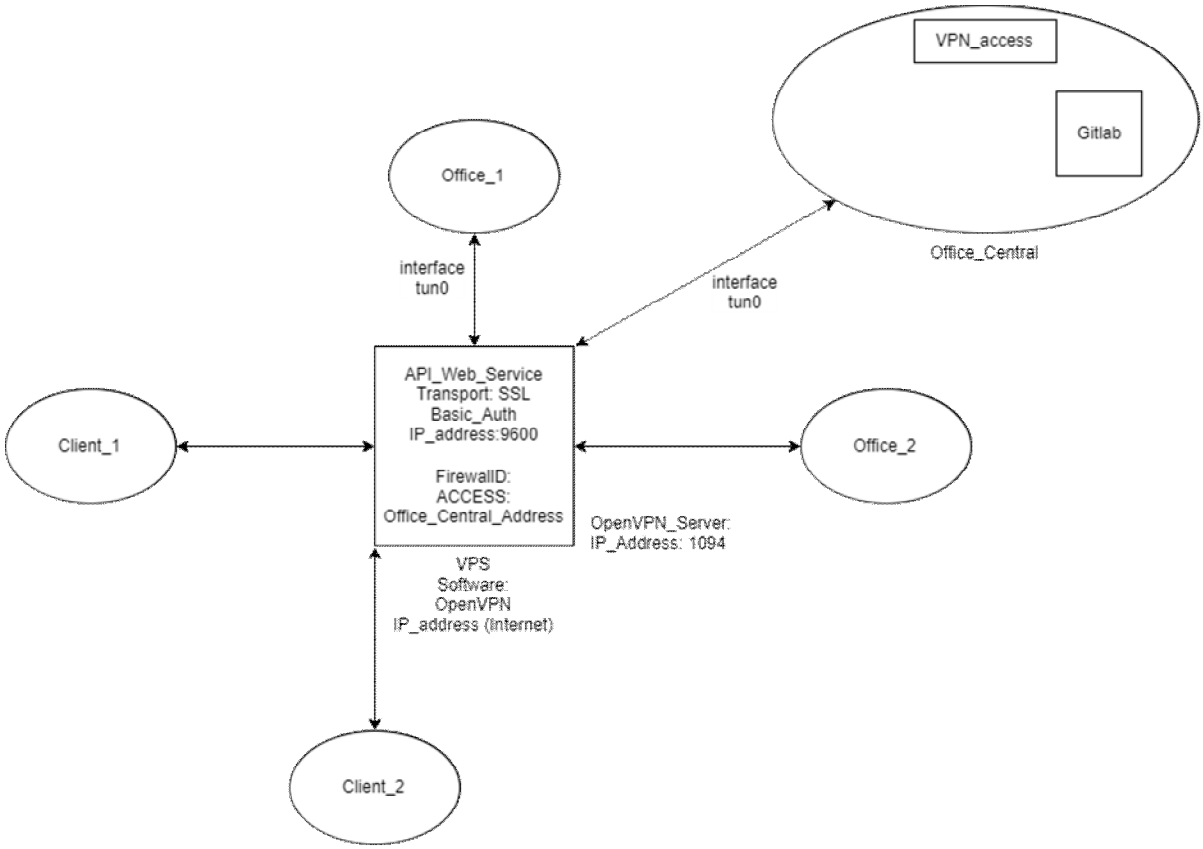

Рисунок 2.3 – Загальна схема роботи системи

Сам додаток обліку клієнтів, іменоване як vpn-access, в даній реалізації знаходиться безпосередньо на офісному сервері, разом з сервісами компанії.

При підключенні ще одного офісу до локальної мережі Office_1 підключає сертифікат в своєму маршрутизаторі і йому стає відомі необхідні маршрути доступу до сервісів. При цьому обов'язково потрібно врахувати, що сітки в офісах повинні бути різними, інакше Office_1 буде думати що сервіси знаходиться в його мережі і не відправлятиме запит.

Так само відбувається і з підключенням до сервісів компанії клієнта.

Моделювання додатку

Для спрощення створення додатку для управління клієнтами та їх даними, була обрана MVC структура проекту. Вбудована в платформу функція генерації цього шаблону не тільки спростить, а й прискорить і автоматизує процес розробки і створення додатка, назвемо його, наприклад, VPN-access.

Шаблон MVC описує простий спосіб побудови структури додатка, метою якого є відділення бізнес-логіки від призначеного для користувача інтерфейсу. В результаті, додаток легше масштабується, тестується, супроводжується і звичайно ж реалізується.

Шаблон проектування MVC передбачає поділ даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: Модель, Представлення і Контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

В архітектурі MVC модель надає дані і правила бізнес-логіки, уявлення відповідає за користувальницький інтерфейс, а контролер забезпечує взаємодію між моделлю і представленням.

Розглянемо концептуальну схему шаблону MVC (див. рис. 2.4):

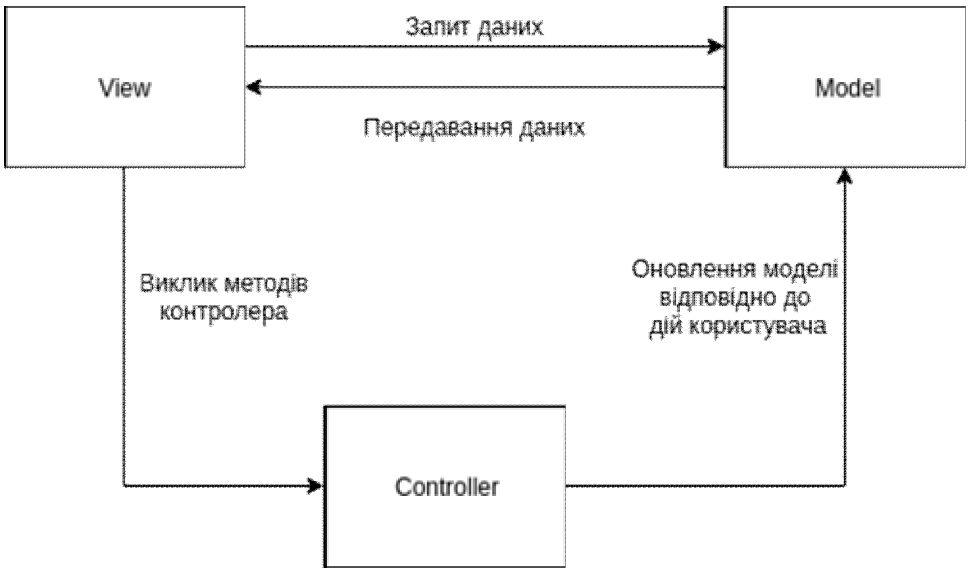

Рисунок 2.4 – Концептуальна схема шаблону MVC

Типову послідовність роботи MVC-додатку можна описати таким чином:

– при заході користувача на веб-ресурс, скрипт ініціалізації створює екземпляр додатку і запускає його на виконання;

– при цьому відображається вид, наприклад, головної сторінки сайту;

– додаток отримує запит від користувача і визначає запитані контролер і дію. У разі головної сторінки, виконується дія за замовчуванням (index);

– додаток створює екземпляр контролера і запускає метод дії, в якому, наприклад, міститися виклики моделі, що зчитують інформацію з бази даних;

– після цього, дія формує уявлення з даними, отриманими з моделі і виводить результат користувачеві.

Для реалізації завдання знадобляться три основні сутності: Користувач, Клієнт і Доступ (або ж Сервіс компанії). На рис. 2.5 представлена схема їх взаємодії.

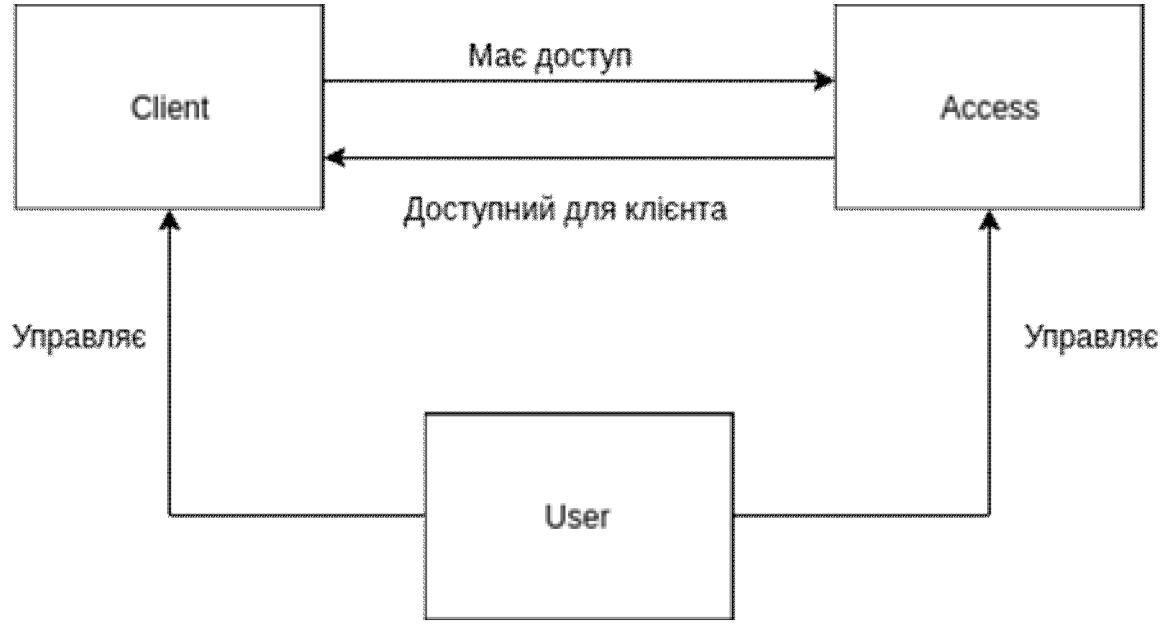

Рисунок 2.5 – Взаємодія сутностей додатку

Модель користувача є керуючою сутністю системи. Вона буде зберігати інформацію про пошту, логін та пароль користувача, а так само дати створення, зміни та останньої активності в системі.

Сутність клієнта зберігає в собі інформацію про імені, логін та пароль клієнта, дату його створення і дати початку і закінчення роботи його конфігураційного файлу.

Сутність сервісу компанії містить інформацію про назву і основного IP адреси цього доступу.

Забезпечивши взаємозв'язок цих сутностей, можна отримати повноцінну систему, яка зберігає інформацію не тільки про користувачів, але так само клієнтів і сервісах компанії, їх взаємозв'язку і належності один одному.

Конфігураційні файли кожного клієнта зберігаються безпосередньо в проекті для полегшення швидкого доступу до них і не є сутністю проекту.

2.5 Розробка сертифікатів доступу

Оскільки була обрана технологія OpenVPN, то необхідно знати, що для користувача даною технологією надається пару видів аутентифікації:

Перший, за допомогою заздалегідь визначеного ключа, це по суті найбільш простий спосіб.

Другий, це сертифікатна аутентифікація, яка є дуже гнучкою в процесі настройки.

І третій: за допомогою логіна і пароля (може працювати без створення клієнтського сертифіката, але серверний сертифікат все одно потрібен).

Налаштування OpenVPN в роботі буде відбуватися так, що програма буде використовувати сертифікати. Це дуже надійний спосіб як зашифрувати переданий трафік, так і захистити підключення до сервера.

Сертифікати – це публічні ключі, завірені підтверджуючий центр сертифікатної автентифікації (CA). Вони використовуються для зашифровування даних. Факт завірення сертифікату підтверджуючим центром CA дозволяє ідентифікувати сторону, яка транслює зашифровані дані. Файл запиту на сертифікат створюється на вузлах мережі, потім він переноситься на вузол, що засвідчує, і там підписується. Створений в результаті підписаний сертифікат переноситься назад на запитав його вузол мережі OpenVPN.

Після створення всіх сертифікатів, потрібно налаштувати безпосередньо сервер і клієнт. Ця установка полягає в тому, що згенеровані файли ключів потрібно скопіювати в правильне розташування на сервері і клієнтів (у кожного з клієнтів по своїй парі публічний-приватний ключ) і вказати шлях до файлів з ключами, вказати IP підключення і деякі інші опції.

Маючи вже налаштований сервер OpenVPN з'являється можливість вирішувати підключення для нових користувачів. Для цього треба випустити новий клієнтський приватний ключ і іменний конфігураційний файл.

Якщо клієнтів буде більше одного, то для кожного необхідно створити відповідну кількість сертифікатів з відповідним ім'ям.

При створенні конфігураційного файлу через додаток і відправці користувачеві ovpn файлу, відбуваються такі дії:

– створення OVPN конфігурації на сервері через api_python_script, який створює відкритий і закритий ключ, підпис CA сертифікатом (див додаток А);

– для клієнта створюється ccd конфігурація, в якій описуються маршрути;

– створюються firewall правила, за допомогою утиліти firewall-cmd на OS: CentOS 7.

Схема отримання сертифіката на рис. 2.6.

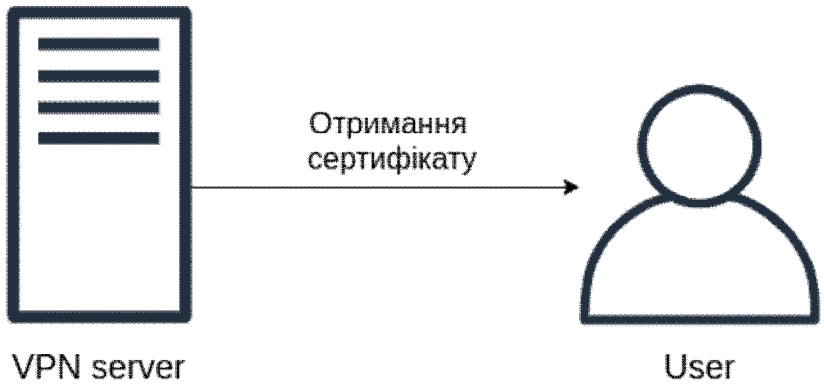

Рисунок 2.6 – Отримання сертифікату

Підключення сертифіката через утиліту OpenVPN-Client і авторизація / з'єднання з сервером (див. рис. 2.7):

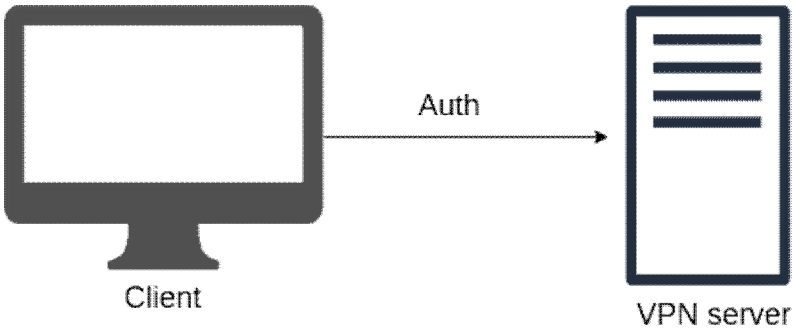

Рисунок 2.7 – З'єднання з сервером

Перевірка ключів і перевірка наявності конфігураційного файлу в директорії ccd (див. рис. 2.8) :

а) перевірка наявності конфігурації ccd;

б) надання маршрутів ip-network.

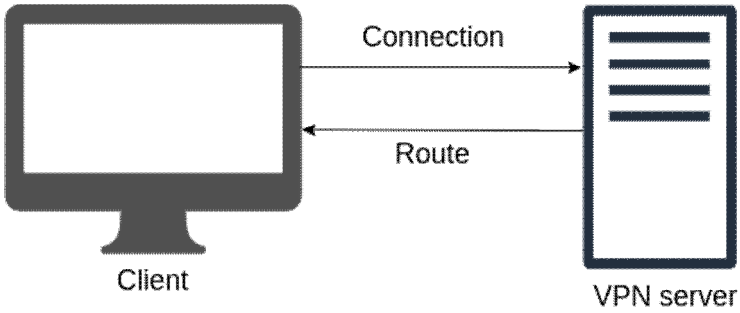

Рисунок 2.8 – Перевірка ключів та наявності файлів

Клієнт отримавши маршрути, додає до себе в таблицю маршрутизації.

Тим самим йому стає відомо за яким маршрутом клієнт може дістатися до вузла.

Загальну схему взаємодії показано на рис. 2.9:

– створення конфігураційного файлу;

– отримання конфігураційного файлу;

– отримання конфігураційного файлу клієнтом;

– з'єднання і отримання маршрутів, отримання IP-address який буде використовуватися в тунелі;

– підключення до Gitlab.

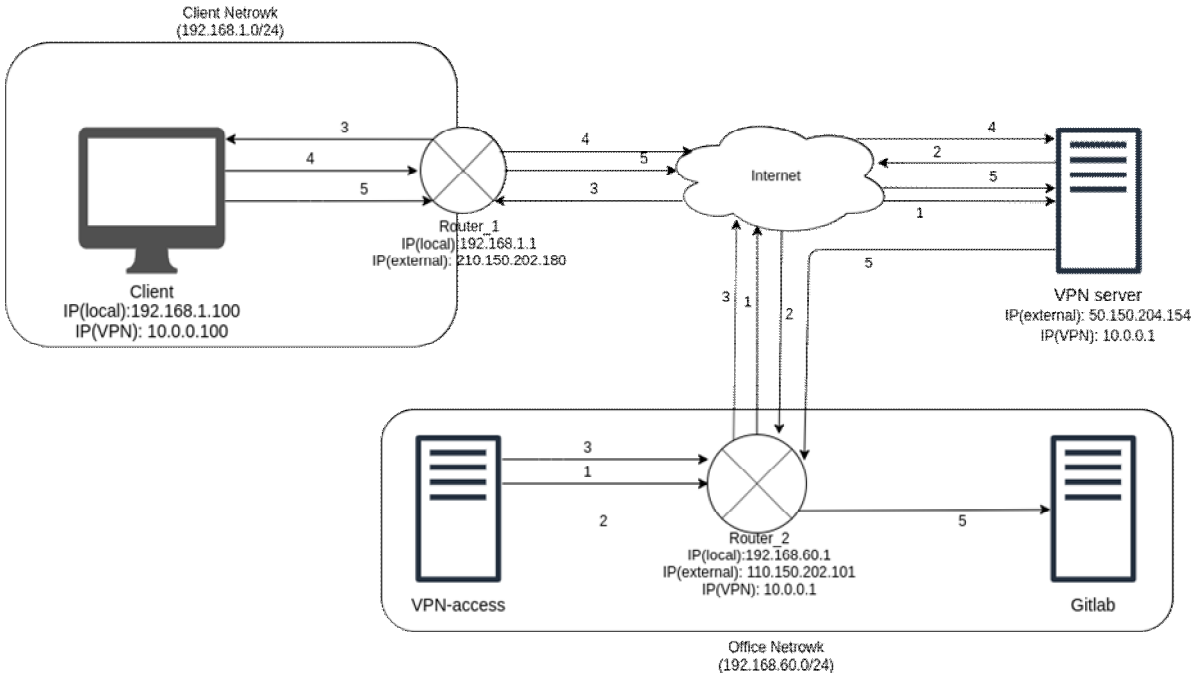

Рисунок 2.9 – Загальна схема роботи мережі

2.6 Проектування допоміжного API

Для взаємодії та назалежної роботи клієнтського та серверного додатків необхідний програмний інтерфейс додатку (API). У випадку web-додатків існує архітектурний стиль для розподілених гіпертекстових систем - REST (Representational State Transfer). В основі REST закладено принципи функціонування всесвітньої павутини і можливості HTTP.

REST, як і кожен архітектурний стиль відповідає ряду архітектурних стилів.

Перша архітектура від якої він успадковує обмеження- це клієнт-серверна архітектура. Її обмеження вимагає розділення відповідальності між

компонентами, які займаються зберіганням та оновленням даних (сервером), і тими компонентами, які займаються відображенням даних на інтерфейсі користувача та реагування на дії з цим інтерфейсом (клієнтом). Таке розділення дозволяє компонентам еволюціонувати незалежно.

Наступним обмеженням є те, що взаємодії між сервером та клієнтом не мають стану, тобто кожен запит містить всю необхідну інформацію для його обробки, і не покладається на те, що сервер знає щось з попереднього запиту.

Додатковим обмеженням стилю REST є те, що системи, написані в цьому стилі, повинні підтримувати кешування, тобто дані, які передаються між сервером повинні містити інформацію про те, чи можна їх кешувати, і якщо можна, то як довго. Це дозволяє збільшувати продуктивність уникаючи зайвих запитів, але також зменшує надійність системи, через те, що дані в кеші можуть бути застарілі.

Ключове поняття в REST - це ресурс. Ресурс має стан, і його можна отримувати або змінювати за допомогою представлень. Додаток відповідає за безліч таких ресурсів. До речі, сукупність станів ресурсів — це і є стан додатку.

Під представленням можна розуміти JSON / HTML / XML / текст в певному форматі або що завгодно, що дозволяє нам розуміти стан ресурсу або його модифікувати. Представлення, яке модифікує стан ресурсу, і представлення, яке щось говорить нам про стан, не обов'язково повинні відповідати один одному.

Не існує якогось єдиного стандарту, який би повністю описував REST. Але існує безліч маленьких стандартів, які доповнюють даний стиль з боку:

– URI і URI Template – про однакові ідентифікатори ресурсів;

– HTTP – як протокол передачі «гіпертексту»;

– JSON, XML можуть використовуватися для представлення;

– HTML можна використовувати не тільки для представлення, а й черпати з нього ідеї для гіпермедіа компоненту.

Через відсутність загального стандарту, між розробниками виникла проблема в домовленості реалізації REST API. І тому розробник чи група розробників почали створювати свої стандарти (домовленості) до REST. Так одним з найпопулярніших сучасних "стандартів" є JSON API.

REST та JSON API – це контракт (договір) між клієнтом і сервером про те як взаємодіяти один з одним.

В JSON API все сконцентровано на форматі JSON - всі ресурси та статуси знаходяться в представлені. А всі операції отримання та відправки даних реалізовані через HTTP метод POST.

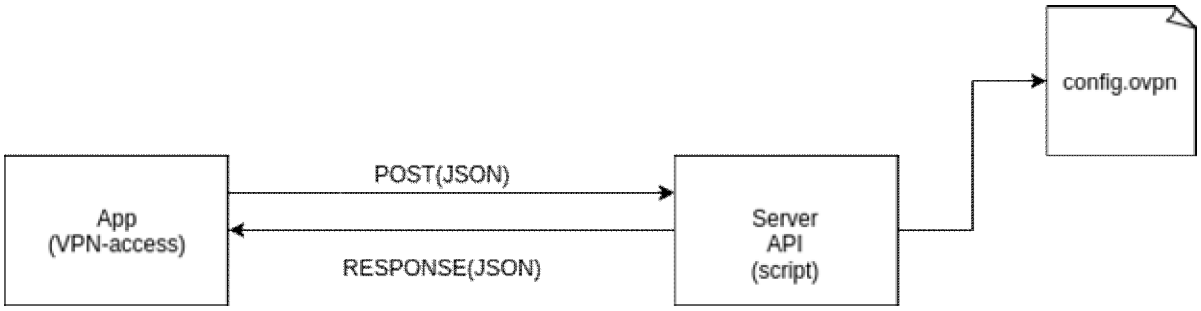

Рисунок 2.9 – Загальна схема взаємодії клієнтської частини із серверною

Тому загальна схема взаємодії клієнтської частини із серверною за допомогою JSON API виглядає таким чином (див. рис 2.9) :

– клієнтська частина встановлює зв’язок із сервером та направляє запит у JSON форматі;

– сервер приймає запит та відповідно до отриманого методу виконує свої функції над конфігураційним файлом;

– сервер повертає результат запиту клієнтській частині також у JSON форматі.

Результат запиту повертає вмісту конфігураційного файлу у вигляді рядка, оскільки зчитувати байти та збирати дані з різних файлів і віддавати у вигляді рядка даних набагато простіше і швидше, ніж зібрати файл, зберегти, а потім цей файл відправити. Тому отриманий рядок зберігається в новостворений файл в самому проекті.

Після цього менеджер може побачити наявність конфігураційного файлу у клієнта і має можливість його скачати для того щоб в подальшому передати клієнту для користування дозволеними того ресурсами компанії.

Сам конфігураційний файл зберігається безпосередньо в папці проекту, так як постійно звертатись на сервер для отримання вже існуючого файлу - більш складний шлях, в якому немає необхідності, адже можуть виникнути проблеми із доступом до серверної частини API.

Так само, над файлом можна виконувати такі дії:

– get – отримати вже існуючий файл з сервера;

– create – створити новий файл на сервері;

– revoke – відкликати конфігураційний файл;

– upd_act – для певного облікового запису перепризначити доступи;

– upd_acs – для декількох користувачів змінити ір певного доступу на новий.

При кожному запиті на сервер необхідно проводити авторизацію.

3 РОЗРОБКА СИСТЕМИ

3.1 Вибір інструментарію

Розробка системи велася на мови програмування PHP в середовищі розробки JetBrains PhpStorm.

За основу при розробці back-end інформаційної системи була обрана платформа Yii2 із застосуванням узгодження до REST – JSON API, яка написана також за допомогою мови PHP.

Ґрунтується ця платформа на базових поняттях MVC - Model View Controller. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами. В моделі відбувається вся головна логіка системи, також можна сказати, що модель - проекція однойменної таблиці з БД. Контролер реагує на вхідні дані та віддає результат запиту. Вигляд відповідає за комфорт введення нових даних і можливість управління функціоналом системи користувачем.

Також чинниками вибору цієї платформи стали можливості Yii2 до генерації базового PHP-коду для CRUD-операцій, міграції бази даних і підтримка REST.

Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм. Важливим критерієм на користь вибору даного мови програмування для розробки є досвід розробки на ньому, отриманий за час навчання.

У якості реалізації інформаційної моделі було використано базу даних MySQL.

3.2 Розробка системи та інтерфейсу

Оскільки проект побудований на принципі МВЦ архітектури, то в структурі коду програми присутні як моделі, так і контролери та подання.

Моделі відповідають за сутності і за їх методи. Контролери пов'язують роботу моделей з уявленнями.

На малюнку 3.1 представлений список використовуваних моделей та наданий список використовуваних контролерів.

Кожній моделі відповідає таблиця в базі даних MySQL. Такі таблиці як, наприклад, ClientHasAccess, є проміжною або ж сполучною таблицею між основними моделями. Тобто виявляється зв’язком багато до багатьох, оскільки у одного клієнта може бути кілька доступів, так само, як і у одного доступу може бути кілька клієнтів.

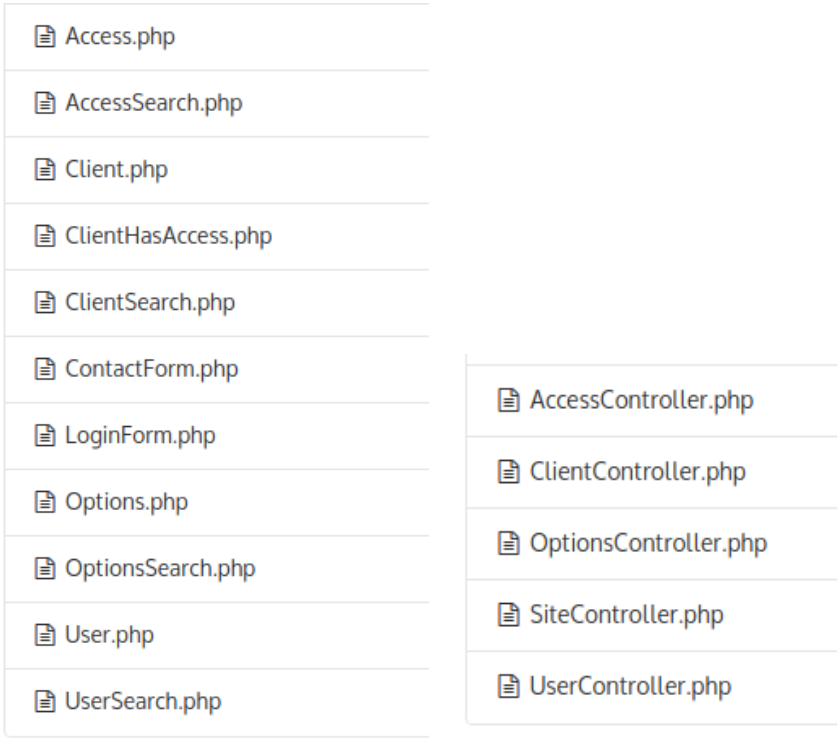

Рисунок 3.1 – Список використовуваних моделей і контролерів

Більш детальний розгляд всіх моделей з їх полями і функціями представлено на рис. 3.2.

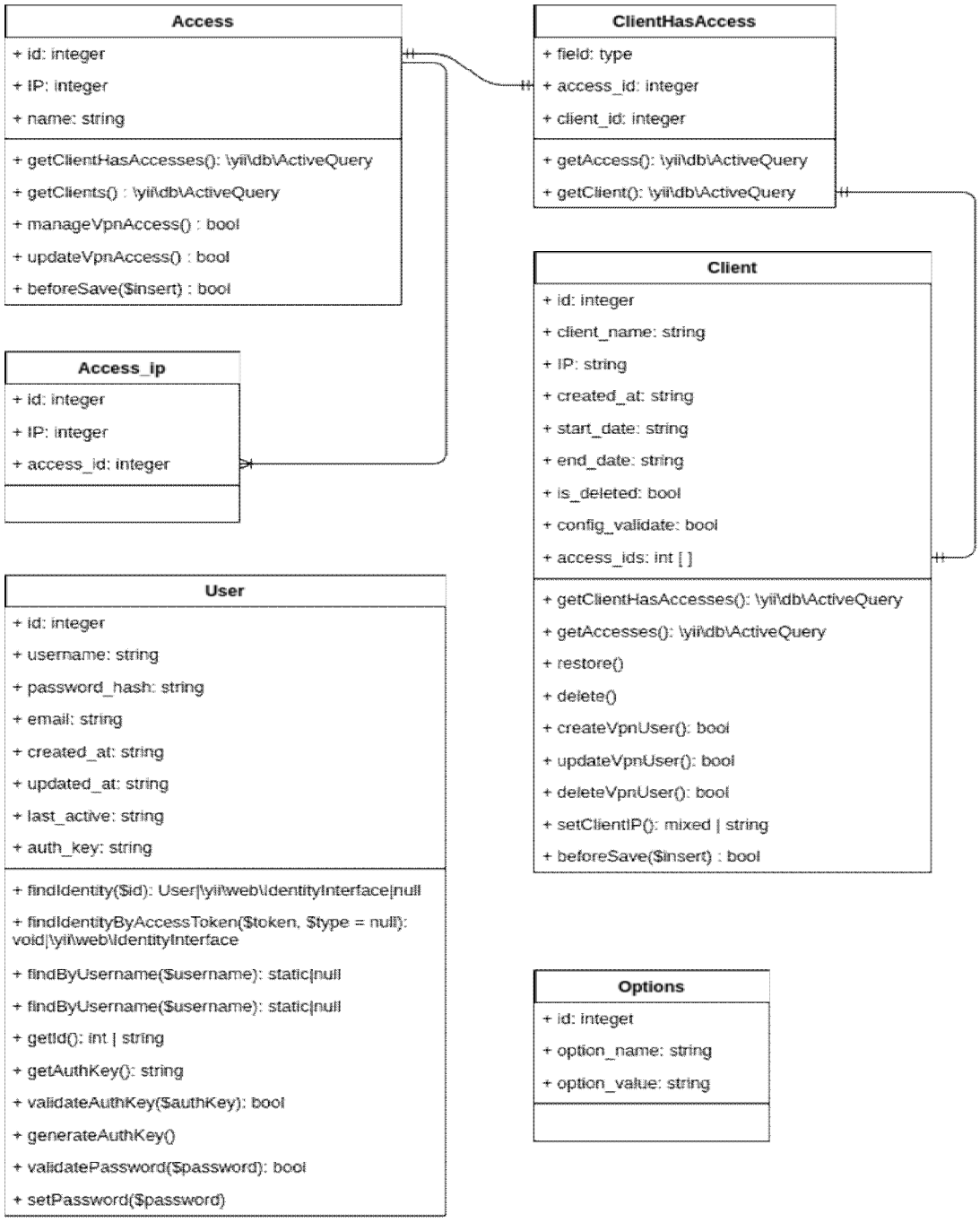

Рисунок 3.2 – Детальна діаграма класів системи

Модель користувачів і модель клієнтів відрізняється між собою. Модель користувачів – це уявлення адміністраторів або ж модераторів, які управляють всіма клієнтами (створюють, видаляють, додають особисту інформацію) і дають той чи інший доступ до того чи іншого клієнта. Модель клієнтів являє собою співробітника, якому доступні один або більше доступів.

Авторизуватися в системі мають право тільки користувачі цієї системи, клієнти не мають права на авторизацію.

Приклад коду авторизації користувача:

```
 public function actionLogin()
    {
        if (!Yii::$app->user->isGuest) {
            return $this->redirect('/client/index');
        }

        $model = new LoginForm();
        if   ($model->load(Yii::$app->request->post())   &&   $model-
>login()) {
            return $this->goBack();
        }

        $model->password = '';
        return $this->render('login', [
            'model' => $model,
        ]);
    }
```

Як тільки користувач створює клієнта, відразу ж для цього клієнта створюється конфігураційний файл з усіма параметрами. Для цього по API йде запит на сервер і повертається файл з налаштуваннями. В будь-який зручний час його можна завантажити з профілю клієнта. Авторизація на сервері через додаток за допомогою API відбувається за допомогою ось такої функції:

```
private function createVpnUser()
    {
        try{
            $manager = new VpnManager();
            $manager->setUsername($this->client_name);
            $manager->setAction(VpnManager::ACTION_CREATE);
            $manager->setExpiry($this->end_date);
            $manager->setAccess($this->access_ids);
            $manager->send();
            return true;
```

```
            }catch (\Exception $exception){
                Yii::$app->session->setFlash('Error',        'Connection
failed to vpn service');
                return false;
            }
        }
```

В даному випадку VpnManager (див. додаток Б) – це окремий клас який і являє собою самє API для цього додатка, що містить необхідні функції для відправки запитів на сервер.

Для зручності роботи був так само створений CurlManager (див. додаток В), необхідний для відправки запиту на сервер. Він заснований на кроссплатформенній службовій программі командного рядка для передачі файлів з використанням різних протоколів з синтаксисом URL. У PHP включена підтримка libcurl, яка підтримує протоколи http, https, ftp, gopher, telnet, dict, file і ldap. libcurl також вміє працювати з сертифікатами HTTPS, посилати запити до HTTP-серверів методами POST і PUT, завантажувати файли по протоколах HTTP і FTP, використовувати проксі-сервери, cookies та аутентифікацію користувачів.

У моделі клієнта в методі beforeSave() знаходиться функціонал, який відповідає за розпізнавання дій над моделлю, наприклад створення, оновлення, видалення, і, відповідно до дії, використовуючи VpnManager та CurlManager відправляє запит на сервер для виконання дій з файлами.

Незважаючи на те, що конфігураційний файл створюється разом з самим клієнтом, користувач системи може змінювати його або видаляти, додавати клієнтові особисту інформацію, а також створювати доступи і додавати клієнтів до цих доступів. Навіть якщо у клієнта є доступ до певної адреси в файлі конфігурації, а користувач системи вирішив заборонити цей доступ, то незважаючи на наявність налаштувань з дозволами у клієнта, доступу до цього ресурсу у нього вже не буде.

Потім в представленні моделі можна буде побачити наявність даного конфігураційного файлу.

3.3 Розробка інструкції користувача

Оскільки це простий додаток для менеджера і буде використовуватися тільки у компанії, було прийнято рішення дизайн інтерфейсу зробити стандартними функціями платформи Yii2.

Інтерфейс підтримує як англійську, так і українську та російську мови.

На початковій сторінці (див. рис. 3.3) можна побачити форму входу в систему, яка запитує логін і пароль користувача. У цій формі так само існує пагінація і перевірка.

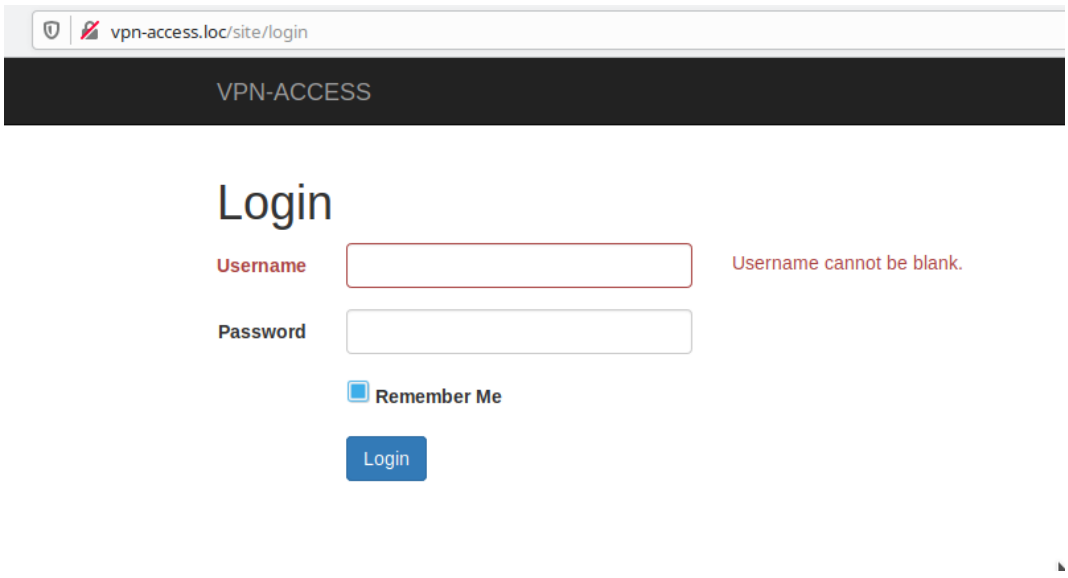

Рисунок 3.3 – Сторінка авторизації

Зручне горизонтальне меню дозволяє переходити між сторінками користувачів, клієнтів і доступних сервісів. Так само є окрема панель користувача, де він може налаштувати необхідні параметри, наприклад додати опції, або ж вийти з системи.

На головній сторінці для зручності менеджера і швидкості пошуку відразу зроблена сторінка зі списком клієнтів системи (див. рис. 3.4). Тут можна переглядати як активних, так і віддалених або заблокованих клієнтів. Для кожного клієнта в списку є можливість перегляду і редагування профілю, завантаження конфігураційного файлу і видалення профілю. Так само є кнопка для швидкого створення нового клієнта.

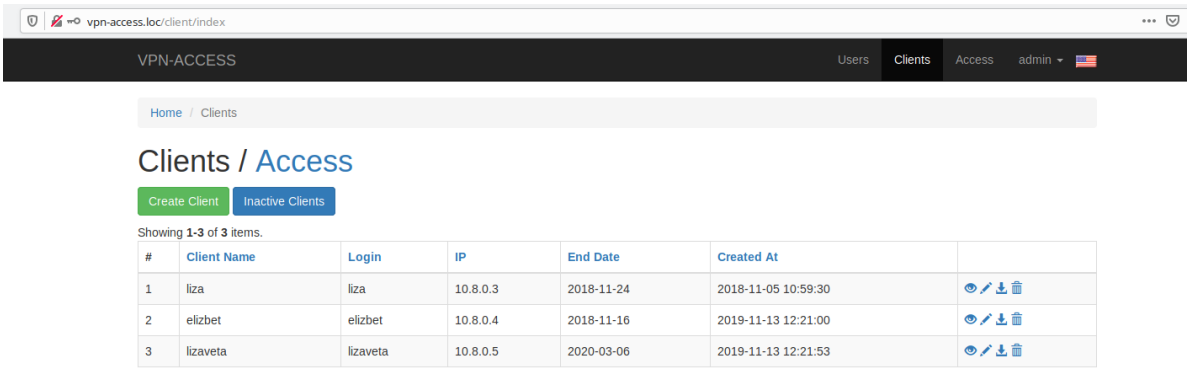

Рисунок 3.4 – Головна сторінка

Форма створення клієнта (див. рис. 3.3.3) передбачає такі поля:

– Client Name – реальне ім'я клієнта;
– Login name – користувача для входу в систему;
– End Date – дата закінчення терміну сертифіката;
– Accesses – список сервісів, до яких клієнтові дозволений доступ.

Рисунок 3.5 – Форма створення клієнта

При перегляді профілю клієнта (див. рис. 3.6) крім загальної інформації про користувача, можна побачити ще й список сервісів і їх IP адрес, до яких йому дозволено доступ.

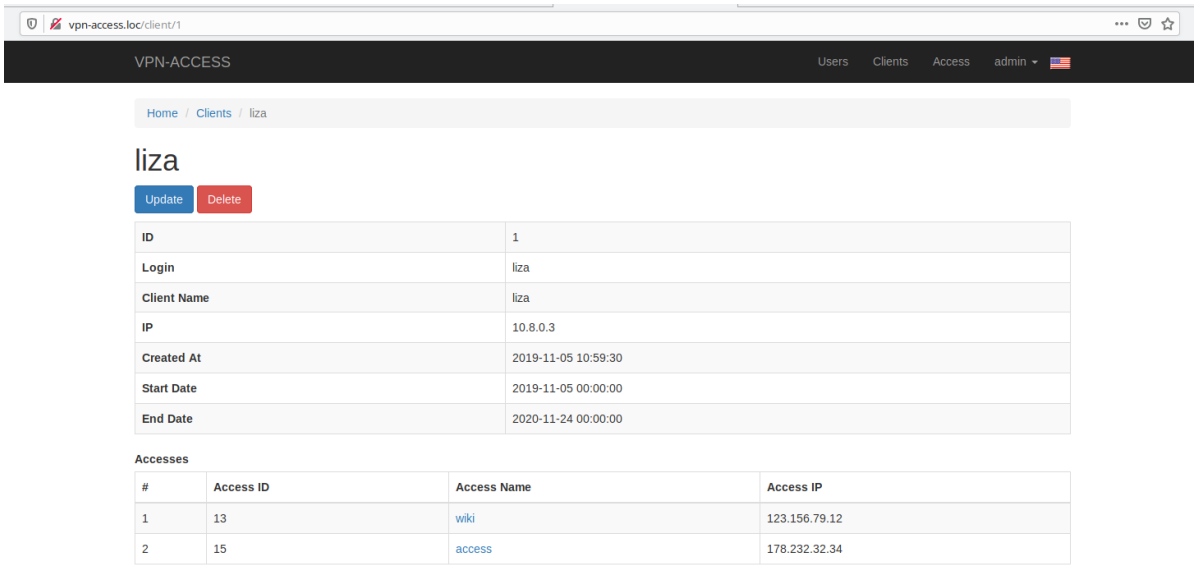

Рисунок 3.6 – Профіль клієнта

Сторінка зі списком доступів так само має інформацію про назву сервісу і його IP адреси. Їх так само можна переглядати і редагувати (див. рис. 3.3.5).

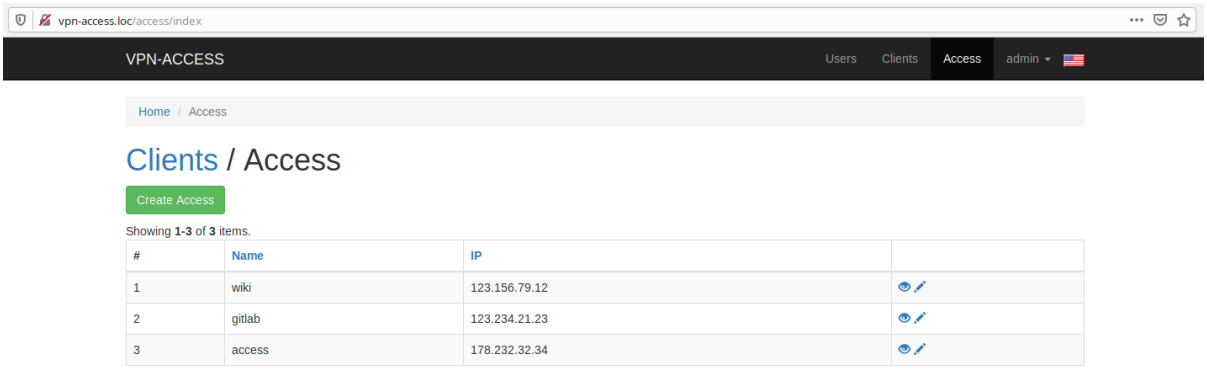

Рисунок 3.7 – Список сервісів компанії

При перегляді загальної інформації про сервіс (див. рис. 3.8) так само видно яким клієнтам дозволений доступ до нього і до якого числа.

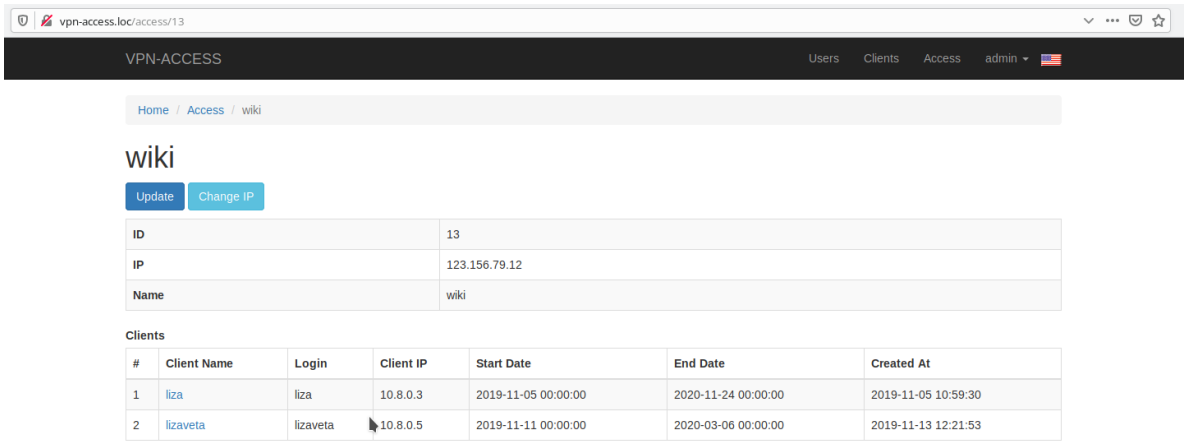

Рисунок 3.8 – Перегляд інформації сервісу

При редагуванні можна додавати або видаляти цей доступ клієнтам або міняти йому назву (див. рис. 3.9).

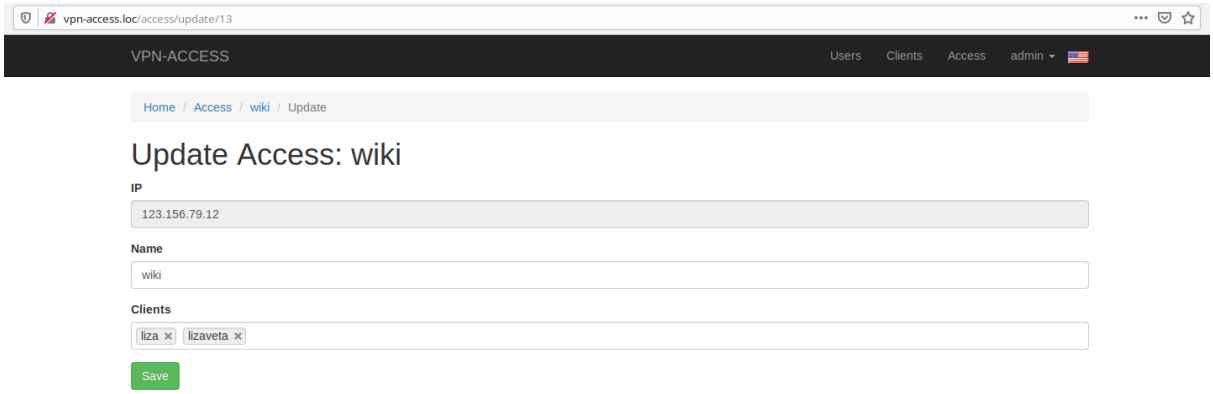

Рисунок 3.9 – Редагування інформації сервісу

У сервісу може бути кілька IP адрес, їх додавання, зміна та видалення винесені в окрему форму для зручності менеджера.

І оскільки ця система передбачена для користування декількома офісами, може бути і кілька адміністраторів або менеджерів, які контролюють додавання клієнтів і сервісів. На сторінці користувачів (див. рис. 3.10) можна побачити їх список з можливостями перегляду, редагування і видалення.

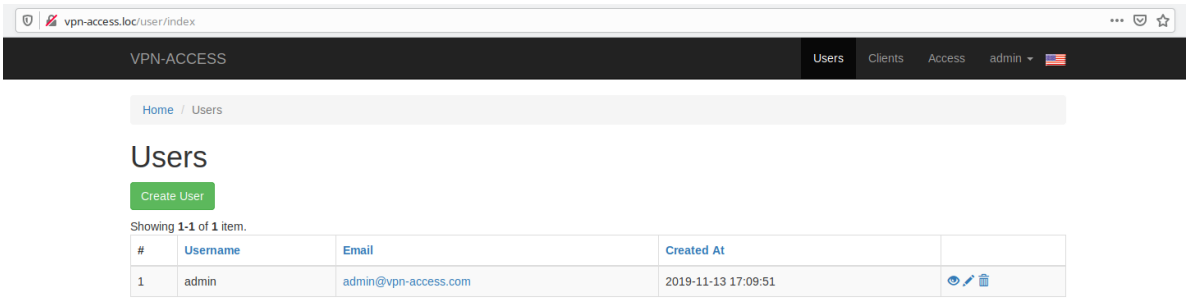

Рисунок 3.10 – Список користувачів системи

Інтерфейс системи зроблений простими стандартними засобами стилізації платформи і інтуїтивно зрозумілий для користувача.

У випадку виникнення помилок на сервері або відсутності з'єднання з ним, користувач побачить спливаюче повідомлення про неможливість виконувати будь-які дії пов'язані з конфігураційними файлами (крім скачування вже існуючих).

3.4 Розробка інструкції по впровадженню

Готовий додаток, повинен бути завантажений на github.com в якості OpenSource-проекту, доступному для завантаження будь-кому. Протягом всього виконання завдання до проекту було підключено систему контролю версій Git. Завдяки цій системі, можна відстежити і подивитися зміни проекту в ході роботи над ним. Спочатку для реалізації завдання, було створено репозиторій із назвою дипломного проекту та за допомогою наступних команд, було завантажено модуль на github.com.

Після виконання команд, спочатку було завантажено пустий проект, скопійовано всі файли та залежності до завантаженої папки та відправлено до сайту github.com, де все розміщено у якості репозиторія. Тепер, проект доступний для завантаження будь-кому.

Модуль успішно викладено на github.com у якості репозиторія, доступного для завантаження будь кому (див. рис. 3.11).

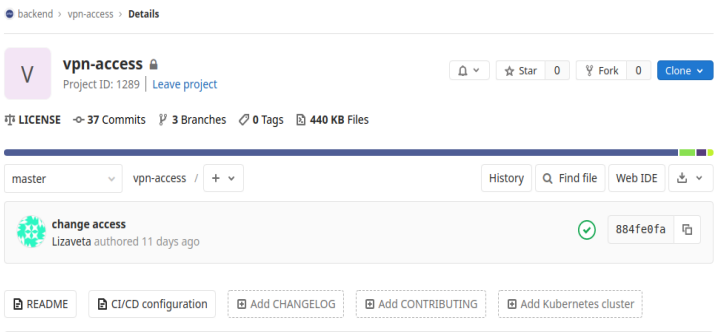

Рисунок 3.11 – Репозиторій на github.com

Після завантаження, будь який розробник може завантажити репозиторій, для безкоштовного використання.

Для установки проекту всі інструкції описані у файлі README.md. У ньому знаходиться структура папок проекту, настройки конфігурації бази даних, команди виконання міграцій і інструкція по створенню першого користувача - адміністратора. Описується які файли налаштувань потрібні для підключення до віддаленого сервера API.

ВИСНОВКИ

В результаті виконання роботи була розроблена програмна система, що включає в себе клієнт-серверний проект захищеної корпоративної локальної мережі.

Клієнт-серверний додаток зберігає інформацію про кожного користувача цієї системи та всіх наявних мережевих доступах, та створює користувачу дозвіл до сервісів компанії у вигляді конфігураційного файлу.

Спроектована структура захищеної корпоративної локальної мережі.

Виконаний аналіз існуючих на даний момент VPN сервісів показав особливості сучасних методів доступу до локальної мережі ззовні та їх порівняльну характеристику.

Використання OpenVPN сервера зробило локальну мережу підприємства більш захищеною і дає можливість клієнтам користуватися ресурсами локальної мережі ззовні.

Впровадження розробленого додатка дозволить спростити облік користувачів яким було дано доступ, а також автоматизувати роботу створення конфігураційних файлів з настройками доступу.

Розміщено на сайті GitHub, в якості Open Source проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы. Санкт-Пітербург : Питер, 2001. 672 с.

2. Anna Moretti, Network Governance, The Network Organization, 10.1007/978-3-319-52093-3_3, (51-85), (2017).

3. Wikipedia. Wikimedia Foundation. URL: http://ru.wikipedia.org/ (дата звернення: 21.09.2019)

4. The Definitive Guide to Yii 2. Wikimedia Foundation. URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата звернення: 24.09.2019)

5. Ubuntu Documentation. Ubuntu Documentation Team. URL: https://help.ubuntu.com/ (дата звернення: 24.09.2019)

6. The PHP Manual. The PHP Group. URL: https://php.net/ (дата звернення: 24.09.2019)

7. OpenVPN. OpenVPN Security Advisory. URL: https://openvpn.net/vpn-server-resources/ (дата звернення: 05.10.2019)

8. GitLab Documentation. GitLab Group. URL: https://about.gitlab.com/ (дата звернення: 05.10.2019)

9. MVC для веб: проще некуда. Vnaz. URL: https://habr.com/ru/post/181772/ (дата звернення: 05.10.2019)

10. Реалізація MVC патерну на прикладі створення сайту-візитки на PHP технологии. URL: https://habr.com/ru/post/150267/ (дата звернення: 10.10.2019)

11. Олищук А. В., Чаплыгин А. Н. Разработка WEB приложений на PHP 5. Профессиональная работа. Санкт-Пітербург : Вильямс. 2006. 352 с.

12. PHP, MySQL и другие веб-технологии. URL: http://www.php.su/ (дата звернення: 11.10.2019)

13. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. Москва : Питер, 2011. 384 с.

14. Бэрон Ш. MySQL. Оптимизация производительности. Москва : Символ-Плюс, 2010. 564 с.

15. Линн Б. Изучаем PHP и MySQL. Москва : Ексмо, 2010. 358 с.

16. Ренди Д.Я. MySQL. MySQL и mSQL. Базы данных для небольших предприятий и Интернета. Москва : Символ-Плюс, 2000. 664 с.

17. Маклафлин Б. PHP, MySQL. Исчерпывающее руководство. Москва : Питер, 2013. 512 с.

18. Сафронов М. Web Application Development with Yii 2 and PHP. Москва : Пакт, 2014. 377 с.

19. Б.Кэкк. Yii 2 For Beginners. Москва : Пакт, 2014. 290 с.

20. Камер Д., Стивенс Д. Сети TCP/IP : Т. 3. Разработка приложений типа клиент/сервер для Linux/POSIX. Санкт-Пітербург : Вильямс, 2002.

21. Кулаков Ю. О., Луцький Г. М. Комп'ютерні мережі. Київ : Юніор, 2003. 395 с.

22. Олифер В. Г., Олифер Н. А. Основы сетей передачи данных. Интернет-университет информационных технологий. ИНТУИТ. ру, 2005.

23. Гук М. Аппаратные средства локальных сетей: Энциклопедия. Санкт-Пітербург : Питер, 2000. 576 с.

24. Інструкція по налаштуванню сервера і клієнта OpenVPN. URL: https://hackware.ru/?p=5369 (дата звернення: 15.10.2019)

25. Налаштування OpenVPN серверу. URL: https://wiki.dieg.info/openvpn (дата звернення: 20.10.2019)

ДОДАТОК А

Скрипт генерації конфігураційних файлів на сервері

```
from flask import Flask
from flask import request
from flask_basicauth import BasicAuth
from switch import switch
import json, subprocess, os, logging

LOG_DIR = os.path.join(os.getcwd(), 'logs')
LOG_NAME = 'main.log'
LOG_PATH = os.path.join(LOG_DIR, LOG_NAME)
LOG_FORMAT = '[%(asctime)s] %(levelname)s: %(message)s'
logging.basicConfig(filename=LOG_PATH, level=logging.INFO, format=LOG_FORMAT,
datefmt='%m/%d/%Y %I:%M:%S')

OVPN_PATH = '/etc/openvpn'
CONF_DIR = OVPN_PATH + '/vpn-access/configs/'
ERSA_PATH = OVPN_PATH + '/easyrsa'
ERSA_BIN = ERSA_PATH + '/easyrsa'
CCD_PATH = OVPN_PATH + '/server/ccd/'
PKI_PATH = ERSA_PATH + '/pki/'

app = Flask(__name__)

app.config['BASIC_AUTH_USERNAME'] = 'admin'
app.config['BASIC_AUTH_PASSWORD'] = 'admin'
basic_auth = BasicAuth(app)

# {"method":"create", "login":"user", "days":"20", "access":["1.1.1.1", "2.2.2.2", "3.3.3.3"]}
# {"method":"get", "login":"user"}
```

```
# {"method":"revoke", "login":"user"}

@app.route('/', methods=['GET', 'POST'])
@basic_auth.required
def index():
    logging.info('Received a request from: ' + request.remote_addr + ' Request data: ' +
str(request.data))
    data = json.loads(request.data)

    if 'method' in data and 'login' in data:
        for case in switch(data['method']):
            if case('create'):
                create_func = create_vpn(data['login'], data['days'], data['access'])
                return answer_func(json.dumps({'message': create_func['message']}), create_func['code'])
                break
            if case('get'):
                get_func = get_vpn(data['login'])
                return answer_func(json.dumps({'message': get_func['message']}), get_func['code'])
                break
            if case('upd_act'):
                account_func = upd_account(data['login'], data['access'])
                return answer_func(json.dumps({'message': account_func['message']}),
account_func['code'])
                break
            if case('upd_acs'):
                access_func = upd_access(data['login'], data['old_ip'], data['new_ip'])
                return answer_func(json.dumps({'message': access_func['message']}),
access_func['code'])
                break
            if case('revoke'):
                revoke_func = revoke_vpn(data['login'])
                return answer_func(json.dumps({'message': revoke_func['message']}),
revoke_func['code'])
                break
```

```
            if case(): #default
                return answer_func(json.dumps({'message': 'Method Not Allowed.'}),405)
    else:
        return answer_func(json.dumps({'message': 'Bad Request.'}),400)

def answer_func(message, code):
    return message, code, {'Content-Type': 'application/json'}

def create_vpn(login, days, access):
    config_data = ''
    if int(days) <= 0:
        return {'code': 400, 'message': 'Days must be greater than 0.'}
    if os.path.exists(CONF_DIR + login + '.ovpn'):
        logging.warning('Create: Client with login: "' + login + '" already exists.')
        return {'code': 409, 'message': 'Config file already exists.'}
    else:
        subprocess.check_output('cd ' + ERSA_PATH + ' && ' + ERSA_BIN + ' --batch --req-cn=' +
login + ' gen-req ' + login  + ' nopass', shell=True)
        subprocess.check_output('cd ' + ERSA_PATH + ' && ' + ERSA_BIN + ' --batch --days=' +
days + ' sign-req client ' + login, shell=True)

        file_list = [[OVPN_PATH + '/genconf/config.ovpn', '', ''],
            [PKI_PATH + 'ca.crt', '<ca>\n', '</ca>'],
            [PKI_PATH + 'issued/' + login + '.crt', '\n<cert>\n', '</cert>'],
            [PKI_PATH + 'private/' + login + '.key', '\n<key>\n', '</key>'],
            [PKI_PATH + 'ta.key', '\n<tls-auth>\n', '</tls-auth>\n']]
        for i in file_list:
            path, tag_b, tag_e = i
            with open(path, 'r') as raw_data:
                config_data = config_data + tag_b + raw_data.read() + tag_e

        create_ccd(login, access)

        with open(CONF_DIR + login + '.ovpn', 'w') as w:
```

```
            w.write(config_data)

        logging.info('Create: Config for login: "' + login + '" was created and sent.')
        return {'code': 200, 'message': config_data}

def get_vpn(login):
    if os.path.exists(CONF_DIR + login + '.ovpn'):
        with open(CONF_DIR + login + '.ovpn', 'r') as r:
            read_data = r.read()
            logging.info('Get: Config with login: "' + login + '" was send.')
            return {'code': 200, 'message': read_data}
    else:
        logging.warning('Get: Config for account: "' + login + '" not found.')
        return {'code': 404, 'message': 'Config for account: ' + login + ' not found.'}

def revoke_vpn(login):
    if os.path.exists(CONF_DIR + login + '.ovpn'):
        subprocess.check_output('cd ' + ERSA_PATH + ' && /usr/bin/echo "yes" | ' + ERSA_BIN + '
revoke ' + login, shell=True)
        subprocess.check_output('cd ' + ERSA_PATH + ' && ' + ERSA_BIN + ' gen-crl', shell=True)
        subprocess.check_output('rm -rf ' + CONF_DIR + login + '.ovpn', shell=True)
        remove_ccd(login)
        logging.info('Revoke: Config with login: "' + login + '" was revoked. CRL file was generate.')
        return {'code': 200, 'message': 'Certificate revoked.'}
    else:
        logging.warning('Revoke: Config for account: "' + login + '" not found.')
        return {'code': 404, 'message': 'Config for account: ' + login + ' not found.'}

def upd_account(login, access):
    if os.path.exists(CCD_PATH + login) and os.path.exists(CONF_DIR + login + '.ovpn'):
        remove_ccd(login)
        create_ccd(login, access)
        return {'code': 200, 'message': 'Account updated.'}
    else:
```

```
        logging.warning('Update account: Config for account: "' + login + '" not found.')
        return {'code': 404, 'message': 'Config for account: ' + login + ' not found.'}

def upd_access(logins, old_ip, new_ip):
    for login in logins:
        with open(CCD_PATH + login, 'r') as access:
            data = access.read()
            replace_data = data.replace(old_ip, new_ip)
        with open(CCD_PATH + login, 'w') as access:
            access.write(replace_data)
    return {'code': 200, 'message': 'Access updated.'}

def create_ccd(login, access):
    if not os.path.exists(CCD_PATH + login):
        subprocess.check_output('touch ' + CCD_PATH + login, shell=True)
    with open(CCD_PATH + login, 'w') as access_data:
        for a in access:
            access_data.write('push "route ' + a + ' 255.255.255.255"\n')

def remove_ccd(login):
    subprocess.check_output('rm -rf ' + CCD_PATH + login, shell=True)
```

ДОДАТОК Б

Реалізація VpnManager.php

```
<?php

namespace app\managers;

use app\models\Access;
use app\models\Client;
use app\models\User;
use InvalidArgumentException;
use Yii;
use yii\web\UnauthorizedHttpException;

class VpnManager
{
   private $username, $url, $auth, $client_id, $action, $expiry_date, $logins, $old_ip, $new_ip;
   private $access = ;

   const ACTION_GET = 'get', ACTION_CREATE = 'create', ACTION_REVOKE = 'revoke',
ACTION_UPDATE_ACCOUNT = 'upd_act', ACTION_UPDATE_ACCESS  = 'upd_acs';
   const SUCCESS_STATUS = 200;

   public function __construct()
   {
      $this->auth =  base64_encode(\Yii::$app->params['vpnConfig']['username'].':'.\Yii::$app-
>params['vpnConfig']['password']);
      $this->url = \Yii::$app->params['vpnConfig']['host'].':'.\Yii::$app->params['vpnConfig']['port'];
   }

   public function setClient($id)
   {
      $this->client_id = $id;
   }

   public function setUsername($username)
   {
      $this->username = $username;
   }

   public function setThisAccess($IP)
   {
      $this->access = $IP;
```

```
}

public function setAction($action)
{

   if ($action != self::ACTION_CREATE
      && $action != self::ACTION_GET
      && $action != self::ACTION_REVOKE
      && $action != self::ACTION_UPDATE_ACCOUNT
      && $action != self::ACTION_UPDATE_ACCESS) {
      throw new \InvalidArgumentException('action must be in class const');
   }

   $this->action = $action;
}

public function setExpiry($expiry)
{
   $this->expiry_date = $expiry;
}

public function setAccess(array $access_ids)
{
   foreach ($access_ids as $access_id) {
      $access = Access::findOne($access_id);
      if ($access){
         $this->access = $access->IP;
      }
   };
}

public function setOldIp($ip)
{
   $this->old_ip = $ip;
}

public function setNewIp($ip)
{
   $this->new_ip = $ip;
}

public function setLogins(array $logins)
{
   $this->logins = $logins;
}

public function send()
{
   if ($this->client_id && Client::findOne($this->client_id)) {
      $client_name = Client::findOne($this->client_id)->client_name;
   }
```

```
$interval = date_create(date('Y-m-d'))->diff( new \DateTime($this->expiry_date) );
if ($this->action === self::ACTION_CREATE) {
    $data = [
        'method' => $this->action,
        'login' => $this->username,
        'days' => strval($interval->days),
        'access' => $this->access,
    ];
}
elseif ($this->action === self::ACTION_UPDATE_ACCOUNT) {
    $data = [
        'method' => $this->action,
        'login' => $this->username,
        'access' => $this->access,
    ];
}
elseif ($this->action === self::ACTION_UPDATE_ACCESS) {
    $data = [
        'method' => $this->action,
        'login' => $this->logins,
        'old_ip' => $this->old_ip,
        'new_ip' => $this->new_ip
    ];
}
else {
    $data = [
        'method' => $this->action,
        'login' => $client_name,
    ];
}

$data = json_encode($data);

$headers = [
    'Content-Type: application/json',
    'Accept: application/json',
    'Authorization: Basic '.$this->auth
];

$client = new CurlManager();

$client->doRequest($this->url, $data, $headers);

if ($client->getStatus() == CurlManager::BAD_REQUEST_CODE){
    Yii::$app->session->setFlash('Error', 'Bad request');
    return false;
}
else if ($client->getStatus() == CurlManager::UNAUTHORIZED_CODE){
    Yii::$app->session->setFlash('Error', 'Unauthorized on VPN server');
    return false;
```

```
        }else if ($client->getStatus() == CurlManager::NOT_FOUND_CODE){
            if ($this->action !== self::ACTION_REVOKE) {
                Yii::$app->session->setFlash('Error', 'Config not found');
            }
            return false;
        }else if ($client->getStatus() == CurlManager::EXIST_CODE){
            Yii::$app->session->setFlash('Error', 'Config file with this username is already exists.');
            return false;
        }else if ($client->getStatus() == CurlManager::SERVER_ERROR_CODE || $client-
>getStatus() == CurlManager::SERVER_CONNECTION_CODE){
            Yii::$app->session->setFlash('Error', 'Server error');
            return false;
        }


        return $client->getResponse();
    }
}
```

ДОДАТОК В

Реалізація відправки API запиту дій з файлами на сервер

```
private function createVpnUser(){
      try{
         $manager = new VpnManager();
         $manager->setUsername($this->client_name);
         $manager->setAction(VpnManager::ACTION_CREATE);
         $manager->setExpiry($this->end_date);
         $manager->setAccess($this->access_ids);
         $manager->send();
         return true;
      }catch (\Exception $exception){
         Yii::$app->session->setFlash('Error', 'Connection failed to vpn service');
         return false;
      }
   }
   private function updateVpnUser()
   {
      try{
         $manager = new VpnManager();
         $manager->setUsername($this->client_name);
         $manager->setAction(VpnManager::ACTION_UPDATE_ACCOUNT);
         $manager->setAccess($this->access_ids);
         $manager->send();
         return true;
      }catch (\Exception $exception){
         Yii::$app->session->setFlash('Error', 'Connection failed to vpn service');
         return false;
      }
   }
   private function deleteVpnUser()
   {
      try{
         $manager = new VpnManager();
         $manager->setClient($this->id);
         $manager->setAction(VpnManager::ACTION_REVOKE);
         $manager->send();
         return true;
      }catch (\Exception $exception){
         Yii::$app->session->setFlash('Error', 'Connection failed to vpn service');
         return false;
      }
```

```
}
```