Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Навчально-науковий інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

студента Пономаренка Андрія Юрійовича
(ПІБ)
академічної групи 123м-22-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему Обґрунтування параметрів комп'ютерної системи ТОВ "ФК"Форза" з детальною розробкою веб-додатку для роботи з базою даних підприємства
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Шедловська Я. І.			
розділів:				
Синтез системи	Доц. Бешта Д. О.			
Розробка програмного забезпечення	Ас. Панферова Я. В.			
Рецензент				
Нормоконтролер	доц. Шедловська Я. І.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____Гнатушенко В.В.
(підпис) (прізвище, ініціали)

"06" вересня 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра

студента Пономаренка А.Ю. **академічної групи** 123м-22-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»

за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)

на тему " Обґрунтування параметрів комп'ютерної системи ТОВ "ФК"Форза" з детальною розробкою веб-додатку для роботи з базою даних підприємства

затверджену наказом ректора НТУ «Дніпровська політехніка» від 09.10.2023 №1227-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	10.10.2023
Теоретична частина	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	25.10.2023
Синтез системи	Розробка комп'ютерної системи	15.11.2023
Розробка програмного забезпечення	Розробка програмного забезпечення	29.11.2023
Експериментальний розділ	Проведення і обробка результатів експерементів	06.12.2023

Завдання видано _____ доц.. Шедловська Я.І.
(підпис керівника) (прізвище, ініціали)

Дата видачі 6 вересня 2023 р.

Дата подання до екзаменаційної комісії 10.12.2023

Прийнято до виконання _____ Пономаренко А.Ю.

РЕФЕРАТ

Пояснювальна записка: 95 с., 33 рис., 10 табл, 1 дод., 15 джерел.

КОМП'ЮТЕРНА СИСТЕМА, ОБЛАДНАННЯ, ВЕБ-ДОДАТОК, МОДЕЛЬ-КОНТРОЛЛЕР-ПРЕДСТАВЛЕННЯ

Об'єкт розробки: комп'ютерна система товариства з обмеженою відповідальністю «Фінансова компанія «Форза».

Мета роботи: розробка комп'ютерної системи для підтримки функціоналу компанії та веб-додатку, розробленого в даній роботі. Обґрунтування структури та параметрів комп'ютерної системи для підтримки функціонування мережі між відділами підприємства та використання веб технології для роботи з базою даних підприємства.

У теоретичному розділі розглянуто та проаналізовано архітектури клієнт-серверних веб-програм, мови програмування для їх створення, фреймворки для роботи з базами даних.

У розділі синтезу комп'ютерної системи створено технічні вимоги до обладнання системи, проаналізована структурна схема підприємства, та підібрано обладнання для системи.

У розділі розробки програмного забезпечення реалізовано створення програмного веб-забезпечення для роботи з базою даних , описані алгоритми роботи та її логічні функції.

В експериментальному розділі суттю експерименту є тестування працездатності функцій розробленого веб-додатку.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ТОВ – товариство з обмеженою відповідальністю

ФК – Фінансова компанія

СКБД – система керування базами даних

MVC – модель- представлення -контролер

MVP – модель-представлення-презентер

HTML – мова гіпертекстової розмітки

ООП – об’єктно-орієнтоване програмування

JVM – віртуальна машина Java

ОС – операційна система

ВСТУП

Сьогодні комп’ютерні системи використовуються скрізь – в аптеках, друкарнях, магазинах тощо. Дана тенденція спричинена розповсюдженням використання інформаційних технологій. Наслідком цього являє собою цифровізація та автоматизація процесів бізнесу. Сфера банкінгу не є винятком. Прикладом цифровізації банків є можливість використовувати можливості банку в режимі онлайн. Задля цього банк повинен мати відповідне програмне забезпечення та стійку, захищену та здатну до масштабування комп’ютерну систему.

Товариство з обмеженою відповідальністю «Фінансова компанія «Форза» це онлайн банкінг, який надає послуги з кредитування, дебетування та надання послуг з факторингу.

Прикладом необхідного програмного забезпечення задля реалізації роботи онлайн з базою даних підприємства є веб-додаток. Веб-додаток в даному випадку являє собою програмний продукт, за допомогою якого відбуватимуться виклики до бази даних з відображенням результату їх виконання в режимі онлайн.

В кваліфікаційній роботі розглядається комп’ютерна система товариства з обмеженою відповідальністю «Фінансова компанія «Форза». Дана комп’ютерна система буде використовуватись підприємством для роботи офісів та підтримки серверу з веб-додатком для роботи з базою даних.

Для вирішення даної задачі підприємство повинно мати функціонуючу комп’ютерну систему і розроблене функціональне програмне забезпечення в вигляді веб-додатку. Це завдання входить в сферу діяльності магістра спеціальності 123 «Комп’ютерна інженерія».

Об’єкт дослідження: комп’ютерна система та мережа ТОВ «ФК «Форза», що забезпечує роботу всіх філіалів підприємства.

Предмет дослідження: структура комп’ютерної системи та мережі ТОВ «ФК «Форза».

Мета і завдання дослідження: обґрунтування структури та параметрів комп’ютерної системи для її модернізації, з метою забезпечення функціонування мережі між відділами підприємства та використання веб технології для роботи з базою даних підприємства.

Ідея роботи: створення веб додатку для роботи користувачів та співробітників підприємства з базою даних та використання мережі для доступу до додатку.

Наразі існує необхідність у модернізації комп’ютерної системи, задля збільшення швидкості роботи банку та забезпечення надійного рівня захисту даних, саме через специфіку банківської сфери діяльності зазначені параметри є основними критеріями системи. У свою чергу, створення веб додатку дасть можливість співробітникам компанії більш ефективно працювати з даними, та виконувати рядові задачі, такі як видачі кредитів, депозитів, створення рахунків, а для звичайних же користувачів веб додаток відкриє доступ до персонального кабінету з можливостями переглядати та керувати власними рахунками в режимі онлайн.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАДАЧІ

1.1 Стисла характеристика галузі та умов застосування КС

В наш час, бізнес процеси починають рухатись у цифровий формат, як наслідок впливу інформаційних технологій на способи комунікації бізнес процесів з клієнтами. Основним методом впливу є цифровізація. Цифровізація як процес являє собою введення цифрових технологій в різноманітні сфери життя, таких як медицина, освіта тощо. Банківська система невиняткова в цьому. Цифровізація банківської системи включає в себе впровадження нових технологій, автоматизацію процесів, зміни у взаєминах з клієнтами та забезпечення кібербезпеки[1].

Важливу роль в цифровізації банківської системи відіграє готовність банків адаптуватися до швидкозмінного технологічного середовища. Інноваційні технології вимагають від банків постійного оновлення своїх процесів та систем[1]. Через це комп’ютерна система банку та його програмна база повинна проходити планові оновлення задля введення в роботу банку нових методів роботи з клієнтами та підтримки сучасної технологічної бази.

З іншої сторони, КС банку має забезпечувати надійний рівень захисту даних як на фізичному рівні, так і на логічному. Повсюдне використання інформаційних технологій відкрило як нові шляхи розвитку для бізнес процесів, так і додало необхідність посилити системи захисту даних. Проектування банківської системи повинно приділити особливу увагу захисту клієнтських персональних даних, запобіганню шахрайству шляхом проведення інструктажів з кібербезпеки як співробітникам банку, так і його клієнтам, та перешкоджанню кібератакам завдяки технічним та програмним засобам.

Як наслідок використання інформаційних технологій в банківській сфері, поліпшився клієнтський досвід роботи з системою банка. Прикладом втручання ІТ в роботу банків є широке використання мобільних та веб-додатків, які дозволяють клієнтам виконувати фінансові операції, такі як

перевірка рахунків та грошові перекази, незалежно від їх місцезнаходження та у будь-який час. Це робить банківські послуги доступнішими та зручнішими для їх використання клієнтами[1].

Використання веб-додатків як співробітниками банку, так і його клієнтами дозволяє автоматизувати рутинні операції. Через це зменшується час та зусилля на виконання операцій. Це дозволяє банкам швидше та ефективніше обробляти операції, зменшує ймовірність помилок та сприяє високій якості обслуговування[1].

Підсумовуючи, банківській сфері сьогодення необхідно мати свою надійну та захищену комп'ютерну систему та прикладне програмне забезпечення для робити з базами даних реалізовану шляхом створення веб-додатку.

1.2 Характеристика і структура об'єкта впровадження

В даній роботі розглядається товариство з обмеженою відповідальністю «Фінансова компанія «Форза». Компанія позиціонує себе як онлайн банкінг, тому основними бізнес процесами є робота з клієнтами. Прикладом послуг, які надає компанія є кредитування, дебетування та надання послуг з факторингу. ТОВ «ФК «Форза» реалізує свої послуги на території України, обслуговує та професіонально проводить фінансові операції з клієнтами. Штат компанії містить у собі 100 працівників.

Через те, що підприємство займається фінансовими операціями з клієнтами, для комп'ютерної системи необхідно забезпечити необхідний рівень захисту даних та розробити програмне забезпечення, яке дозволяє працювати з клієнтською базою.

Підприємство складається з декількох структурних підрозділів, які виконують внутрішню та зовнішню діяльність компанії, і виконують характерні операції для підтримки зворотного зв'язку з клієнтами. ТОВ «ФК «Форза» розподіляє зони відповідальності між структурними підрозділами,

що дозволяє кожному опрацьовувати окремий напрямок діяльності задля максимізації результату роботи підприємства.

Організаційно-управлінська структура товариства з обмеженою відповідальністю «Фінансова компанія «Форза» зіставлена з трьох рівнів управління: верхнього, середнього і нижчого[2].

До верхнього рівню відносять генерального директора, який керує філіалами підприємства в загальному вигляді і є відповідальним за звіти з бухгалтерії та відділу технічної підтримки філіалів[2].

На середньому рівні розташовані керівники підрозділів підприємства, які виконують обов'язки з забезпечення діяльності ТОВ «ФК «Форза» та виконання бізнес операцій і завдань, затверджених на вищому рівні[2].

До нижчого рівня управління належить персонал підрозділів підприємства, який підпорядковується середньому та вищому рівню управління компанії і виконує завдання, поставлені там. Схема організаційної структури ТОВ «ФК «Форза» зображена на рисунку 1.1.[2]

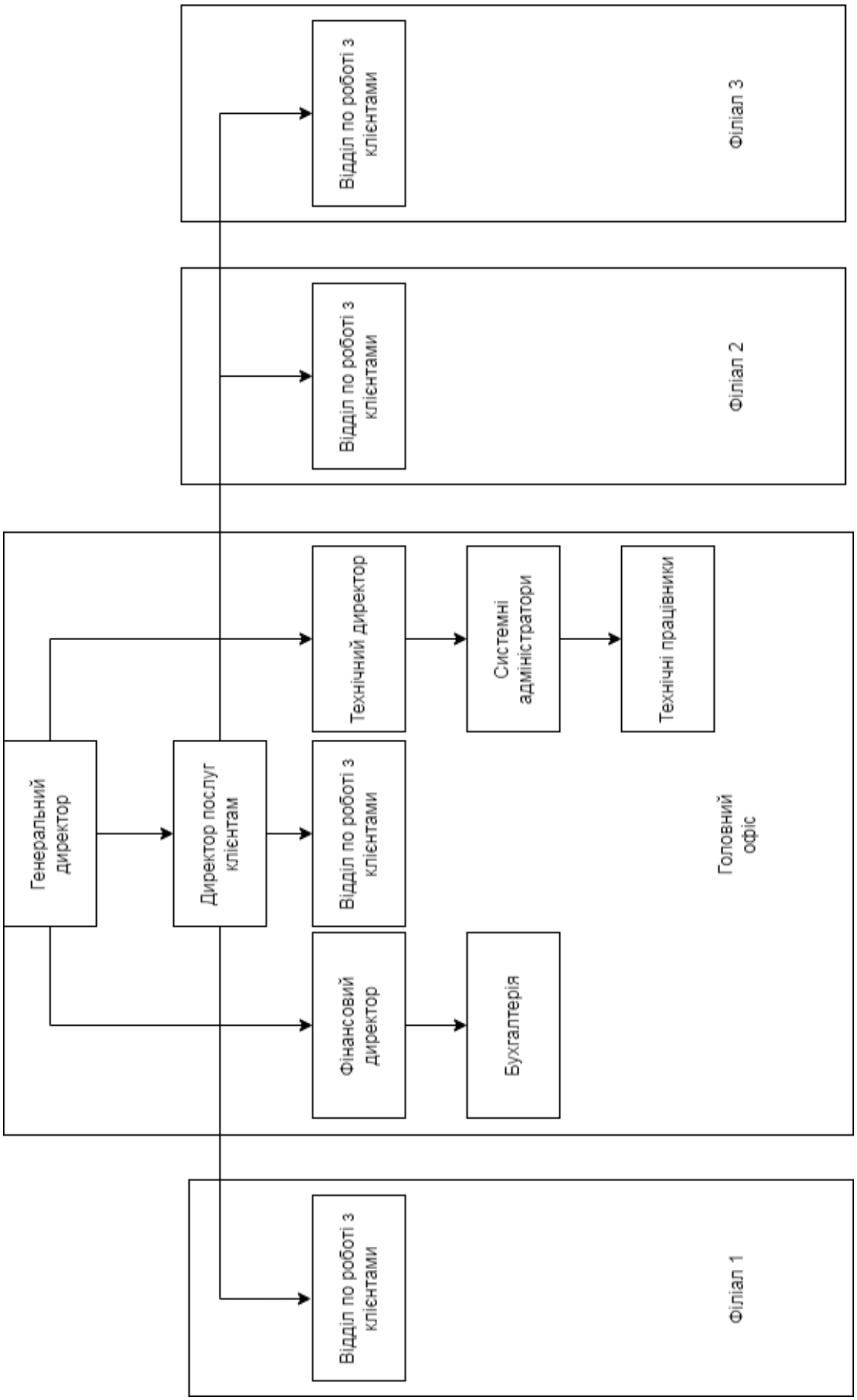

Рисунок 1.1 – Схема організаційної структури

До складу компанії входять головний офіс та три філіали. Головний офіс знаходиться за адресою м. Київ, вул. В. Гетьмана, 6, адреси філіалів:

– м. Київ, проспект Перемоги, 94/1;
– м. Київ, вул. Старовокзальна,23;
– м. Київ, вул. Костянтинівська,2А.

Схема георозміщення компонентів КС зображена на рисунку 1.2

В головному офісі знаходяться бухгалтерія та відділ технічної підтримки, який є відповідальним за технічне обладнання комп'ютерної системи компанії, і відділ по роботі з клієнтами. Філіали містять на своїй базі ресурси, необхідні для виконування бізнес операцій ТОВ «ФК «Форза»[2].

Комп'ютерна система у вигляді корпоративної мережі дозволить компанії збільшити продуктивність роботи та забезпечить зв'язок між базою даних, розробленим веб-додатком та філіалами.

Рисунок 1.2 – Схема георозміщення

1.3 Принципи, технічні способи та математичні методи інформаційного забезпечення КС ТОВ «ФК «ФОРЗА»

КС для ТОВ «ФК «Форза» створена за допомогою технології локальних обчислювальних мереж.

Мережа повинна мати такі властивості:

- масштабованість;
- надмірність;
- можливість віртуалізації;
- безпеку мережевих пристроїв;
- використання обладнання, яке має міжнародну сертифікацію.

Передача інформації між відділами компанії відбувається за допомогою звітів, створених співробітниками підприємства. Інформація про клієнтів міститься в базі даних підприємства. Веб-додаток забезпечує доступ до бази даних як співробітників, так і клієнтів, завдяки різним інтерфейсам та рівням доступу. Інформація, що проходить через зовнішню мережу, повинна шифруватися заради безпеки персональних даних клієнтів та забезпечення безпеки бази даних підприємства. Вся інформація та база даних зберігається на серверах в головному офісі.

1.4 Завдання і мета роботи

Метою роботи є організація комп’ютерної системи для ТОВ «ФК «Форза» із детальною розробкою програмного забезпечення в вигляді веб-додатку для роботи з базою даних підприємства.

Для вирішення поставленої мети в роботі вирішуються наступні завдання:

- аналіз мережевої архітектури корпоративної мережі ТОВ «ФК «Форза» та кабельної системи з метою вибору обладнання для комп’ютерної системи;
- розробка фізичної та логічної топології мережі підприємства;
- запровадження безпеки мережі;

– вибір архітектури для програмного забезпечення ТОВ «ФК «Форза»;
– вибір мови програмування;
– вибір фреймворку для роботи з Web;
– налаштування параметрів аудиту для користувачів;
– розробка логічної частини ПЗ;
– налаштування конфігурацій для роботи з базою даних.

Необхідно створити безпечну та гнучку комп’ютерну систему для підприємства з можливістю масштабування, розробити раціональне та гнучке програмне забезпечення, передбачивши можливості його оновлень, і так само опрацювати питання забезпечення доступу до бази даних з різними правами доступу.

2 ТЕОРИТИЧНИЙ РОЗДІЛ

2.1 Види програмного забезпечення для онлайн сервісів

В наш час, великої популярності набуває перехід процесів в онлайн режим, котрий відбувається в усіх сферах життя – медицина, розваги та бізнес сфери. Банківська справа не є винятком у цих процесах. Перехід відбувається завдяки поширенню та активному використанню інформаційних технологій, зокрема комп'ютерних мереж. Це спричинило необхідність мати кожному підприємству власну комп'ютерну систему та необхідне програмне забезпечення для виконання бізнес процесів.

Оскільки ТОВ «ФК «Форза» представляє себе як онлайн-банкінг, для роботи йому необхідно ПЗ, яке має змогу працювати за допомогою Інтернету. Прикладами такого забезпечення є мобільні додатки та веб-додатки.

Мобільний додаток – це додаток, який створений для роботи з UNIX-системами, такими як Android, для використання його за допомогою смартфону або будь-якого мобільного пристрою. Прикладами таких додатків є сервіси Google, такі як Gmail, Youtube тощо. Основними інструментами для створення мобільних додатків є мови програмування Java та Kotlin з фреймворками, наприклад Codename One. Даний варіант програмного забезпечення має свої переваги та недоліки. Недоліками є висока вартість даного типу розробки та його вразливості до шкідливого програмного забезпечення. В найгіршому випадку шкідливе ПЗ може проникнути в мобільний додаток, та, шляхом зміни запитів, має шанс зруйнувати серверну частину додатку. Очевидною перевагою є зручність для користувача.

Веб-додаток – це програмне забезпечення, яке створене для роботи у браузері користувача. Прикладами даних додатків є web-версії популярних соціальних мереж, таких як Facebook, Telegram, Instagram. Для створення веб-додатків використовують мови програмування C# та Java з відповідними фрейморками – Entity Framework та Spring. Використання такого типу ПЗ, як веб-додаток має свої позитивні та негативні сторони. З позитивної сторони,

розробка такого прикладного програмного забезпечення обходиться дешевше та такий додаток можливо запустити з будь-якого пристрою, який підтримує браузер. До недоліків можна віднести проблеми з розробкою UI, так як HTML та CSS може бути недостатньо для реалізації певних дизайнерських рішень.

Підсумовуючи вищесказане, у вигляді прикладного програмного забезпечення для ТОВ «ФК «Форза» підходить веб-додаток, так як він дозволить масштабніше використовувати банківські послуги, що сприятиме розвитку компанії.

2.2 Шаблони проектування веб-додатків

Для проектування веб-додатків використовують такі шаблони:

– MVC;
– MVP;
– Створення додатку з використанням веб-форм.

MVC – шаблон проектування model-view-controller, який використовує окремі модулі для реалізації різних частин програмного забезпечення. Model відповідає за класи додатку та його логічну частину, view представляє собою візуальну частину, яку бачить користувач, і controller – частина програми, яка оброблює зміни користувача. Схема роботи шаблону зображена на рисунку 2.1.

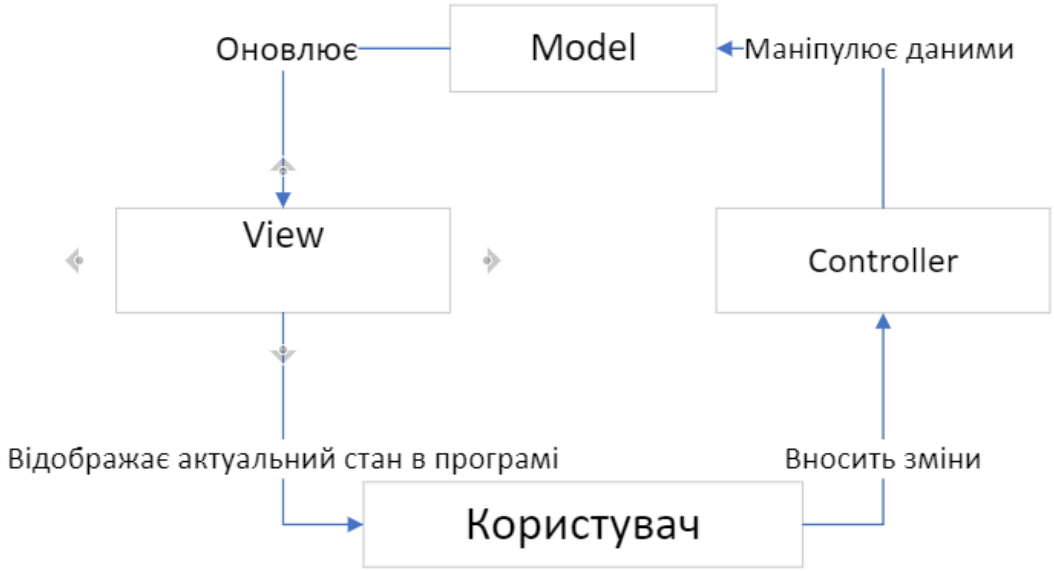

Рисунок 2.1 – Схема роботи шаблону MVC

Згідно з рисунком, користувач при запуску веб-додатку бачить її view частину, яка представляє собою HTML сторінку, вносить зміни в неї та викликає один з тригерів програми. Далі в роботу програми вступає controller, який являє собою перевалочним пунктом між логікою додатку та його відображеним даним для користувача, і передає в model нові значення даних. В model відбуваються зміни відповідно до логічної частини програми і нове подання відображається у view. Перевагами даного рішення є розділення логічної і візуальної частини додатку, це дає змогу розробляти дані секції окремо та робить так, що користувач не має доступу до функціонального коду додатку напряму, також це полегшує розробку та масштабування додатку, підтримка викликів HTTP. Недоліком є те, що для підтримки запущеного веб-додатку потрібні достатні ресурси сервера, на якому він знаходиться.

MVP – шаблон проектування model-view-presenter, який є похідним від MVC. В даному шаблоні model відповідає за класову структуру додатку, view ,ідентично до MVC, відображає інтерфейси для користувачів, а presenter

відповідає за логічну частину додатку. Схема роботи шаблону проектування зображена на рисунку 2.2.

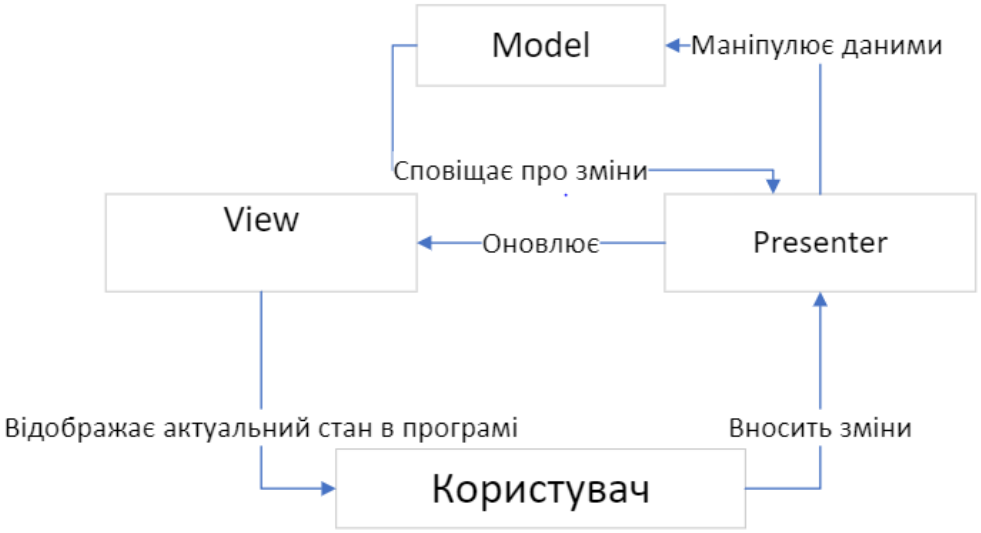

Рисунок 2.2 – Схема роботи шаблону MVP

Користувач після запуску веб-додатку працює з view у вигляді HTML сторінок, вносить зміни та викликає логічну частину програми. Після цього, presenter обробляє дані і оновлені дані відправляє в model для перевірки та парсингу. Отримавши відповідь від model, presenter оновлює view, заповнюючи його обробленою інформацію логічними діями програми. Перевагами даного шаблону є те, що такий проект легше автоматизувати, тестувати та він має відділену бізнес логіку від користувача, підтримка викликів HTTP. До недоліків слід відзначити те, що додаток потребує потужного серверу для підтримки і максимальна результативність шаблону відбувається тільки тоді, коли view має декілька HTML сторінок, до яких потрібно надавати доступ одному й тому самому коду виконання.

Веб-форми – це сторінки, які користувачі викликають у браузері. Дані сторінки можуть бути написані за допомогою комбінації HTML, клієнтського сценарію, серверних елементів керування та коду. Коли користувачі викликають сторінку, вона компілюється та виконується на сервері

фреймворком, а потім фреймворк генерує розмітку HTML, яку браузер може відобразити[3].

Даний шаблон проектування буде розглянуто на прикладі веб-форм ASP.NET. Веб-форми ASP.NET мають такі властивості:

- сумісні з будь-яким браузером або пристроєм, який може підтримувати браузерні програми;
- дозволяють використовувати будь-яку мову програмування платформи .NET;
- є гнучкими, тому що можливе додавання компонентів користувачів та інших розробників.

Використання веб-форм має такі переваги:

- підтримка HTTP викликів,що дозволяє розробляти бізнес додатки;
- має вбудований функціонал для відображення сторінок різними мовами;
- є результативним в малих командах розробників та дизайнерів, бо ASP.NET Web Forms має великий доступний функціонал;
- є дешевшим, ніж інші шаблони проектування.

До недоліків можна віднести неможливість контролювати повністю поведінку програми програмістом через те, що користувачі такого програмного забезпечення можуть використовувати різні браузери з різними налаштуваннями, що може викликати невідомі стани в програмі, які або не будуть оброблюватись взагалі, або відображатимуть некоректну відповідь.

2.3 Мови програмування для розробки веб-додатків

Основними мовами для розробки веб-додатків є C# на платформі .NET з використанням ASP та Java.

2.3.1 C# та .NET

C# – об'єктно-орієнтована мова програмування з безпечною статичною системою типізації, розроблена спеціально для платформи .NET. Для

розробки даною мовою програмування використовують IDE Microsoft Visual Studio.

На сьогоднішній день, C# являє собою одну з найпопулярніших та найпотужніших мов програмування, яка швидко оновлюється та розвивається і має багато вакансій в галузі IT. За допомогою цієї мови можна програмувати безліч різноманітних програм, незалежно від їх масштабу, середовища використання та кількості користувачів, які будуть користуватися програмним забезпеченням. Тобто, C# є гарним інструментом для реалізації веб-сервісів та веб-додатків.

C# не молода мова і, як і вся платформа .NET, вже мала безліч можливостей для перевірки. Перша версія була створена разом з релізом Microsoft Visual Studio .NET у лютому 2002 року. Останньою на даний момент версією є C# 12, яка вийшла 14 листопада 2023 разом з релізом .NET 8 [4].

C# є об’єктно-орієнтованою мовою програмування і через це має багато подібного з іншими об’єктно-орієнтованими мовами програмування, такими як Java та C++. C# підтримує основні принципи ООП – поліморфізм, інкапсуляцію, успадкування та абстрагування, також має статичну типізацію та може використовувати перевантаження операторів. Об’єктно-орієнтований підхід дозволяє вирішувати завдання з розробки великих, але в той же час гнучких, здатних до масштабування додатків. І C# продовжує активно розвиватися і з кожною новою версією з’являється більше можливостей [4].

Фреймворк .NET розроблений як потужну та багатофункціональну платформу для створення різноманітних додатків. Основними перевагами даної платформи є:

- підтримка декількох мов програмування;
- кросплатформність;
- велика кількість створених для платформи бібліотек і класів.

Реалізацією підтримки декількох мов для .NET є загальномовне середовище виконання віртуальної машини Common Language Runtime, завдяки чому код ,який написаний підтримуваними мовами програмування,

компілюється в Common Intermediate Language – машинний код для CLR. Це дозволяє створювати окремі модулі для додатків різними мовами програмування.

Прикладом кросплатформеності платформи .NET є те, що середовище дозволяє створювати, з невеликими змінами та обмеженнями, та виконувати програми на різних операційних системах. На даний момент, остання версія .NET 8 має підтримку ОС Windows, MacOS, Linux.

Велика кількість бібліотек та класів для платформи .NET дозволяє створювати за допомогою C# програми прикладного типу різного виду, наприклад веб-сервіси, веб-додатки та десктопні додатки. В .NET для створення веб-додатків та веб-сервісів використовується ASP.NET.

ASP.NET розроблена як частина архітектури .NET та може:

- працювати в реальному часі;
- використовувати різноманітні веб API;
- використовуватися для розроблення мобільних, веб додатків та сайтів;
- підтримувати одно сторінкові програми;
- працювати з веб-хуками та HTTP-запитами[5].

Для веб-додатків розроблено різні модулі, такі як ASP.NET Web forms, ASP.NET MVC та ASP.NET Single Page Application. Відмінність між ними в тому, що вони використовують різні шаблони для проектування програмного забезпечення та, виходячи з цього, мають різні набори спеціальних функцій, бібліотек та класів.

ASP.NET MVC представляє собою потужний, заснований на шаблонах, спосіб створення динамічних веб-сайтів, який забезпечує чітке розділення завдань і надає розробнику повний контроль над розміткою для приємної гнучкої розробки. ASP.NET MVC містить багато функцій, які забезпечують швидку, дружню до TDD розробку для створення складних програм, які використовують найновіші веб-стандарти, оскільки використовує платформу .NET [5].

ASP.NET Web Forms використовується для створення динамічні веб-сайтів за допомогою керування подіями. Варіанти дизайну та сотні елементів керування та компонентів дозволяють швидко створювати складні, потужні сайти на основі UI-інтерфейсу з доступом до даних [5].

2.3.2 Java

Java — це об’єктно-орієнтована мова програмування, яка була випущена компанією Sun Microsystems у 1995 році[6]. При роботі з Java використовується IDE IntelliJ IDEA.

Java була розроблена як мова програмування, за допомогою якої можливо створювати додатки незалежно від платформи їх використання. Дана властивість була реалізована за допомогою створення спеціальної віртуальної машини JVM та запозичення об’єктно-орієнтованої моделі мов C/C++ з деякими відмінностями. Основна відмінність в тому, що Java не має таких низькорівневих засобів, і робота з вказівниками реалізована в автоматичному збирачеві сміття. Через це програми, написані на Java, працюють повільніше, ніж C/C++, проте сама мова програмування є легшою для опанування. Під час створення Java, розробники хотіли реалізувати наступні принципи :

- мова повинна підтримувати багатопотоковість;
- мати високу продуктивність під час виконання ПЗ;
- скомпільована програма один раз повинна запускатися будь-де;
- мова повинна бути безпечною та безвідмовною .

Реалізація принципу «скомпільовано раз, запускається всюди» виконана шляхом компіляції початкового Java коду у байт-код, який створений у вигляді спрощеного машинного коду. Таким чином, програму можна запустити на будь-якій платформі, що підтримує JVM, яка інтерпретує байткод у код, пристосований до конкретного технічного та програмного обладнання пристрою, на якій запущено JVM.

Java підходить для розробки багатьох видів програмного забезпечення, таких як мобільні, десктоп та веб-додатки. Дана мова навіть може

використовуватися при написанні операційних систем, прикладом цього є всім відома ОС Android.

На даний момент, Java є однією з найпопулярніших мов програмування через свою легкість, можливість кросплатформенного використання, надійності та продуктивності написаних нею програм.

Розробка веб-додатків за допомогою даної мови програмування дозволяє інтегрувати в нього будь-які цифрові та логічні рішення, та дозволяє використовувати програмний продукт у будь-якій сфері – медицині, розвагах, банкінгу, освіті тощо. При розробці використовується фреймворк Spring.

Перевагами створеного веб-додатку мовою Java є:

– кросплатформеність додатку;
– надійність;
– високий рівень безпеки;
– висока продуктивність.

В результаті створення веб-додатків на Java стало стандартом якості для реалізації великих проектів з високими вимогами до продуктивності, відмовостійкості та безпеки, таких як фінансові, банківські та торговельні електронні системи.

2.4 Фреймворки для роботи з БД

Для розглянутих мов програмування найчастіше використовують такі фреймворки – Entity Framework для C# та Spring для Java.

2.4.1 Entity Framework Core

Entity Framework використовує платформу .NET та ORM-технологію для доступу до даних. Фреймворк дозволяє використовувати бази даних незалежно від її структури та таблиць та працювати з даними як з об’єктами класів, незалежно від їх типу. Якщо фізично ми оперуємо таблицями, індексами, первинними та зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, робота йде з об'єктами [7].

Оскільки Entity Framework використовує платформу .NET, технологія доступу до даних може використовуватись кросплатформено. Тобто, завдяки стеку .NET, фреймворк може використовувати різноманітні платформи, такі як Windows Forms, різноманітні консольні програми та ASP.NET. Дана властивість дозволяє за допомогою EF розробляти різноманітні додатки під різні ОС та платформи. Для роботи з веб-додатками EF може використовувати ASP.NET MVC. Даний фреймворк постійно оновлюється під потреби сучасного світу разом з платформою .NET.

Entiry Framework створений як проект з відкритим кодом, який можна знайти за посиланням https://github.com/dotnet/efcore. EF Core розроблений так, що дозволяє працювати з багатьма системами баз даних. Тобто, наявна можливість для роботи з будь-якою СКБД, якщо для неї є потрібний провайдер в складі фреймворку або платформи .NET. Прикладом підтримуваних баз даних є MySQL, SQLite, PostgreSQL.

У EF Core також вбудований універсальний API для роботи з даними. Тобто, при зміні СКБД, основні зміни проекту будуть стосуватись лише файлів конфігурацій та підключення до бази даних, і провайдера бази даних. Весь розроблений код, який виконує запити, додає в базу інформацію або видаляє, змінювати непотрібно.

Основною концепцією фреймворка є entity, що в перекладі означає сутність. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому ця технологія передбачає роботу не з таблицями, а з об'єктами та їх колекціями [7].

Сутності, як і будь-які об’єкти, мають ряд властивостей. Властивості можуть бути як простими даними, такими як int або double, так і розробленими комплексними типами даних. Проте, як і в реляційних базах даних, кожна сутність повинна мати мінімум одну властивість, яка буде робити дану сутність унікальною. Така властивість називається ключем.

Зв’язки між сутностями можуть бути ідентичними до типових зв’язків в базах даних – один до багатьох, один до одного і багато до багатьох. Дана

властивість схожа з тим, як у реляційній базі даних відбувається взаємозв’язки за допомогою зовнішніх ключів та ідентифікаторів.

Entity Framework для роботи з СКБД використовує службу LINQ, яка представляє мову запитів до даних. За її допомоги можлива реалізація різноманітних запитів до об’єктів, використовуючи асоціативні зв’язки. EF при виконанні запитів транслюватиме LINQ вирази у вирази, які використовуються в конкретних СКБД.

Підсумовуючи вищесказане, EF Core прекрасний інструмент для роботи з базами даних та створення веб-додатків за допомогою платформи ASP.NET.

2.4.2 Spring

Spring Framework – це універсальний фреймворк для розробки веб-додатків мовою програмування Java. Він складається з багатьох невеликий фреймворків, спроектованих під різні задачі, та має модульну структуру.

Одною з особливостей фреймворку є принцип інверсії управління. Розробник не налаштовує залежності об’єктів напряму, а це робить сам фреймворк, значно зберігаючи час та запобігає виникненню помилок. Через це код стає більш модульним та гнучким для використання [8].

Для розробки веб-додатків частіше за все використовують додатковий модуль Spring Boot. Це фреймворк, який є частиною сімейства фреймворків Spring. Свою популярність він здобув завдяки тому, що спрощує розробку веб-додатків шляхом автоматичних конфігурацій за замовчуванням. Таке рішення виключає безліч проблем та незручностей, порівняно з налаштуваннями проекту, використовуючи базову версію Spring. Наприклад, Spring Framework потребує окремого налаштування HTTP-викликів та серверу для нього, в той же час в Spring Boot є вбудований для цього сервер Tomcat.

Оскільки архітектура MVC є досить популярною у веб-розробників, для неї в Spring створено спеціальний модуль Spring MVC, котрий дозволяє використовувати дану архітектуру в повній мірі.

Для роботи з даними в фреймворці реалізовано модуль Spring Data для взаємодії з різними базами даних. Він спрощує роботу з СКБД, підвищує швидкість при роботі з даними та зв’язує класи з типами даних в таблицях, використовуючи ORM.

Spring має власний функціонал для організації безпеки додатків – Spring Security. В ньому реалізовано методи для захисту веб-додатків, такі як аутентифікація та авторизація користувачів, розподілення прав доступу та дозволи для них.

Оскільки Spring використовує мову Java, то всі її переваги в розробці додатків успадковує і фреймворк. Тобто, додатки, створені за допомогою Spring/ Spring Boot є:

- кросплатформенними, для виконання необхідна лише підтримка браузера;
- з високою швидкістю роботи, що є важливо з великою кількістю оброблюваних даних;
- надійною та безпечною, шляхом створення AAA політик;
- легко масштабуються, є гнучкими.

Spring Framework, як інструментарій для створення веб-додатків, є гарним рішенням, бо з його можливостями для створення додатків, які можуть вміщувати будь-яку бізнес логіку, не виникатиме ніяких складнощів перед розробником. Ще однією перевагою є те, що платформа постійно рухається вперед та має величезну аудиторії, що сприяє регулярним оновленням та додаванням нових функцій до неї.

Підсумовуючи, Spring Framework з використанням Java має багато переваг при створенні веб-додатків, тому вибір даного фреймворку є непоганим рішенням для Java-розробників.

2.5 PostgreSQL

ТОВ «ФК «Форза» використовує базу даних PostgreSQL.

PostgreSQL це об’єктно-реляційна система керування базами даних, яка є альтернативою як комерційних СКБД, так і СКБД з відкритим кодом.

PostgreSQL є однією з найвідоміших реляційних баз даних сучасності та використовуються в різноманітних сферах – торгівля, медицина, виробництво, фінансова та банківська сфери. Причиною цього є те, що дана СКБД є винятково гнучкою та цілісною, порівняно з її реляційними аналогами. Прикладом даних властивостей можна назвати підтримку PostgreSQL як реляційних, так і нереляційних запитів.

PostgreSQL є базою даних з відкритим кодом, тому вона не належить жодній компанії. Її оновленням займається спільнота розробників, яка налічує більше ніж 600 учасників. Такий підхід дозволяє PostgreSQL оновлюватися згідно з викликами сучасного бізнесу та сфер її використання [9].

Перевагами даної СКБД є:

– підтримка JSON;
– підтримка різноманітних мов програмування, таких як C#, Java, C++, Python;
– можливість легкого масштабування;
– можливість розширення шляхом додавання модулів.

У розпорядженні користувачів PostgreSQL безліч типів даних, включаючи JSONB і PostGIS. Крім того, користувачі можуть легко додавати нові типи даних PostgreSQL. Кожен із типів даних призначений для унікальної мети, наприклад для даних для повнотекстового пошуку та для даних дати та часу. При створенні таблиці користувачі спочатку вибирають певний тип даних кожного стовпця. Ці стовпці призначені позначення типів даних, які стосуються полю таблиці [9].

Найчастіше вживаються такі типи даних:

– логічний тип;
– символьний тип;
– дата та час;
– числовий тип.

Логічний тип даних представляє два значення – true та false, також може містити значення NULL. Цей тип даних використовується для виконання умовних операторів. Символьний тип зберігає в собі ряди символів, і використовується для зберігання текстових даних. Числовий тип даних зберігає в собі десяткові цілі числа та числа з плаваючою крапкою. В цей тип входять всім знайомі дані типу integer та float.

Через те, що PostgreSQL є надійною, безпечною базою даних та вона здатна до розширення, а також, маючи велику кількість додаткових засобів для роботи, вона може використовуватися розробниками в різноманітних сценаріях. Дана СКБД створена як кросплатформлена, сумісна з основними ОС сучасноті – Linux, Windows, IOS. Вона підтримує всі основні типи даних, такі як текст, зображення, звуки та відео, а також може працювати з додатковими, дане програмне забезпечення є популярним у різноманітних компаніях, зокрема банківської сфери. PostgreSQL є другою реляційною базою даних за популярністю, програвши лише MySQL[9].

2.6 Висновки до розділу

В даному розділі описуються типи онлайн застосунків, розглянуті приклади їх архітектури з визначенням їх переваг та недоліків, розглянуті мови програмування для розробки веб-додатків та фреймворки для цього. Надано теоретичні відомості про базу даних ТОВ «ФК «Форза».

Проаналізувавши всю інформацію, було прийняте рішення, що створення веб-додатку для роботи з базою даних підприємства буде виконано мовою програмування C# з використанням Entity Framework для архітектури MVC.

3 СИНТЕЗ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Цілі створення КС для ТОВ «ФК «Форза»

Для даного підприємства необхідно мати свою комп'ютерну систему задля виконання онлайн банківських функцій, якими займається компанія. Вона повинна бути реалізована шляхом з'єднання локальних мереж в загальну.

3.2 Розробка вимог до комп'ютерної системи

Виходячи с організаційної структури, комп'ютерна система розподілена на 3 підсистеми:

- підсистеми бухгалтерії;
- підсистеми відділу технічного обслуговування;
- підсистеми відділу роботи з клієнтами.

Дані підсистеми мають реалізовані у вигляді локальних мереж та бути з'єднані між собою як напряму, так і з використанням Інтернет.

Структурна схема комп'ютерної системи ТОВ «ФК «Форза» зображена на рисунку 3.1.

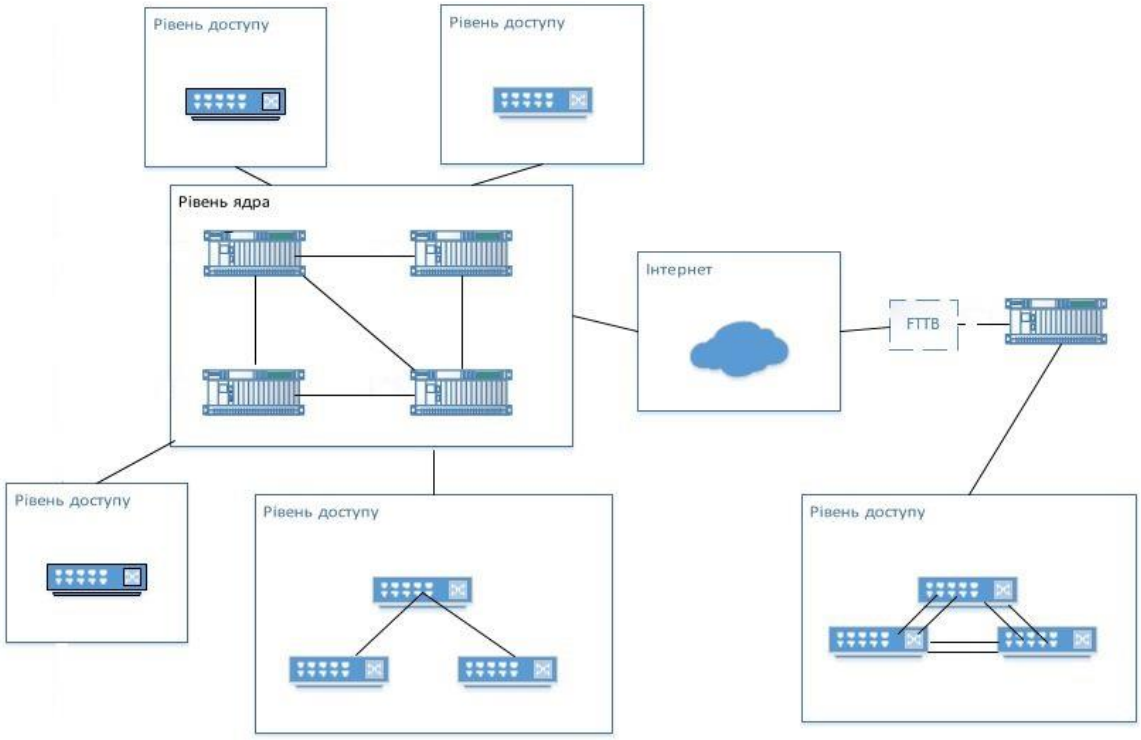

Рисунок 3.1 – Структурна схема КС ТОВ «ФК «Форза»

В цій схемі, роль рівню ядра виконують маршрутизатори, які використовуються для з’єднання філіалів, рівень доступу реалізовано комутаторами для співробітників підприємства.

Система повинна :

- зберігати, обробляти і передавати інформацію між підсистемами;
- мати доступ до Інтернету;
- мати можливість бекапів;
- мати сервери для розміщення бази даних, бекапів та прикладного ПЗ;
- реалізовувати шифроване з’єднання між локальними мережами, що з’єднуються через Інтернет;
- забезпечувати безпеку інформації від спотворення, знищення та незаконного копіювання;
- мати систему аудиту для рівня ядра;
- зберігати, обробляти і передавати інформацію між веб-додатком та користувачем;

– отримувати інформацію з веб додатку і вносити її в базу даних на сервері;
– забезпечити доступ до веб-додатку, розташованого на сервері;
– забезпечити можливість консультувати клієнтів в режимі онлайн через Інтернет;
– забезпечити відділам для роботи з клієнтами можливості зміни даних про клієнтів;
– мати засоби для безперебійного живлення;
– мати запасні одиниці обладнання для заміни у разі виходу з ладу основного.

3.3 Топологічна схема комп'ютерної системи

Топологічна схема комп'ютерної системи ТОВ «ФК «Форза» зображена на рисунку 3.2.

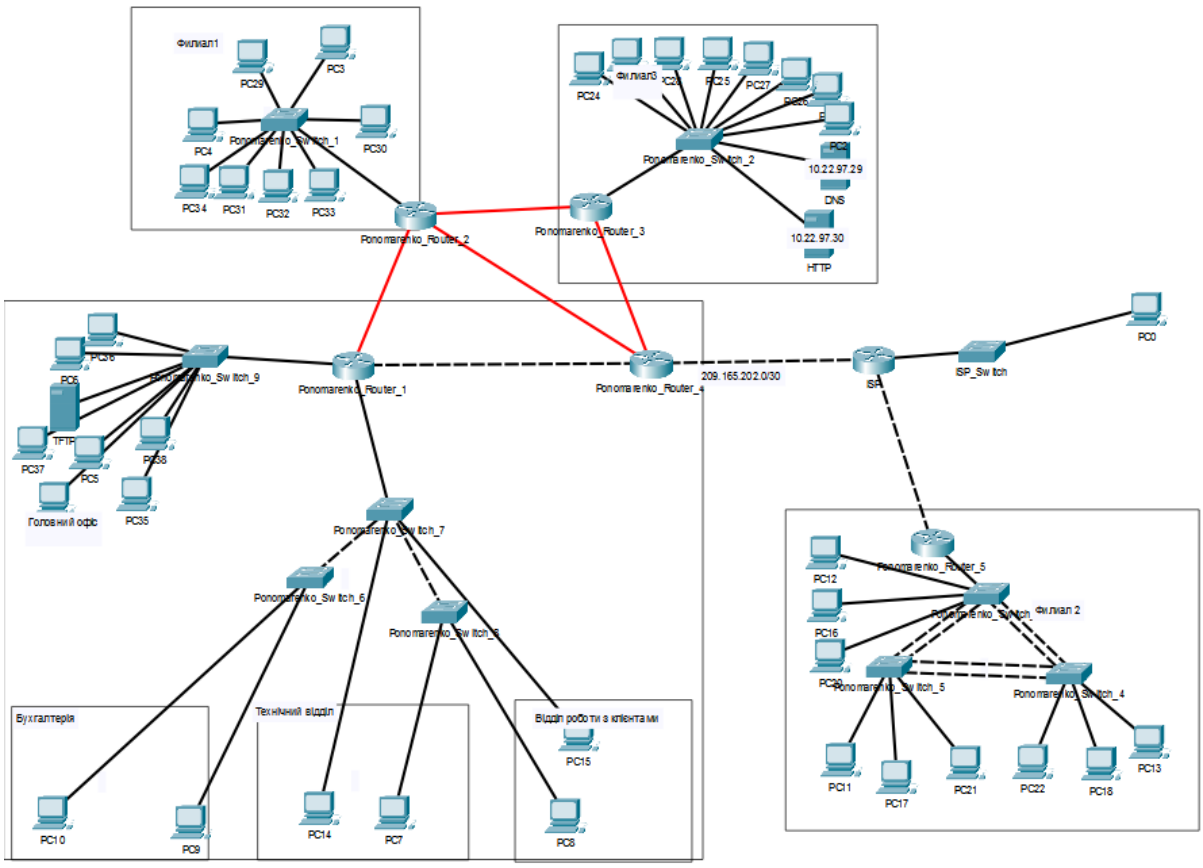

Рисунок 3.2 – Топологічна схема ТОВ «ФК«Форза»

Комп’ютерна мережа реалізована шляхом з’єднання локальних мереж підприємства в одну єдину мережу. В головному офісі знаходяться розташовано дві локальні мереже – бухгалтерська та мережа для обслуговування клієнтів підприємства. Локальні мережі філіалів реалізовані як відділи по роботі з клієнтами. Сервери розташовані в головному офісі. На серверах розміщено базу даних підприємства, створено алгоритми для аутентифікації та аутентифікації користувачів, реалізовано можливість бекапу. Згідно з георозміщенням, всі локальні мережі підприємства знаходяться на віддаленні одне від одного.

3.4 Функціональна схема комп’ютерної системи

Проаналізувавши структурну схему КС ТОВ «ФК «Форза», розроблена функціональна схема зображена на рисунку 3.3.

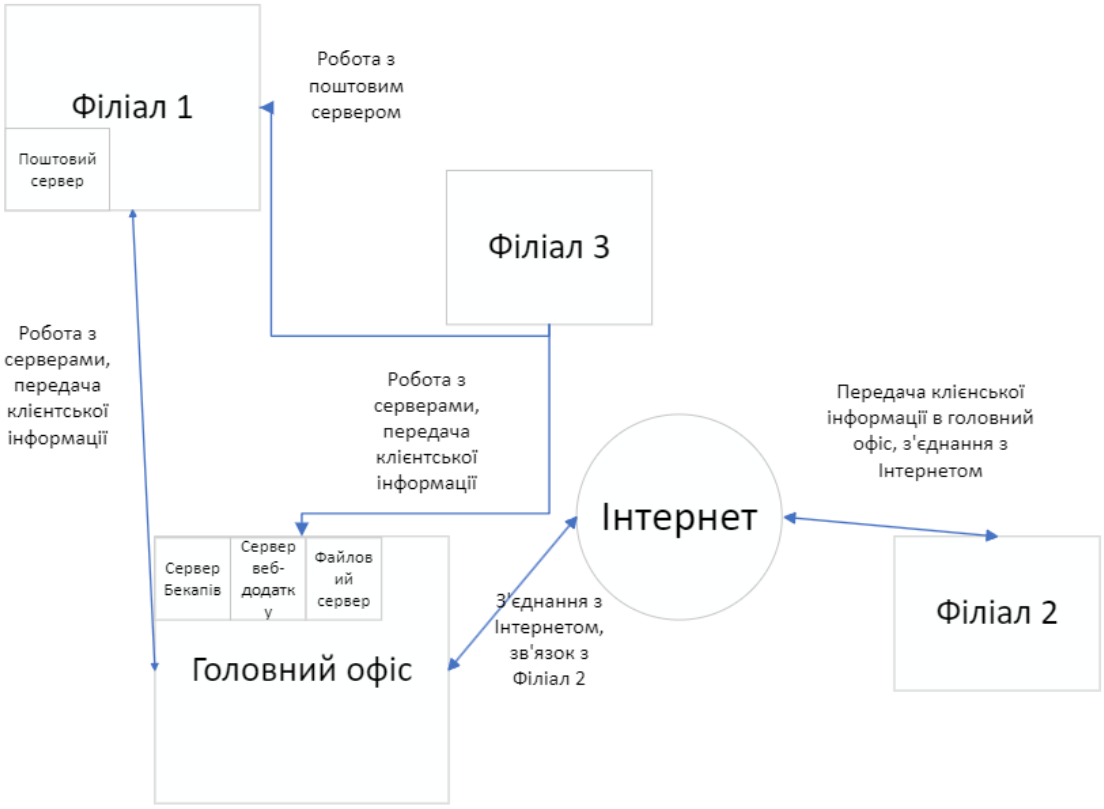

Рисунок 3.3 – Функціональна схема ТОВ «ФК«Форза»

На даній схемі, головний офіс виступає головним функціональним вузлом в системі, через який проходить весь трафік системи, і він підтримує зовнішній зв’язок системи з Інтернетом та філіалом 2. Філіали односторонньо використовують серверне забезпечення, яке розташоване в філіалі 1 та головному офісі. Головний офіс має доступ до всіх підмереж підприємства.

3.5 Аналіз функціоналу та кількості обладнання для системи

Згідно з вимогами, описаними вище, система повинна реалізовувати такі функції:

- мати з’єднання між локальними мережами як напряму, так і через мережу Інтернет;
- виконувати шифрування даних при передачі їх через Інтернет;
- один сервер повинен мати підтримку RAID-масивів для реалізації бекапів;
- підтримка AAA;
- підтримувати цілодобове живлення для серверів;
- мати захист від несанкціонованого входу ззовні.

Проаналізувавши дані функції, а також вимоги до системи, список необхідного обладнання з описом виконуваних функцій та кількістю одиниць наведено в таблиці 3.1.

Таблиця 3.1 – Перелік та функції обладнання

№ п/п	Пристрій	Опис	Кількість
		Вихід в зовнішні мережі	
1	Центральний маршрутизатор	З’єднання з віддаленою мережею та Інтернет, підтримка VPN	1
		Підсистема головного офісу	
2	Сервер бекапів	Підтримка RAID	1

Продовження таблиці 3.1

3	Файловий сервер	Збереження інформації підприємства	1
4	Сервер	Підтримка веб-додатку, підтримка AAA	1
5	Комутатор	З’єднання користувачів в підмережі, використання VLAN	4
6	Робоча станція	Стаціонарний ПК для роботи	30
7	Робоча станція	Портативний ПК для роботи	10
8	Маршрутизатор	Розподілення на підмережі всередині офісу, з’єднання між підмережами	1
		Підсистема філіалу 1	
9	Маршрутизатор	З’єднання між підмережами	1
10	Комутатор	З’єднання з маршрутизатором, між користувачами	1
11	Поштовий сервер	Для листів всередині підприємства	1
12	Робоча станція	Стаціонарний ПК для роботи	20
13	Робоча станція	Портативний ПК для роботи	5
		Підсистема філіалу 2	
14	Шлюз	З’єднання між підмережами через Інтернет	1
15	Комутатор	З’єднання з маршрутизатором, між користувачами	3
16	Робоча станція	Стаціонарний ПК для роботи	20

Продовження таблиці 3.1

		Підсистема філіалу 3	
17	Маршрутизатор	З’єднання між підмережами	1
18	Комутатор	З’єднання з маршрутизатором, між користувачами	1
19	Робоча станція	Стаціонарний ПК для роботи	15

3.6 Мінімальні вимоги до обладнання

Обладнання повинно відповідати таким вимогам :

- маршрутизатори – повинні мати можливість використання оптоволоконного з’єднання, мати інтерфейси Gigabit Ethernet або Fast Ethernet 100Base-T, мати підтримку безпровідної мережі, VPN з’єднань, списків ACL;
- комутатори – підтримка FastEthernet 100Base-T, VLAN, технологій агрегування каналів;
- сервери – процесор Intel Xeon E3-1270 і вище, 8 ГБ ОЗП і більше;
- робочі станції – процесор Intel i3 10-го покоління і вище, 8 ГБ ОЗП і більше.

3.7 Вибір елементної бази

Комп’ютерна система ТОВ «ФК «Форза» використовуватиме мережеве обладнання Cisco через його гнучкість в налаштуванні та забезпечення всіма необхідними функціями.

В ролі маршрутизатора використовується модель Cisco 2911[10]. Дана модель зображена на рисунку 3.4.

Рисунок 3.4 – Маршрутизатор Cisco 2911

Характеристики даного маршрутизатора наведені в таблиці 3.2.

Таблиця 3.2 – Характеристики маршрутизатора Cisco 2911

Кількість ОЗП	512 МБ
Протоколи передачі даних	Ethernet, FastEthernet , GigabitEthernet
Інтерфейси	3 порта Ethernet 10Base-T/100Base-TX/1000Base-T 1 консольний порт
Слоти розширення	4 слоти для HWIC
Розмір накопичувача	256 МБ
Підтримувані стандарти	IEEE 802.1ag, IEEE 802.1ah
ОС	Cisco IOS Security
Параметри безпеки	Підтримує VPN, Firewall, функції фільтрування контенту, ACL
Протоколи маршрутизації	EIGRP
Індикація портів	Цілісність з’єднання, вимкнення, активність
Тип індикатору	Світлодіод

Дана модель маршрутизатора забезпечить систему функціями захисту інформації та зробить можливою передачу інформації між підмережами. Оскільки підмережі філіалу 1, 3 та головного офісу знаходяться в будівлях недалеко, для під'єднання в єдину мережу необхідне оптоволоконне з'єднання. Cisco 2911 має слоти розширення HWIC, для яких існує модуль HWIC-1GE-SFP з використанням GLC-LH-SMD. За допомогою них на даному маршрутизаторі можна використовувати оптоволоконне з'єднання. Також, використовуючи HWIC, реалізовано бездротове підключення до маршрутизатору шляхом додавання модуля HWIC-AP-G-E.

В ролі комутаторів для даної комп'ютерної системи обрана модель Cisco 2960-24TT[11]. Модель зображено на рисунку 3.5.

Рисунок 3.5 – комутатор Cisco 2960-24TT

Характеристики пристрою зображено в таблиці 3.3.

Таблиця 3.3 – Характеристики комутатора Cisco 2960 24TT.

Кількість портів	24 порти Fast Ethernet 2 порти Gigabit Ethernet
Підтримувана мережева технологія	10/100/1000Base-T
Підтримувана Ethernet технологія	Fast Ethernet
Кількість ОЗП	64 МБ
Підтримувана кількість Mac-адрес	До 8 тисяч

Продовження таблиці 3.3

Ємність внутрішнього накопичувача	32 МБ
Підтримка VLAN	Є
Наявність високошвидкісних портів	Є
Підтримка QoS	Є
Кількість можливих VLAN	До 255
Технологія захисту портів	DHCP snooping Link State Tracking
Агрегація каналів	PAgP LACP
Індикація портів	Цілісність з’єднання, вимкнення, активність
Тип індикатору	Світлодіод
Підтримка ACL	Є

Даний комутатор підходить для комп’ютерної мережі ТОВ «ФК «Форза», бо має необхідні системи параметри захисту, такі як ACL, та підтримує VLAN, що дозволить без додаткових витрат відокремити службові підмережі, такі як бухгалтерія і технічний відділ, від загальної.

Для підтримки серверної частини використовується ARTLINE Business R22v01[12]. Пристрій зображено на рисунку 3.6.

Рисунок 3.6 – Сервер ARTLINE Business R22v01

Характеристики серверу наведені в таблиці 3.4.

Таблиця 3.4 – Характеристики серверу ARTLINE Business R22v01.

Процесор	AMD 4-core Ryzen 3 PRO 4350G
Тактова частота процесора	3,8 ГГц
Кількість ядер процесора	4
Материнська плата	PRIME B550M-A з чипсетом AMD B550
Об’єм ОЗП	16 ГБ
Характеристики ОЗП	DDR4-3200 МГц 4 слоти в наявності
Підтримка рівнів RAID	0/1/10
Розмір накопичувача	HDD: 2 x 1 ТБ SSD: 2 x 250 ГБ
Інтерфейс накопичувача	SATA
Блок живлення	Seasonic 400 W 80+ Bronze
Слоти для HDD/SSD	6 шт.
Наявність ОС	Без ОС
Графічний адаптер	Інтегрований
Швидкість мережевого адаптера	1Гбіт/с

За допомогою цієї моделі сервера, в комп’ютерній системі можливо реалізувати функції передачі листів, зберігання інформації та створення бекапів. В цьому допоможе підтримка RAID, для використання якого потрібен контролер Dell Perc H730.

Для робочої станції підприємства у вигляді стаціонарного ПК обрано моноблок Lenovo IdeaCentre 3 271AP7[13]. Формат моноблоку обраний заради економії місця в офісі при обладнанні робочих місць. Моноблок зображений на рисунку 3.7.

Рисунок 3.7 – Моноблок Lenovo IdeaCentre 3 271AP7

Характеристики моноблоку наведені в таблиці 3.5.

Таблиця 3.5 – Характеристики моноблоку Lenovo IdeaCentre 3 271AP7.

Процесор	Intel Core i3-1215U
Тактова частота процесора	3,3 ГГц
Відеокарта	Intel UHD Graphics, інтегрована
Екран	27 дюймів
Максимальне розширення екрану	1920x1080
Тип матриці екрану	IPS
Об’єм ОЗП	8 ГБ
Швидкість LAN Ethernet	1 Гбит/с
Об’єм накопичувача	SSD: 512 ГБ
Модуль Wi-Fi	Наявний
Наявність ОС	Без ОС
Наявність Web-камери	Наявна, зі шторкою

Дана модель персонального комп’ютера дозволить реалізовувати всі необхідні дії при роботі персоналу з клієнтами, підтримує необхідні програми

для роботи з документацією, прикладне програмне забезпечення для роботи бухгалтерії.

Для використання робочої станції у вигляді портативного ПК обрано ноутбук ASUS Vivobook 15 X1500EA-BQ3733. Ноутбук зображено на рисунку 3.8.

Рисунок 3.8 – Ноутбук ASUS VivoBook 15 X1500EA-BQ3352

Характеристики пристрою зображені в таблиці 3.6.

Таблиця 3.6 – Характеристики ноутбуку ASUS VivoBook 15 X1500EA-BQ3352.

Екран	15,6 дюймів
Максимальне розширення	1920x1080
Матриця екрану	VGA
Процесор	Intel Core i3-115G4
Тактова частота процесору	3,0 ГГц
Об’єм ОЗП	12 ГБ
Тип ОЗП	DDR4
Об’єм накопичувача	SSD: 512 ГБ
Відеоадаптер	Intel UHD Graphics, інтегрований

Продовження таблиці 3.6

Модуль Wi-Fi	Наявний
Наявність Web-камери	Наявна, зі шторкою

Даний ноутбук використовується для тестування функцій і роботи прикладного програмного забезпечення компанії, підтримує функціонал роботи з клієнтами та може використовуватись як допоміжний засіб при налаштуванні пристроїв технічним персоналом.

3.8 Кабельне з’єднання

Для з’єднання комутаторів та робочих станцій в системі використовується неекранована вита пара прямого кабелю з конектором RJ-45, яка має 4 пари мідних жил – для підтримки стандарту 100Base-T від фірми Omega. В головному офісі з’єднання між маршрутизаторами відбувається за допомогою екранованої витої пари перехресного кабелю з конектором RJ-45, яка має 8 пар мідних жил – для реалізації стандарту 1000BASE-T від фірми Omega. З’єднання між головним офісом системи і філіалами 1 та 3 відбувається за допомогою оптоволоконного з’єднання від фірми Finmark. Для з’єднання головного офісу і філіалу 1 необхідно 400 метрів оптоволоконного кабелю, для філіалу 3 та головного офісу – 300 метрів, для філіалу 1 та 3 – 500 метрів. Для з’єднання комутаторів з маршрутизаторами необхідно 10 метрів екранованої витої пари з 8 жилами, для з’єднання комутаторів з серверами та робочими станціями необхідно 25 метрів неекранованої витої пари з 4 жилами. Для з’єднання маршрутизаторів в головному офісі використовується 10 метрів перехресного кабелю екранованої витої пари з 8 жилами. Для з’єднання комутаторів між собою використовується 10 метрів перехресного кабелю екранованої витої пари з 8 жилами. Кабелі до робочих станцій та до комутаторів йдуть вздовж стін будівлі знаходження, розміщені в кабель каналах. Для кабелів живлення

використовуються кабелі з конектором IEC-C13 для робочих станцій та IEC-C14 для комутаторів та маршрутизаторів для 220 В мережі.

3.9 Комплектація

Для комплектації комп'ютерної системи обрано обладнання, розглянуте в пункті 3.7 та кабельне з'єднання, розглянуте в пункті 3.8.

3.9.1 Комплект головного офісу

Для реалізації функцій та потреб головного офісу необхідно:

- сервер ARTLINE Business R22v01 в кількості 3 штуки;
- маршрутизатор Cisco 2911 – 2 штуки;
- моноблок Lenovo IdeaCentre 3 271AP7 – 30 штук;
- ноутбук ASUS VivoBook 15 X1500EA-BQ3352 – 10 штук;
- комутатор Cisco 2960-24TT – 4 штуки;
- модулі HWIC-1GE-SFP та GLC-LH-SMD – 1 штука кожного;
- модуль HWIC-AP-G-E – 1 штука;
- екранована вита пара з 8 жилами – 50 метрів;
- неекранована вита пара з 4 жилами – 750 метрів;
- конектори RJ-45 – 70 штук.

3.9.2 Комплект філіалу 1

Для реалізації функцій та потреб філіалу 1 необхідно:

- сервер ARTLINE Business R22v01 в кількості 1 штуки;
- маршрутизатор Cisco 2911 – 1 штуки;
- моноблок Lenovo IdeaCentre 3 271AP7 – 20 штук;
- ноутбук ASUS VivoBook 15 X1500EA-BQ3352 – 5 штук;
- комутатор Cisco 2960-24TT – 1 штуки;
- модулі HWIC-1GE-SFP та GLC-LH-SMD – 3 штуки кожного;
- модуль HWIC-AP-G-E – 1 штука;

- екранована вита пара з 8 жилами – 10 метрів;
- неекранована вита пара з 4 жилами – 500 метрів;
- конектори RJ-45 – 42 штуки.

3.9.3 Комплект філіалу 2

Для реалізації функцій та потреб філіалу 2 необхідно:

- маршрутизатор Cisco 2911 – 1 штуки;
- моноблок Lenovo IdeaCentre 3 271AP7 – 20 штук;
- комутатор Cisco 2960-24TT – 3 штуки.
- модуль HWIC-AP-G-E – 1 штука;
- екранована вита пара з 8 жилами – 70 метрів;
- неекранована вита пара з 4 жилами – 500 метрів;
- конектори RJ-45 – 54 штуки.

3.9.4 Комплект філіалу 3

Для реалізації функцій та потреб філіалу 3 необхідно:

- маршрутизатор Cisco 2911 – 1 штуки;
- моноблок Lenovo IdeaCentre 3 271AP7 – 15 штук;
- комутатор Cisco 2960-24TT – 1 штуки;
- модулі HWIC-1GE-SFP та GLC-LH-SMD – 3 штуки кожного
- модуль HWIC-AP-G-E – 1 штука;
- екранована вита пара з 8 жилами – 10 метрів;
- неекранована вита пара з 4 жилами – 375 метрів;
- конектори RJ-45 – 32 штуки.

3.10 Висновок

В даному розділі були розроблені вимоги до комп’ютерної системи ТОВ «ФК «Форза», проаналізовані її функції, підібране базове обладнання, з

обґрунтуванням його вибору, та створено комплектацію для підмереж підприємства на основі підібраного обладнання.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Призначення і область використання застосунку

Метою розробки програмного забезпечення для ТОВ «ФК «Форза» є відображення та редагування даних користувачів в веб-інтерфейсі.

Програмне забезпечення реалізовано у вигляді веб-додатку з розподілом ролей користувачів, для адміністративних ролей – доступ до клієнтських даних, для клієнтів – доступ до свого профілю.

Областю використання додатку є адміністрування облікових записів.

4.2 Обґрунтування технічних характеристик програми

4.2.1 Постановка завдання на розробку програми

Необхідно розробити веб-додаток, який має можливості аутентифікації та авторизації з можливостю редагування даних профілів.

Додаток працює в режимі реального часу і використовує базу даних підприємства ТОВ «ФК «Форза».

Розробити інтерфейс веб-сторінок для відображення інформації про користувачів.

Програма повинна підтримуватись на більшості браузерів.

4.2.2 Опис алгоритму функціонування програми

Веб-додаток має п’ять видів HTML-сторінок, які виконують такі функції:

- відображує головну сторінку додатку;
- відкриває форму для реєстрації;
- відкриває форму для авторизації;
- виводить інформації про користувача;
- виводить список користувачів.

При запуску роботи програми користувач потрапляє на головну сторінку додатку. В залежності від того, авторизований користувач чи ні, головна

сторінка має різні варіації кнопок. При відсутності авторизованого користувача, на сторінці зображено кнопки для реєстрації та входу в додаток. При натисканні на них, викликається форма для відповідної дії. Алгоритм роботи форми для реєстрації зображено на рисунку 4.1.

Рисунок 4.1 – Алгоритм роботи форми реєстрації

Алгоритм роботи форми для авторизації в додатку зображено на рисунку 4.2.

Рисунок 4.2 – Алгоритм роботи форми авторизації

Авторизовані користувачі переносяться на головну сторінку та з неї мають доступ до функціоналу, передбаченого ролями:

– Admin-користувачі мають можливість переглядати весь список користувачів-клієнтів додатку та редагувати та видаляти профілі за потреби;
– Користувачі-клієнти мають доступ до редагування та перегляду власного профілю.

Алгоритм роботи перегляду власного профілю користувача зображено на рисунку 4.3.

Рисунок 4.3 – Алгоритм роботи перегляду власного профілю користувача

Алгоритм роботи перегляду списку профілів користувачів зображено на рисунку 4.4.

Рисунок 4.4 – Алгоритм роботи перегляду списку профілів користувачів

4.3 Опис і обґрунтування вибору методу організації вхідних та вихідних даних

Для додатку вхідними даними являються виклики користувача з веб-інтерфейсу програми. Вихідними даними є оброблена серверною частиною інформація з виклику користувача та відображення оновленого веб-інтерфейсу програми, відповідно до виконаних операцій.

4.4 Опис і обґрунтування вибору та складу технічних та програмних засобів

У якості технічних складових для підтримки роботи програми необхідно використовувати сервер, який підтримує ОС Windows.

Архітектурою веб-додатку є модель-контролер-вид, що означає розподілення програми на клієнтську та серверну частину. Дане розділення дозволить легше масштабувати та тестувати додаток.

Серверна частина програми реалізована за допомогою мови програмування C# з використанням платформи ASP.NET та Entity Framework Core. Мова програмування C# з використанням платформи ASP.NET широко використовується при розробці веб-додатків. Популярність використання спричинена тим, що розроблені додатки мають всі необхідні властивості для бізнесу, а саме:

- робота в реальному часі;
- використовування різноманітні веб API;
- підтримка кросплатформленість;
- забезпечення безпеки даних;
- підтримування різноманітних СКБД;
- робота додатку на великій швидкості;
- підтримка асинхронності.

Серверна частина програми працює з СКБД PostgreSQL. Для роботи з даною базою даних використовується Entity Framework Core з використанням провайдеру Npgsql.PostgreSQL. Для виконання запитів використовується LINQ to Entities – мова запитів до джерела даних.

Клієнтська частина реалізована за допомогою Razor Pages – HTML-сторінок, які підтримують C#, з використанням CSS-класів Bootstrap. На даних сторінках для зв’язку з серверною частиною додатку використовуються HTTP-форми з методом POST.

4.5 Опис розробленої програми

4.5.1 Загальні відомості та функціональне призначення

Програма реалізована веб-додатком з архітектурою MVC та призначена для перегляду та редагування даних профілів користувачів, з винесенням інформації про це у браузер.

4.5.2 Опис логічної структури програми

Програма складається з серверної та клієнтської частини. Серверна частина з'єднана с базою даних PostgreSQL. База даних зображена на рисунку 4.5.

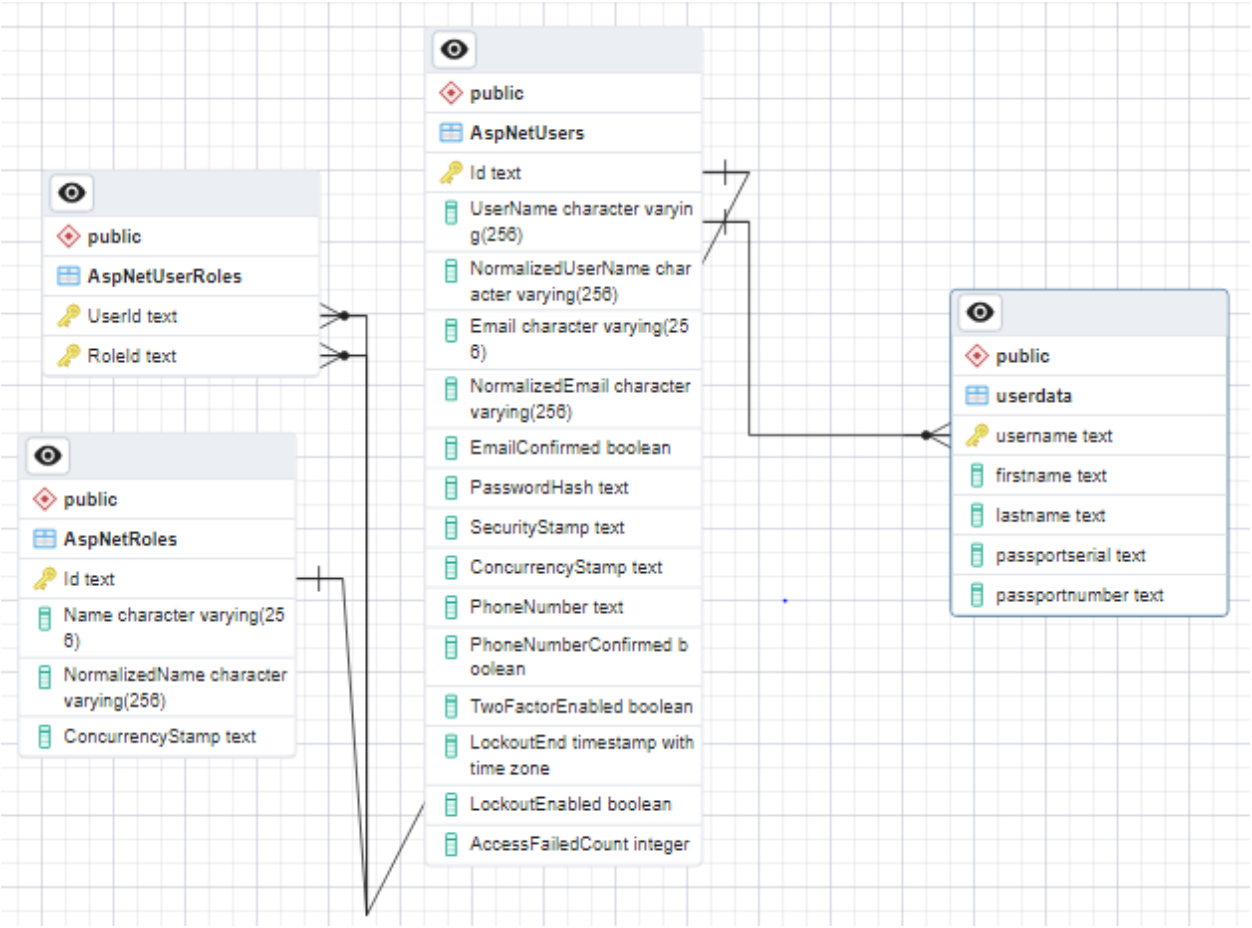

Рисунок 4.5 – База даних ТОВ «ФК «Форза»

Інформація про використовувані поля таблиці AspNetUser зображено в таблиці 4.1.

Таблиця 4.1 – Таблиця AspNetUser

Назва поля	Тип даних
Id	text
UserName	Character varying[256]
PasswordHash	text

Інформація про таблицю AspNetRoles зображено в таблиці 4.2.

Таблиця 4.2 – Таблиця AspNetRoles

Назва поля	Тип даних
Id	text
Name	Character varying[256]
NormalizedName	Character varying[256]

Інформація про таблицю AspNetUserRoles зображено в таблиці 4.3.

Таблиця 4.3 – Таблиця AspNetUserRoles

Назва поля	Тип даних
UserId	text
RoleId	text

Інформація про таблицю userdata зображено в таблиці 4.4.

Таблиця 4.4 – Таблиця AspNetUserRoles

Назва поля	Тип даних
username	text
firstname	text
lastname	text
passportserial	text
passportnumber	text

З даної бази даних створити модель необхідно тільки для таблиці userdata, бо таблиці ASP згенеровані за допомогою ASP.NET Identity. Створена модель для таблиці виглядає так:

```
public partial class Userdatum
    { public string Username { get; set; } = null!;
        public string? Creditcard { get; set; }
        public string? Firstname { get; set; }
        public string? Lastname { get; set; }
```

```
public string? Passportserial { get; set; }
public string? Passportnumber { get; set; } }
```

Розроблені моделі для авторизації та реєстрації зображені на рисунку 4.6.

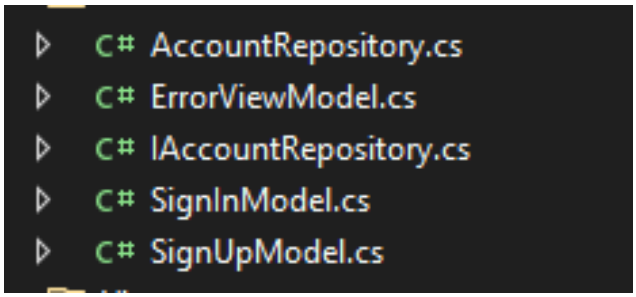

Рисунок 4.6 – Моделі для авторизації та реєстрації

Зв’язок між відображеннями та моделлю забезпечує контролер, який реагує на HTTP-виклики з клієнтської сторони.

Клієнтська сторона виконана в Razor Pages. Для кожного типу користувача є різна навігація по додатку. Навігація для неавторизованого користувача зображена на рисунку 4.7.

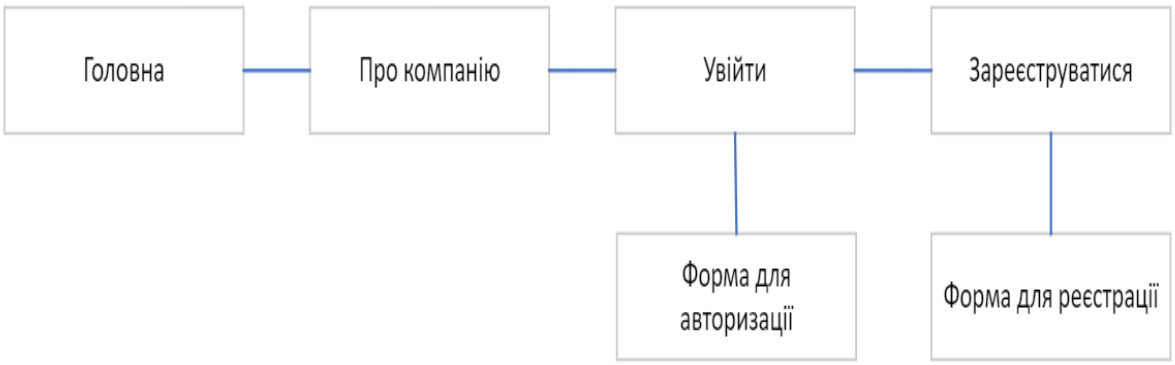

Рисунок 4.7 – Навігація для неавторизованого користувача

Навігація для користувача клієнта зображена на рисунку 4.8.

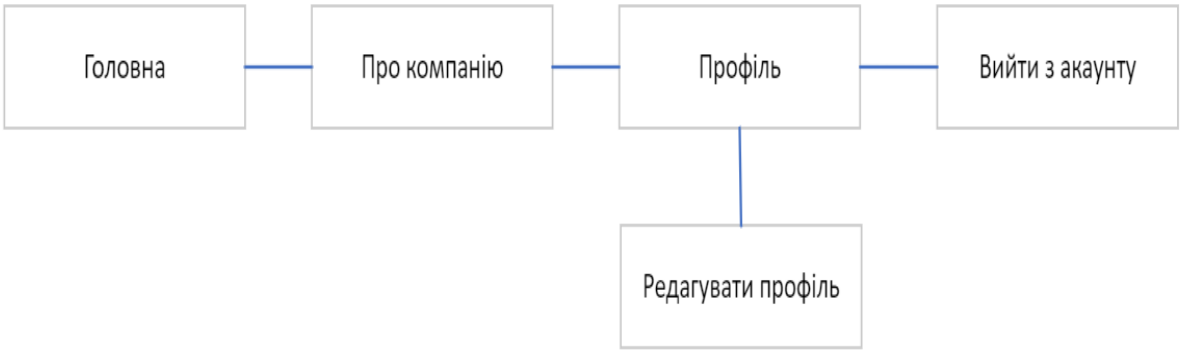

Рисунок 4.8 – Навігація для користувача клієнта

Навігація для користувача адміністратора зображена на рисунку 4.9.

Рисунок 4.9 – Навігація для користувача адміністратора

Клієнтська сторона передає HTTP запити до контролерів, яких в даному додатку розроблено два – AccountController та HomeController. AccountController відповідає за роботу реєстрації, входу та відображення користувацької інформації, HomeController – за неавторизована використання додатку.

Razor Pages, які використовуються AccountController зображено на рисунку 4.10.

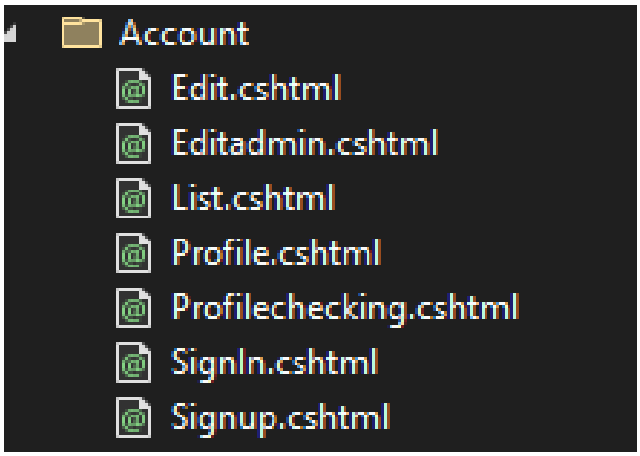

Рисунок 4.10 – Сторінки, запитами яких керує AccountController

Razor Pages, які використовуються HomeController зображено на рисунку 4.11.

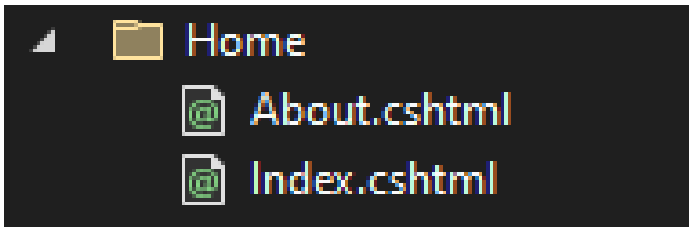

Рисунок 4.11 – Сторінки, запитами яких керує HomeController

Всі сторінки, крім сторінок реєстрації та входу, використовують загальний шаблон розмітки HTML-сторінки, який в проекті називається _Layout. Логічна частина для зміни кнопок реалізована в _Login шляхом наявності авторизованого користувача та перевірки його ролі.

4.5.3 Використані технічні засоби

Для роботи веб-додатку необхідно виділити окремий сервер з ОС Windows. Також необхідно забезпечити стабільне Інтернет-з’єднання завдяки надійній комп’ютерній мережі.

4.5.4 Виклик і завантаження програми

Програма викликається за допомогою зовнішньої адреси серверу за допомогою браузеру і працює в режимі реального часу з використанням

асинхронних операцій. Під час завантаження програми запускається сторінка веб-інтерфейсу для неавторизованого користувача.

4.6 Очікувані технічно-економічні показники

Розроблений веб-додаток за допомогою мови програмування C# та платформи ASP.NET з фреймворком Entity Framework Core дозволить підприємству ТОВ «ФК «Форза» успішніше керувати даними користувачів, покращить користувацький досвід, що позитивно відобразиться на успіхах компанії.

4.7 Висновок

В ході розробки програмного забезпечення було спроектовано веб-додаток з архітектурою MVC мовою C# з використанням платформи ASP.NET та фреймворку Entity Framework Core та створенням веб-інтерфейсів за допомогою Razor Pages.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Завдання та мета експерименту

Завданням експерименту є перевірка працездатності створеного веб-додатку. Метою є перевірка коректності роботи додатку при некоректно та правильно заповнених формах для реєстрації та авторизації, відображення списку користувачів та користувацьких профілів.

5.2. Сутність експерименту

В ході експерименту, для перевірки роботи форм, будуть створюватися коректні та некоректні моделі для них з відображенням внесення результату виконання операції в базу даних, викликаними помилками про вводі некоректних даних. Для перевірки функцій відображення профілів користувачів та користувацьких профілів буде використано авторизованих користувачів з адміністративними та клієнтськими ролями.

5.3 Хід експерименту

Для початку експерименту необхідно запустити розроблений веб-додаток. Для тестування він розташований по адресі localhost з портом 7190. Дана адреса прописана в конфігураційних файлах додатку, тому при зміні порту вручну на сторінці браузера відобразиться помилка скидання підключення, так як порт може використовуватися іншим процесом системи. Успішний запуск веб-додатку зображено на рисунку 5.1.

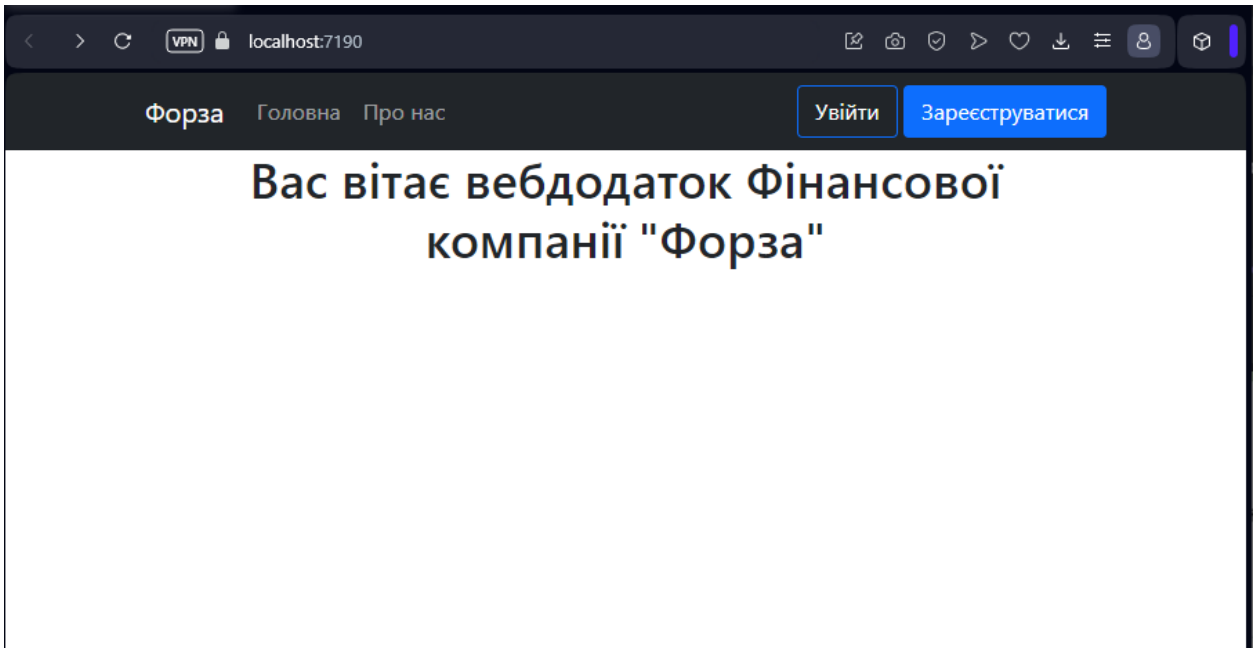

Рисунок 5.1 – Успішний запуск веб-додатку

Форми, які підлягають тестуванню, знаходяться під посиланнями на кнопках «Увійти» та «Зареєструватися». Форма для реєстрації зображена на рисунку 5.2.

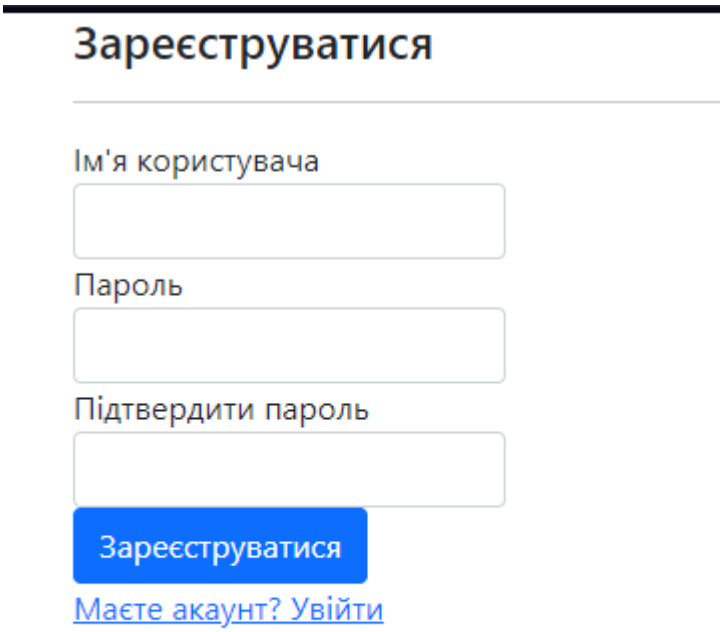

Рисунок 5.2 – Форма реєстрації

Дана форма має такі обмеження:

– ім’я користувача повинно бути не менше 5 символів латиницею;
– пароль повинен складатися з латинських символів, мінімум одної великої та малої літери, мати мінімум одну цифру та один спеціальний знак, і бути не менше, ніж 5 символів;
– поля «Пароль» та «Підтвердити пароль» повинні співпадати.

Варіант введення некоректного паролю зображено на рисунку 5.3.

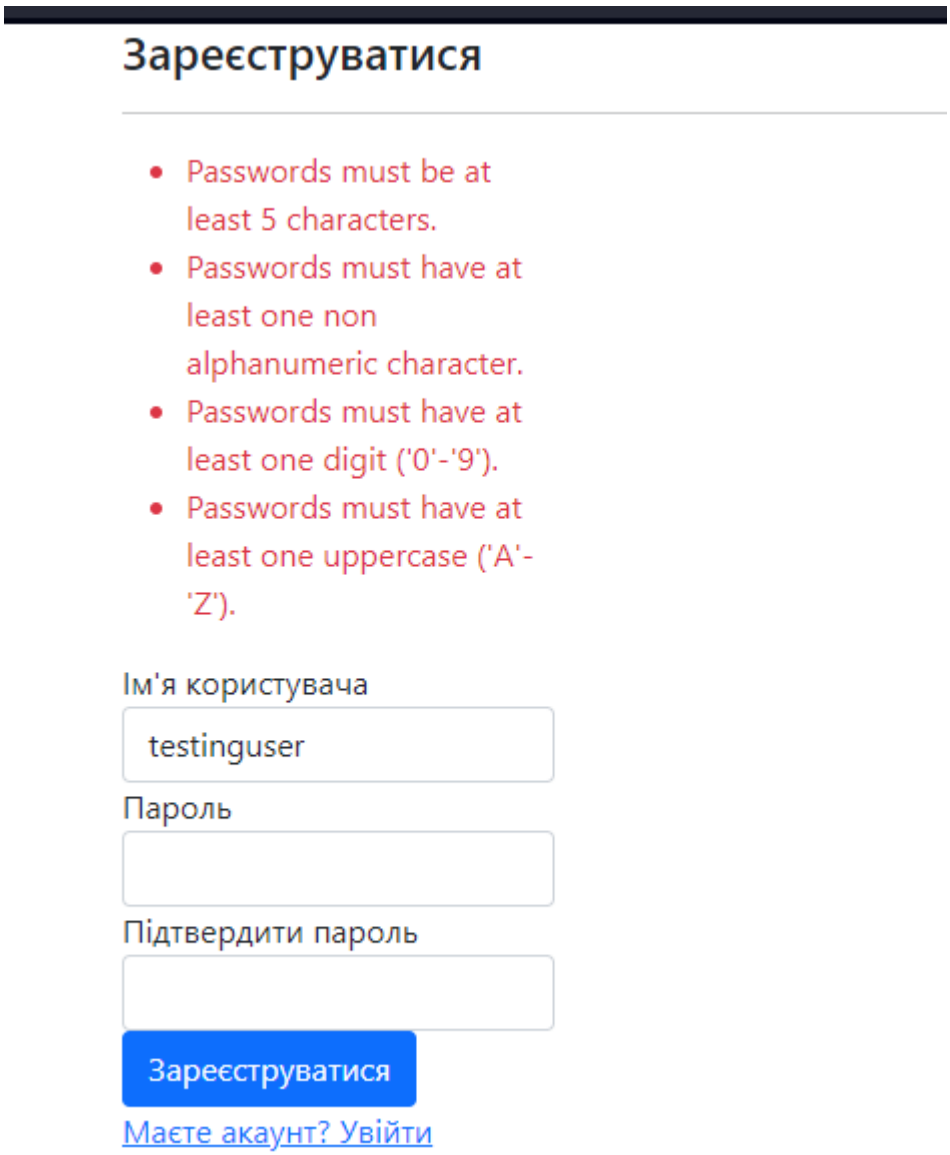

Рисунок 5.3 – Помилка при введені некоректного паролю

Результат роботи програми при не співпадінні полів «Пароль» та «Підтвердити пароль» зображено на рисунку 5.4.

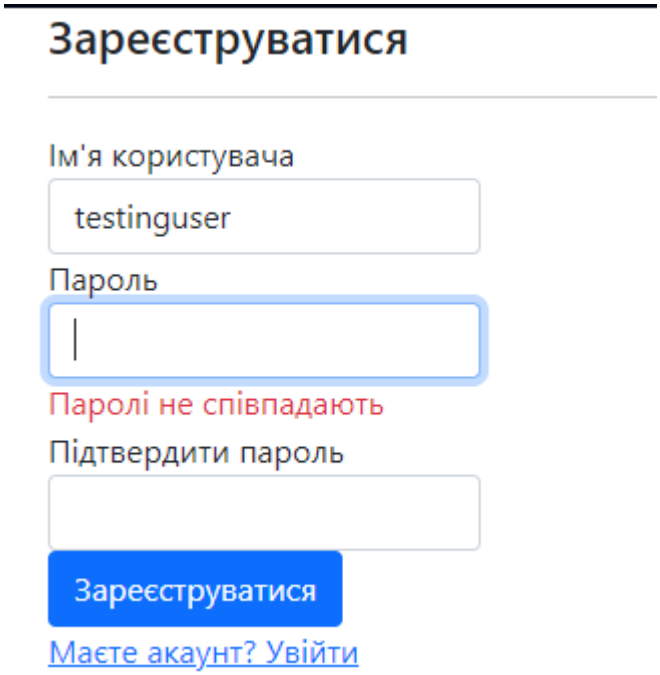

Рисунок 5.4 – Помилка при різних даних в полях паролю

В разі успішної реєстрації, форма адресує користувача на головну сторінку та вносить запис в базу даних. Відображення нового створеного користувача з іменем testinguser зображено на рисунку 5.5.

	Id [PK] text	UserName character varying (256)	NormalizedUserName character varying (256)
1	0185d229-3d8b-4dd5-a118-d6268e336e71	Ponomarenkouser	PONOMARENKOUSER
2	74a3ad26-4391-45da-ad2b-857c60de3c27	testinguser	TESTINGUSER
3	85880f4b-381b-43a7-b53c-43836bdc70ee	testuser	TESTUSER
4	8a4c9d05-1476-40c2-8801-6d78e1e21ca0	applicationuser	APPLICATIONUSER

Рисунок 5.5 – Внесений користувач в базу даних

Форма для авторизації зображена на рисунку 5.6.

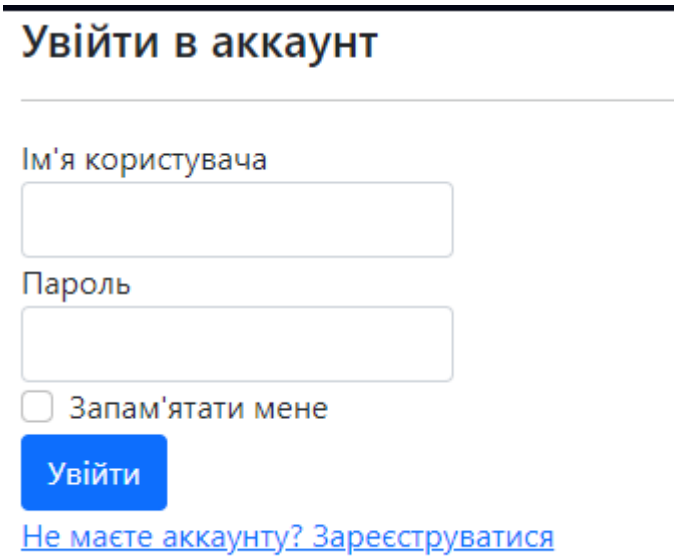

Рисунок 5.6 – Форма авторизації

Дана форма звіряє введені дані в неї з даними з бази даних користувачі. В разі співпадіння переадресує користувача на головну з відображенням входу в акаунт. Для цієї форми є тільки одна помилка – при не знаходженні або не співпадінні облікових даних веб-додаток виводить повідомлення про неправильно введене ім’я користувача або пароль. Ця помилка зображена на рисунку 5.7.

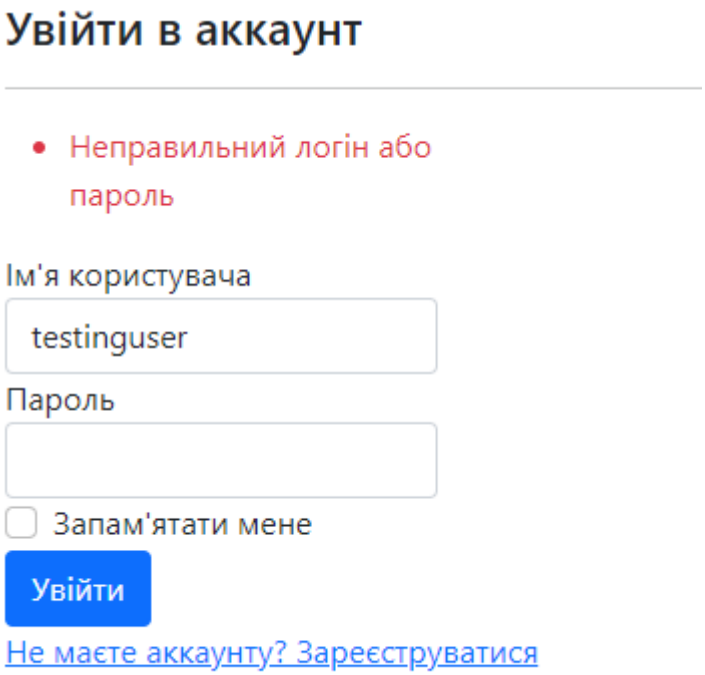

Рисунок 5.7 – Помилка при авторизації

Успішна спроба входу зображена на рисунку 5.8.

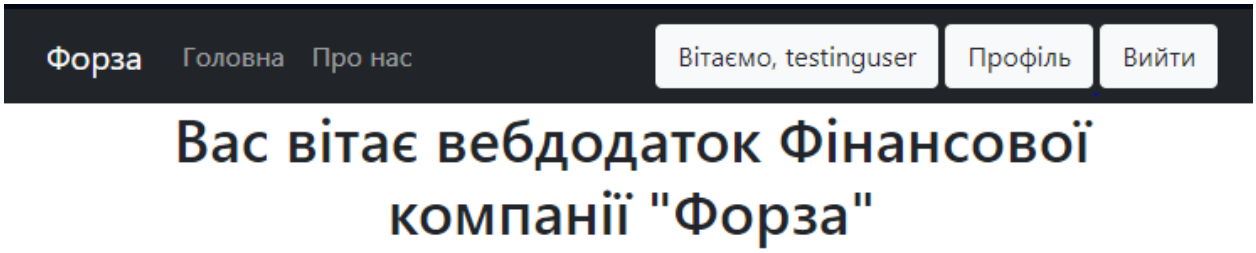

Рисунок 5.8 – Успішний вхід

Всі користувачі мають клієнтську роль за замовчуванням, адміністративна роль додається тільки за допомогою прямого запиту в базу даних. Клієнтські користувачі мають доступ до свого профілю з можливістю редагування. Приклад профілю користувача зображено на рисунку 5.9

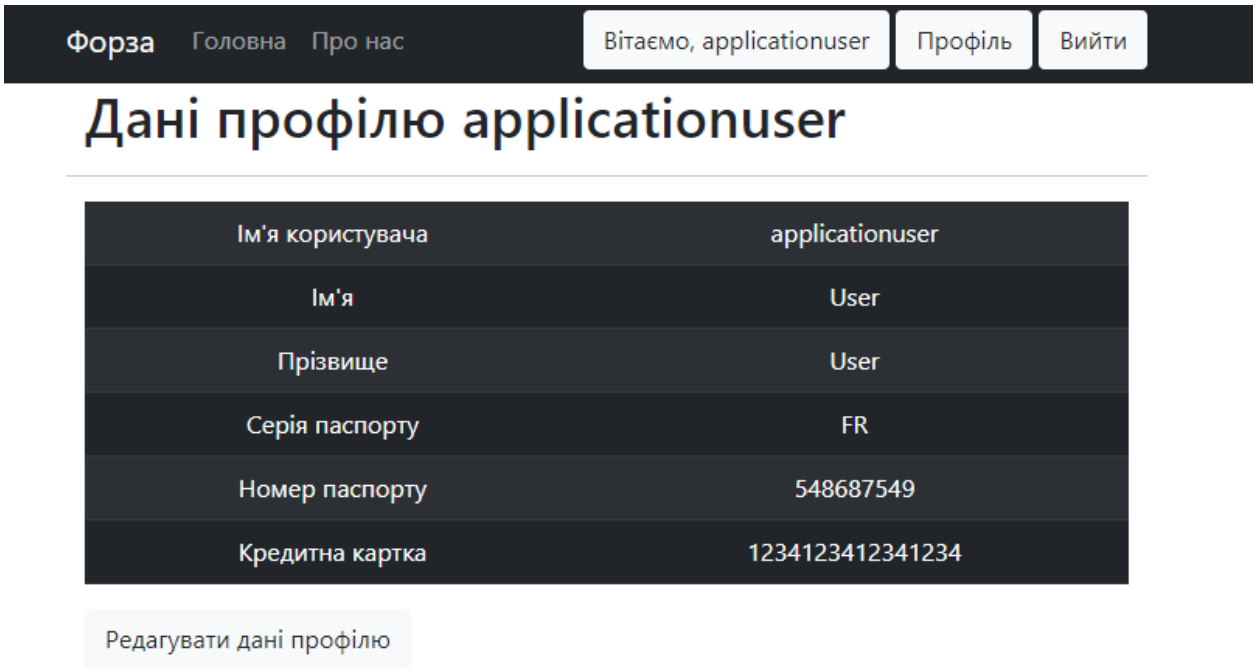

Рисунок 5.9 – Дані профілю applicationuser

На рисунку 5.10 відображено форма для редагування даних користувача.

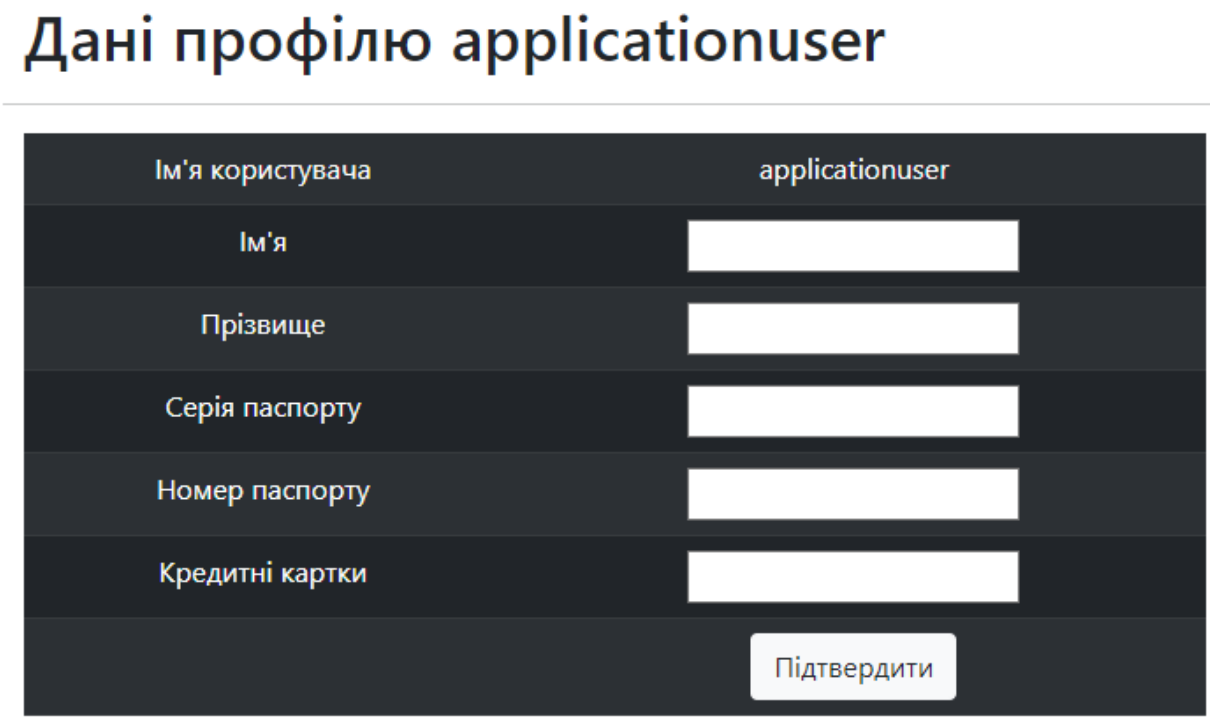

Рисунок 5.10 – Форма редагування профілю

На рисунку 5.11 відображено оновлений профіль користувача.

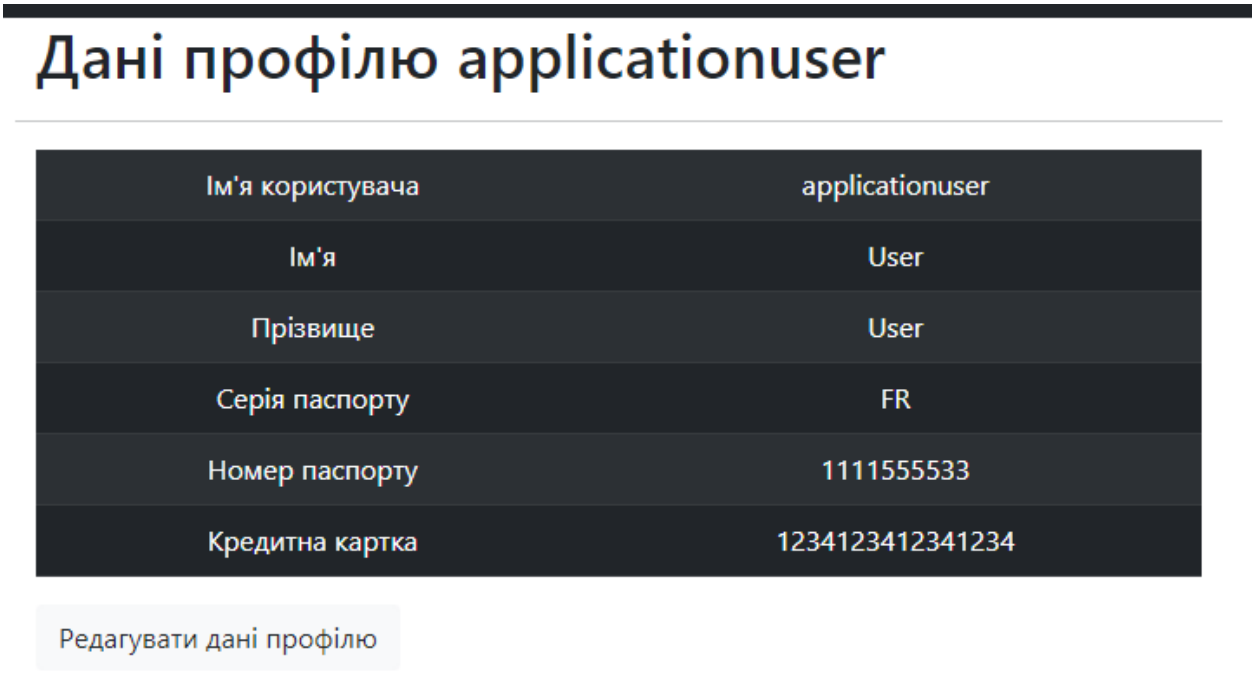

Рисунок 5.11 – Оновлені дані профілю

Відображення відповідних змін в базі даних відображено на рисунку 5.12.

	username [PK] text	creditcard text	firstname text	lastname text	passportserial text	passportnumber text
1	applicationuser	1234123412341234	User	User	FR	1111555533

Рисунок 5.12 – Зміни в базі даних

Для відображення списку користувачів використовується користувач testuser. Відображення списку користувачів зображено на рисунку 5.13.

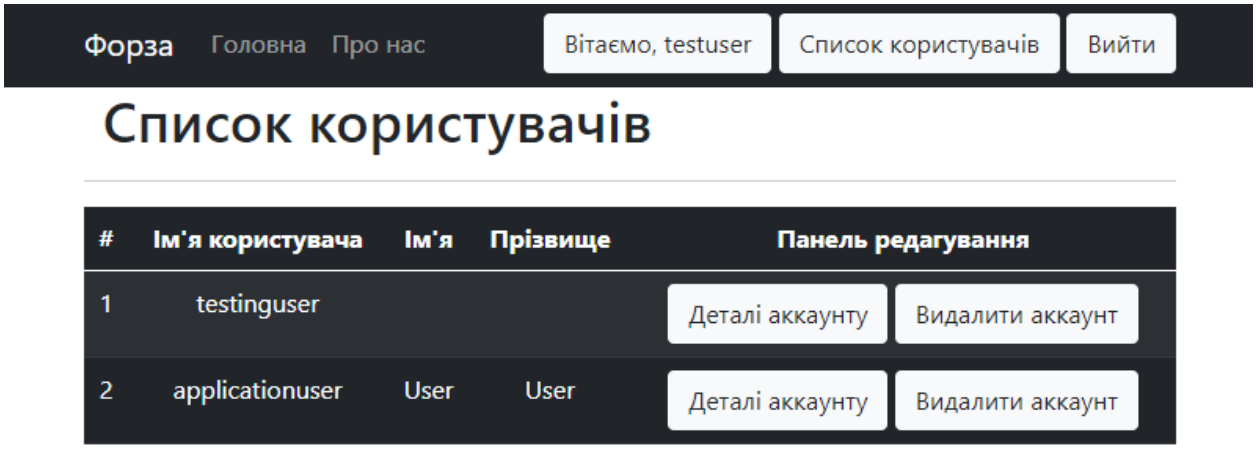

Рисунок 5.13– Список користувачів

5.4 Висновок

В ході експерименту було підтверджено функціонування програми, протестовано форми на реагування на помилки та внесення інформації в базу даних в разі успіху, зчитування користувацьких даних з бази даних для авторизації, перевірено роботу системи профілів для адміністративних та клієнтських ролей користувачів

ВИСНОВОК

Дана кваліфікаційна робота є завершеною науковою роботою, в якій реалізована науково-практична задача – створено веб-додаток для роботи з базою даних товариства з обмеженою відповідальністю «Фінансова компанія «Форза» з обґрунтуванням параметрів комп’ютерної системи підприємства для реалізації його роботи.

Основні висновки і результати роботи полягають у наступному:

1. Для кращого аналізу під час вибору обладнання для комп’ютерної систему, було розглянуто сферу діяльності підприємства.
2. Проаналізовано архітектури веб-додатків, мови їх написання, використовувані фреймворки для роботи з базами даних та обрано програмно-технічні засоби для розробки веб-додатку.
3. Розроблено комп’ютерну систему компанії, яка реалізована шляхом локальних обчислювальних мереж, і є відмовостійкою, безпечною, швидкодіючою та реалізує можливість виділення ресурсів на серверах для розміщення веб-додатку.
4. Розроблено веб-додаток з архітектурою MVC на мові C# з використанням фреймворку Entity Framework Core для роботи з базою даних PostgreSQL.
5. Проведені експериментальні тестування програми на працездатність, підтримку коректних відображень помилок при хибній валідації даних на формах та занесенням нових даних в базу даних, відображенням інформації про профілі шляхом запитів до баз даних за рахунок Entity Framework.

ПЕРЕЛІК ПОСИЛАНЬ

1. Цифровізація банківської системи [Електронний ресурс]: https://denzadnem.com.ua/aktualno/150685
2. Пономаренко, А.Ю. Комп’ютерна система ТОВ «Фінансова компанія «Форза» з детальним опрацюванням побудови, налаштування та безпеки корпоративної мережі / А. Ю. Пономаренко ; Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2022. – 13 с
3. Веб-форми ASP.NET [Електронный ресурс] : https://learn.microsoft.com/en-us/aspnet/web-forms/
4. Мова програмування C# [Електронный ресурс]: https://learn.microsoft.com/en-us/dotnet/csharp/
5. Платформа ASP.NET [Електронный ресурс] : https://learn.microsoft.com/en-us/aspnet/overview
6. Мова програмування Java [Електронный ресурс] : https://www.java.com/en/download/help/whatis_java.html
7. Фреймворк Entity Framework Core [Електронный ресурс] : https://learn.microsoft.com/en-us/ef/core/
8. Фреймворк Spring [Електронный ресурс] : https://docs.spring.io/spring-framework/docs/5.0.0.M1/spring-framework-reference/html/overview.html
9. PostgreSQL [Електронный ресурс] : https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-postgresql
10. Маршрутизатор Cisco 2911 [Електронний ресурс]: https://stack-systems.com.ua/marshrutizator-cisco-2911-k9
11. Комутатор Cisco 2960-24TT [Електронний ресурс]: https://stack-systems.com.ua/kommutator-cisco-ws-c2960-24tt-l
12. Сервер ARTLINE Business R22v01 [Електронний ресурс]: https://rozetka.com.ua/ua/artline_r22v01/p238592581/

13.Моноблок Lenovo IdeaCentre 3 271AP7 [Електронний ресурс]: https://hard.rozetka.com.ua/ua/lenovo-f0gj00q7uo/p392944584/

14. Ноутбук ASUS VivoBook 15 X1500EA-BQ3352 [Електронний ресурс]: https://rozetka.com.ua/ua/asus-90nb0ty6-m040w0/p379852182/

15.Атестація здобувачів вищої освіти. Методичні рекомендації до виконання кваліфікаційної роботи магістра студентами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, С.М. Ткаченко, В.В. Гнатушенко. – Д.: НТУ «ДП», 2022. – 57 с.

ДОДАТОК А

Текст програми веб-додатку

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
РОЗРОБКА ВЕБ-ДОДАТКУ ASP.NET MVC

Текст програми

804.02070743.23019-01 12 01

Листів 26

2023

АНОТАЦІЯ

Дана програма містить в собі програмний код моделей, інтерфейсів та контролерів, необхідних для функціонування програмного забезпечення.

ЗМІСТ

1 МОДЕЛІ

1.1 Таблиця бази даних

```
public partial class Userdatum
  {
    public string Username { get; set; } = null!;
    public string? Creditcard { get; set; }
    public string? Firstname { get; set; }
    public string? Lastname { get; set; }
    public string? Passportserial { get; set; }
    public string? Passportnumber { get; set; }
  }
```

1.2 Форма реєстрації

```
public class SignUpModel
  {
    [Required(ErrorMessage ="Введіть ім'я користувача")]
    [StringLength(maximumLength:16,MinimumLength =5,ErrorMessage ="Ім'я користувача повинно бути не менше 5 символів")]

    [Display(Name ="Ім'я користувача")]
    public string Username { get; set; }
    [Required(ErrorMessage = "Введіть пароль")]
    [Compare("ConfirmPassword",ErrorMessage ="Паролі не співпадають")]

    [DataType(DataType.Password)]
    public string Password { get; set; }
    [Required(ErrorMessage ="Підтвердіть пароль")]
    [DataType(DataType.Password)]
    public string ConfirmPassword {get; set; }
   }
```

1.3 Форма входу

```
public class SignInModel
  {
    [Required]
    public string Username { get; set; }
    [Required]
    [DataType(DataType.Password)]
```

```
    public string Password { get; set; }
    public bool RememberMe { get; set; }
  }
```

1.4 Робота з акаунтами

```
public class AccountRepository : IAccountRepository
  {
    private readonly UserManager<IdentityUser> _usermanager;
    private readonly SignInManager<IdentityUser> _signInManager;

    public AccountRepository(UserManager<IdentityUser> userManager,SignInManager<IdentityUser>
signInManager)
    {
      _usermanager = userManager;
      _signInManager = signInManager;
    }
    public async Task<IdentityResult> CreateUserAsync(SignUpModel userModel)
    {
      var user = new IdentityUser()
      {
        UserName = userModel.Username

      };
      var result= await _usermanager.CreateAsync(user,userModel.Password);
      return result;

    }
    public async Task<SignInResult> PasswordSignInAsync(SignInModel signInModel)
    {
      var result = await _signInManager.PasswordSignInAsync(signInModel.Username,
signInModel.Password,signInModel.RememberMe,false);
      return result;
    }
    public async Task<bool> Role(SignUpModel userModel)
    {
      var user = new IdentityUser()
      {
        UserName = userModel.Username
      };

     var result= await _usermanager.IsInRoleAsync(user,"Admin");
```

```
        return result;
    }
    public async Task SignOutAsync()
    {
        await _signInManager.SignOutAsync();
    }
}
```

1.5 Модель роботи з базою даних

```
public partial class ForzaContext : IdentityDbContext
{
    public ForzaContext()
    {
    }

    public ForzaContext(DbContextOptions<ForzaContext> options)
        : base(options)
    {
    }



    public virtual DbSet<Userdatum> Userdata { get; set; } = null!;

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        if (!optionsBuilder.IsConfigured)
        {

optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=Forza;Username=postgres;Password=postgres");
        }
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {

        modelBuilder.Entity<Userdatum>(entity =>
        {
            entity.HasKey(e => e.Username)
                .HasName("userdata_pkey");

            entity.ToTable("userdata");

            entity.Property(e => e.Username).HasColumnName("username");
```

```
            entity.Property(e => e.Creditcard).HasColumnName("creditcard");

            entity.Property(e => e.Firstname).HasColumnName("firstname");

            entity.Property(e => e.Lastname).HasColumnName("lastname");

            entity.Property(e => e.Passportnumber).HasColumnName("passportnumber");

            entity.Property(e => e.Passportserial).HasColumnName("passportserial");

        });

        base.OnModelCreating(modelBuilder);
    }

}
```

2 КОНТРОЛЕРИ

2.1 AccountController

```
public class AccountController : Controller
  {
    private readonly IAccountRepository _accountRepository;
    private readonly ForzaContext db = new ForzaContext();
    public AccountController(IAccountRepository accountRepository)
    {
      _accountRepository = accountRepository;
    }

    [Route("signup")]
    public IActionResult SignUp()
    {
      return View();
    }
    [Route("signup")]
    [HttpPost]
    public async Task<IActionResult> Signup(SignUpModel userModel)
    {
      if(ModelState.IsValid)
      {
        var result = await _accountRepository.CreateUserAsync(userModel);
        if(!result.Succeeded)
        {
          foreach(var error in result.Errors) { ModelState.AddModelError("", error.Description); }

        }
        else
        {
          var user = new Userdatum();
          user.Username= userModel.Username;
          await db.Userdata.AddAsync(user);
          await db.SaveChangesAsync();

          return RedirectToAction("Index", "Home");
        }


      }
```

```
    return View();
}
[Route("signin")]
public IActionResult SignIn()
{
    return View();
}
[Route("signin")]
[HttpPost]
public async Task<IActionResult> SignIn(SignInModel signInModel)
{
    if(ModelState.IsValid)
    {
       var result= await _accountRepository.PasswordSignInAsync(signInModel);
        if (result.Succeeded)
        {
            return RedirectToAction("Index", "Home");
        }
        else ModelState.AddModelError("", "Неправильний логін або пароль");
    }
    return View();

}
[Route("signout")]
public async Task<IActionResult> SignOut()
{

    await _accountRepository.SignOutAsync();
    return RedirectToAction("Index", "Home");
}
[Route("profile")]
public IActionResult Profile()
{
    var result=db.Userdata.Where(m=>m.Username==User.Identity.Name).FirstOrDefault();


    return View(result);
}
[Route("profilechecking")]
public IActionResult ProfileChecking(Userdatum user)
{
    var result = db.Userdata.Where(m => m.Username == user.Username).FirstOrDefault();
```

```
    return View(result);
}
[Route("edit")]
public IActionResult Edit()
{
    return View();
}
[HttpPost]
public async Task<IActionResult> Edition(Userdatum user)
{
    user.Username = User.Identity.Name;
    db.Update(user);
    await db.SaveChangesAsync();
    return RedirectToAction("Profile","Account");
}
[Route("list")]
public IActionResult List()
{
    var result = db.Userdata.ToList();
    var model=new List<Userdatum>();
    model = result;
    return View(result);
}
[Route("check")]
public IActionResult Check(Userdatum user)
{


    return RedirectToAction("ProfileChecking", user);
}
[Route("editadmin")]
public IActionResult Editadmin(Userdatum user)
{


    return View(user);
}
[HttpPost]
public async Task<IActionResult> Editionadmin(Userdatum user)
{
```

```
            db.Update(user);
            await db.SaveChangesAsync();
            return RedirectToAction("List", "Account");
        }
        [Route("delete")]
        public async Task<IActionResult> Delete(Userdatum user)
        {
            db.Remove(user);
            await db.SaveChangesAsync();
            return RedirectToAction("List", "Account");
        }
    }
```

2.2 HomeController

```
public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly ForzaContext db= new ForzaContext();

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
     //      var result = db.Usrs.OrderBy(a=>a.Username).Last();
            return View();
        }

        public IActionResult About()
        {
            return View();

        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }    }
```

3 ВІДОБРАЖЕННЯ

3.1 _Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>@ViewData["Title"] - Форза</title>
   <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
   <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
   <link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />
   <link rel="stylesheet" href="~/Forza.styles.css" asp-append-version="true" />
</head>
<body>
   <header>

      <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-dark">
         <div class="container">
            <a class="navbar-brand" href="/Home/Index">Форза</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-
controls="navbar-nav" aria-expanded="false" aria-label="Toggle navigation">
             <span class="navbar-toggler-icon"></span>
            </button>
            <div class="navbar-collapse collapse  d-sm-inline-flex flex-sm-row-reverse">
            <ul class="navbar-nav">
                <partial name="_Login"/>
            </ul>
               <ul class="navbar-nav flex-grow-1" >
                  <li class="nav-item">
                  <a class="nav-link" href="/Home/Index" >Головна</a>
                  </li>
                  <li class="nav-item">
                  <a class="nav-link" href="/Home/About">Про нас</a>
                  </li>
               </ul>
         </div>
         </div>
      </nav>
   </header>
```

```
    <div class="container" >
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted text-center">
        <div class="container">
            &copy; @DateTime.Now.Year - Фінансова компанія "Форза"
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

3.2 _Login.cshtml

```
@using Forza.data;
@using Forza.Models;
@using Microsoft.AspNetCore.Identity;



@inject Microsoft.AspNetCore.Identity.SignInManager<IdentityUser> _signInManager;
@if(_signInManager.IsSignedIn(User))
{
    <li class="nav-item">
        <a class="btn btn-light text-dark" href="#">Вітаємо, @User.Identity.Name</a>
        @if (ViewContext.HttpContext.User.IsInRole("Admin"))
        {
            <a class="btn btn-light text-dark" asp-action="List" asp-controller="Account">Список користувачів</a>
        }
        else
        {
            <a class="btn btn-light text-dark" asp-action="Profile" asp-controller="Account">Профіль</a>
        }
        <a class="btn btn-light text-dark" asp-action="SignOut" asp-controller="Account">Вийти</a>
    </li>
}
```

```
else{

<li class="nav-item">
   <a class="btn btn-outline-primary text-white" href="/signin">Увійти</a>
   <a class="btn btn-primary text-white" href="/signup">Зареєструватися</a>
</li>
}
```

3.3 Index.cshtml

```
@{
   ViewData["Title"] = "Головна";
}

<section class="jumbotron text-center">
   <div class="container">
      <h1 vertical-align="center">Вас вітає вебдодаток Фінансової компанії "Форза"</h1>

   </div>

</section>
```

3.4 About.cshtml

```
@{
   ViewData["Title"] = "Про нас";
}
<h1>@ViewData["Title"]</h1>

<p>Товариство з обмеженою відповідальністю «Фінансова компанія «Форза» позиціонує себе як онлайн банкінг, тому основними бізнес процесами є робота з клієнтами. Прикладом послуг, які надає компанія є кредитування, дебетування  та надання послуг з факторингу. ТОВ «ФК «Форза» реалізує свої послуги на території України, обслуговує та професіонально проводить фінансові операції з клієнтами. Штат компанії містить у собі 100 працівників. </p>
<p>
   До складу компанії входять головний офіс та три філіали. Головний офіс знаходиться за адресою м. Київ, вул. В. Гетьмана, 6, адреси філіалів:
   <ul>
      <li>м. Київ, проспект Перемоги, 94/1;</li>
      <li>
         м. Київ, вул. Старовокзальна,23;</li>
      <li>м. Київ, вул. Костянтинівська,2А.
```

```
    </li>
  </ul>
</p>
```

3.5 Edit.cshtml

```
@model Forza.data.Userdatum;
@using Microsoft.AspNetCore.Identity;
<div class="container"> <h1 text="center">Дані профілю @User.Identity.Name</h1></div>
<hr />
<div class="container" id="#profile">
<form asp-action="Edition" asp-controller="Account">

  <table class="table table-striped table-dark text-center">

      <tr>

        <td>Ім'я користувача</td>
        <td>@User.Identity.Name</td>

      </tr>
      <tr>


      <td>Ім'я</td>
      <td><input asp-for="Firstname"></input></td>

      </tr>
      <tr>
        <td>Прізвище</td>
      <td><input asp-for="Lastname"></input></td>
      </tr>
    <tr>


      <td>Серія паспорту</td>
      <td><input asp-for="Passportserial"></input></td>

    </tr>
    <tr>
```

```
        <td>Номер паспорту</td>
        <td><input asp-for="Passportnumber"></input></td>

      </tr>
        <tr>
          <td>Кредитні картки</td>
          <td><input asp-for="Creditcard"></input></td>
        </tr>
        <tr>
          <td></td>
        <td>
          <div class="form-group">
                <input type="submit" value="Підтвердити" class="btn btn-light text-dark" />
          </div></input>
        </td>
        </tr>

   </table>
</form>
</div>
```

3.6 Editadmin.cshtml

```
@model Forza.data.Userdatum;
@using Microsoft.AspNetCore.Identity;
<div class="container"> <h1 text="center">Дані профілю @User.Identity.Name</h1></div>
<hr />
<div class="container" id="#profile">
<form asp-action="Editionadmin" asp-controller="Account">

   <table class="table table-striped table-dark text-center">

        <tr>

          <td>Ім'я користувача</td>
          <td>@Model.Username</td>

        </tr>
        <tr>


        <td>Ім'я</td>
```

```
        <td><input asp-for="Firstname"></input></td>

        </tr>
        <tr>
          <td>Прізвище</td>
        <td><input asp-for="Lastname"></input></td>
        </tr>
      <tr>


        <td>Серія паспорту</td>
        <td><input asp-for="Passportserial"></input></td>

      </tr>
      <tr>


        <td>Номер паспорту</td>
        <td><input asp-for="Passportnumber"></input></td>

      </tr>
        <tr>
          <td>Кредитні картки</td>
          <td><input asp-for="Creditcard"></input></td>
        </tr>
        <tr>
          <td></td>
        <td>
          <div class="form-group">
                <input type="submit" value="Підтвердити" class="btn btn-light text-dark" />
          </div></input>
        </td>
        </tr>

    </table>
</form>
</div>
```

3.7 List.cshtml

```
@using Forza.data;
@model List<Userdatum>
```

```
@{
   int i = 1;
}
<div class="container"> <h1 text="center">Список користувачів</h1></div>
<hr />
<table class="table table-striped table-dark text-center">
   <thead>
      <tr>
         <th scope="col">#</th>
         <th scope="col">Ім'я користувача</th>
         <th scope="col">Ім'я</th>
         <th scope="col">Прізвище</th>
         <th scope="col">Панель редагування</th>
      </tr>
   </thead>

   @foreach (Userdatum item in Model)
   {

      <tr>
         <td>@i</td>
         <td>@item.Username</td>
         <td>@item.Firstname</td>
         <td>@item.Lastname</td>
         <td>
            <a href="/check?Username=@item.Username" class="btn btn-light text-dark">Деталі аккаунту</a>
            <a href="/delete?Username=@item.Username" class="btn btn-light text-dark">Видалити аккаунт</a>

         </td>
      </tr>

         i++;

   }



</table>
```

3.8 Profile.cshtml

```
@model Forza.data.Userdatum;
```

```
<div class="container"> <h1 text="center">Дані профілю @User.Identity.Name</h1></div>
<hr />
<div class="container">
   <table class="table table-striped table-dark text-center">

      <tr>

         <td>Ім'я користувача</td>
         <td>@User.Identity.Name</td>

      </tr>
      <tr>


         <td>Ім'я</td>
         <td>@Model.Firstname</td>

      </tr>
      <tr>
         <td>Прізвище</td>
         <td>@Model.Lastname</td>
      </tr>
      <tr>


         <td>Серія паспорту</td>
         <td>@Model.Passportserial</td>

      </tr>
      <tr>


         <td>Номер паспорту</td>
         <td>@Model.Passportnumber</td>

      </tr>
      <tr>
         <td>Кредитна картка</td>
         <td>@Model.Creditcard</td>
      </tr>

   </table>
```

```
    <a asp-action="Edit" asp-asp-controller="Account" class="btn btn-light text-dark">Редагувати дані профілю</a>
</div>
```

3.9 Profileadmin.cshtml

```
@model Forza.data.Userdatum;

<div class="container">
    <h1 text="center">
        Дані профілю @Model.Username</td>
    </h1>
</div>
<hr />
<div class="container">
    <table class="table table-striped table-dark text-center">

        <tr>

            <td>Ім'я користувача</td>
            <td>@Model.Username</td>

        </tr>
        <tr>


            <td>Ім'я</td>
            <td>@Model.Firstname</td>

        </tr>
        <tr>
            <td>Прізвище</td>
            <td>@Model.Lastname</td>
        </tr>
        <tr>


            <td>Серія паспорту</td>
            <td>@Model.Passportserial</td>

        </tr>
        <tr>
```

```
            <td>Номер паспорту</td>
            <td>@Model.Passportnumber</td>

        </tr>
        <tr>
            <td>Кредитна картка</td>
            <td>@Model.Creditcard</td>
        </tr>

    </table>
    <a class="btn btn-light text-dark" href="/editadmin?Username=@Model.Username">Редагувати дані профілю</a>
</div>
```

3.10 SignIn.cshtml

```
@model Forza.Models.SignInModel

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>@ViewData["Title"] - Forza</title>
<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
<link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
<link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />
<link rel="stylesheet" href="~/Forza.styles.css" asp-append-version="true" />
<title>Увійти</title>
</head>
<body>
    <div class="container">
<h4>Увійти в аккаунт</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="SignIn">
```

```
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Username" class="control-label">Ім'я користувача</label>
                <input asp-for="Username" class="form-control" />
                <span asp-validation-for="Username" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password" class="control-label">Пароль</label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group form-check">
                <label class="form-check-label">
                    <input class="form-check-input" asp-for="RememberMe" /> Запам'ятати мене
                </label>
            </div>
            <div class="form-group">
                <input type="submit" value="Увійти" class="btn btn-primary" />
            </div>
        </form>
    </div>

        </div>
<div>
    <a asp-action="signup" asp-controller="Account">Не маєте аккаунту? Зареєструватися</a>
</div>
    </div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
</body>
</html>
```

3.11 SingUp.cshtml

```
@model Forza.Models.SignUpModel

@{
    Layout = null;
}

<!DOCTYPE html>
```

```
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewData["Title"] - Forza</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />
    <link rel="stylesheet" href="~/Forza.styles.css" asp-append-version="true" />
    <title>Зареєструватися</title>
</head>
<body>
<div class="container">
<h4>Зареєструватися</h4>
<hr />

<div class="row">
    <div class="col-md-4">
        <form asp-action="Signup">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Username" class="control-label">Ім'я користувача </label>
                <input asp-for="Username" class="form-control" />
                <span asp-validation-for="Username" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password" class="control-label">Пароль</label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ConfirmPassword" class="control-label">Підтвердити пароль</label>
                <input asp-for="ConfirmPassword" class="form-control" />
                <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Зареєструватися" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
```

```
<div>
    <a asp-action="SignIn" asp-controller="Account">Маєте акаунт? Увійти</a>
</div>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
</body>
</html>
```