КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

РОЗРОБКА СИСТЕМИ ВІДНОВЛЕННЯ ЛОГІНУ ДО ПОШТОВОГО СЕРВЕРУ MAIL.UNIV.NET.UA

студентки 4 року навчання

Спеціальність: 123 «Комп'ютерна інженерія»

Олександри ПРИЩЕПИ

Науковий керівник:

кандидат фізико-математичних наук, доцент

Сергій ЗАГОРОДНЮК

До захисту допускаю

Завідувач кафедрою

Ухвалено на засіданні кафедри «__» __ 2023 р., протокол № __

КИЇВ 2023

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складає: 48 сторінок, містить 28 рисунків, 9 додатків і використано 11 інформаційних джерел.

Ключові слова: Active Directory, LDAP, ASP.NET, c#, html, css, js, mail.univ.net.ua, логін, пароль, заявка, оператор, пошук, глобальний адміністратор.

Досліджено питання важливості відновлення логіну. Розроблено вебсайт для зручного відновлення логіну та паролю користувача до поштового серверу mail.univ.net.ua. Детально описано процес розробки, виконано його програмну реалізацію.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

AD – Active Directory

LDAP - Lightweight Directory Access Protocol

DNS – Domain Name System

JS – Java Script

БД – база даних

РОЗДІЛ 1. ВСТУП

Університети по всьому світу активно використовують поштові сервери для обміну інформацією між студентами, викладачами та іншими учасниками освітнього процесу. Однак, забування логіну та пароля до поштового серверу стає поширеною проблемою, яка може призвести до незручностей та обмежень у доступі до важливих повідомлень та матеріалів.

Дослідження та статистика свідчать про значну кількість випадків, коли користувачі забувають свої паролі та змушені використовувати процедуру відновлення логіну та скидання паролю.

Наприклад, згідно з опитування Cyclonis Password Security Report[1] тільки 9,52% завжди пам’ятають свої дані для авторизації. 41,03% респондентів забувають свої паролі від одного до трьох разів на рік. 31,5% опитаних виконують процедуру відновлення паролю від чотирьох до дев’яти разів на рік. Від 10 до 15 разів відновлюють свої дані аж 10,99% опитаних, і 6,96% забувають свої паролі 16 або більше разів на рік (див. рис. 1.1).

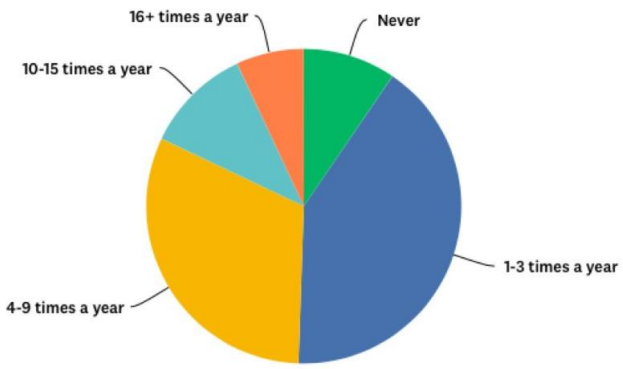


Рис.1.1 Графік опитування Cyclonis Password Security Report про те, скільки разів на рік користувачі забувають онлайн-паролі та змушені їх скидати.

Додатково, за результатами дослідження використання паролів, проведеного компанією HYPR протягом двох з половиною років серед більш ніж 500 штатних працівників у Сполучених Штатах і Канаді[2], виявлено, що за останні 90 днів 78% респондентів відновлювали пароль у своєму особистому житті, а 57% - у роботі (див. рис. 1.2).

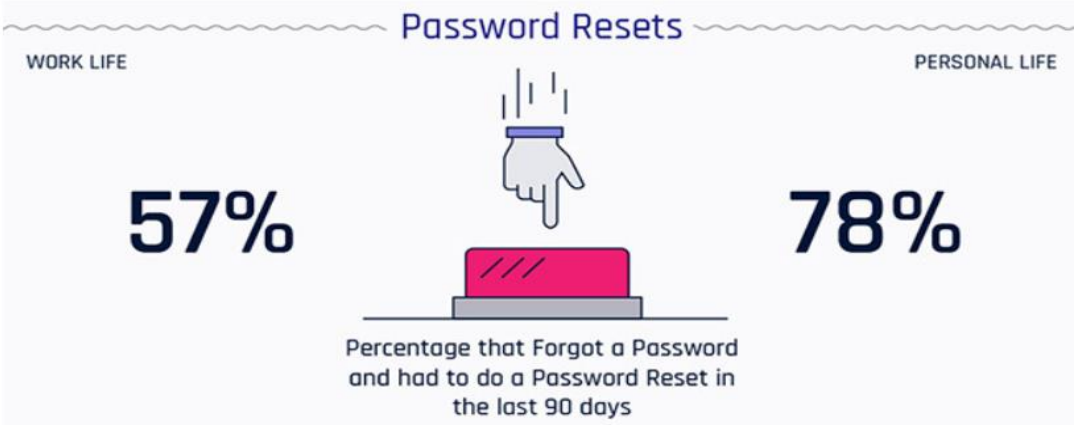


Рис. 1.2 Результати опитування проведеного компанією HYPR.

Отже, ці дані ще раз підтверджують, що забування логіну та пароля є поширеною проблемою. Важливість питання спрощення процедури відновлення логіну є досить високою.

Метою цієї роботи є розробка системи відновлення логіну до поштового серверу університету mail.univ.net.ua у вигляді вебсайту, який надасть користувачам зручний спосіб відновлення доступу до їхніх електронних скриньок. Основне завдання полягає в розробці механізму відновлення, який буде адаптованим під даний поштовий сервер та забезпечить можливість відновлення логіну у дистанційному форматі без особистої присутності на факультеті, а також надасть можливість оператору підтвердити заявку на відновлення логіну також дистанційно.

Актуальність роботи полягає у необхідності розробки ефективної системи відновлення логіну до поштового серверу університету в контексті навчання під час воєнного стану[3], адже студенти та працівники університету не мають можливості постійно знаходитися на факультеті особисто, з метою проведення процедури відновлення логіну та зміни паролю, а за необхідності - наражатимуть себе на небезпеку. Розробка вебсайту дозволить студентам і співробітникам університету відновити логін та змінити пароль дистанційно.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ВІДНОВЛЕННЯ ЛОГІНУ ТА ПАРОЛЮ

Цей розділ присвячений огляду та аналізу різних систем відновлення логіну, які використовуються в різних сервісах та платформах. Цей аналіз допоможе нам краще зрозуміти та врахувати найкращі практики при розробці власної системи відновлення логіну для поштового сервера mail.univ.net.ua.

Один з прикладів такої системи, на який варто звернути увагу, є система відновлення логіну в Gmail - популярному сервісі електронної пошти, розробленому компанією Google. Google пропонує користувачам можливість відновити свій логін за допомогою резервної електронної адреси або номеру телефону. Також обов’язково потрібно пам’ятати повне ім’я, указане у обліковому записі[4]. При відновленні логіну, Google надсилає відповідну інформацію на зазначену резервну електронну адресу або номер телефону, що дозволяє користувачу виконати необхідні кроки для відновлення доступу до свого облікового запису. Цікаво, в разі якщо користувач не пам’ятає ім’я користувача, компанія рекомендує створити новий обліковий запис[5], тобто дані вже відновити не вдасться.

Instagram реалізували декілька способів відновлення логіну та паролю. По аналогії з Google - з допомогою електронної пошти або номеру телефону, або за бажанням користувач може підв’язати свої облікові записи Instagram та Facebook. Проте якщо не маєте доступу до електронної пошти, облікові записи Instagram та Facebook не пов’язані, в такому разі доступ до облікового запису неможливо відновити[6].

Основною причиною, що унеможливлює реалізацію схожої системи відновлення логіну для mail.univ.net.ua, є відсутність номеру телефону та резервної електронної адреси в базі даних. При реєстрації електронної скриньки студенти та працівники університету не зобов'язані надавати ці дані. Раніше, коли забували логін або потребували змінити пароль, користувачі могли особисто звернутися до відповідальної особи на факультеті з оригіналом студентського квитка або

документу, який засвідчує особу, їм завжди допомагали у відновленні доступу до облікового запису.

РОЗДІЛ 3. ПОСТАНОВА ЗАДАЧІ

Технічним завданням є розробити вебсайт, який надасть можливість користувачам відновлювати свій логін та пароль до поштового серверу mail.univ.net.ua шляхом подання заявок.

Передбачається, що користувачами вебсайту будуть студенти, та працівники університету, які за певних причин не пам’ятають свій логін, випливаючи з цього потрібно реалізувати сторінку для прийняття заявок на відновлення логіну та паролю. На даній сторінці має бути реалізована форма, де студенти можуть ввести свої персональні дані, такі як: повне ім'я, логін, номер телефону, відмітити належність до певного підрозділу та прикріпити фото студентського квитка.

Зауважимо, що не всі користувачі вносили при реєстрації поштової скриньки унікальну інформацію про себе, наприклад: номер телефону, резервну пошту, номер студентського квитка чи унікальні паспортні дані такі, як серія або номер паспорту чи ідентифікаційний номер. З цього випливає, що реалізувати систему відновлення логіну без відповідальних осіб неможливо. Тож окремою частиною технічного завдання є реалізувати функціонал для оператора, тобто відповідальної особи за відновлення логіну у певному підрозділі, тобто факультеті.

Затребуваний функціонал для оператора:

- Авторизація;
- Пошук студентів;
- Перегляд поточних заявок;
- Прийняття заявки;
- Відхилення заявки.

Оператор повинен мати доступ тільки до тих користувачів та заявок, які належать до його підрозділу, тобто оператор, який відповідальний за біологічний факультет, може прийняти заявку на відновлення логіну та паролю тільки у студента біологічного факультету.

У зв'язку з наявністю багатьох підрозділів і окремих операторів у кожному з них, функціонал управління операторами є критично важливим елементом проекту з налагодження системи відновлення логіну. Його реалізація доручена глобальному адміністратору.

Затребуваний функціонал для адміністратора:

- Авторизація;
- Загальний пошук;
- Перегляд поточних заявок;
- Перегляд прийнятих заявок;
- Перегляд відхилених заявок;
- Прийняття заявки;
- Відхилення заявки;
- Створення оператора;
- Видалення оператора;

Передбачається, що адміністратор матиме доступ до всього функціоналу програми.

Однією з основних вимог до розробки веб-сайту є його сумісність з платформою Windows Server 2008 R2. Зважаючи на це, важливо використовувати технології, фреймворки та інструменти, які підтримуються на цій платформі.

РОЗДІЛ 4. ПІДГОТОВКА ДО РОЗРОБКИ ТА ПРОЕКТУВАННЯ ВЕБСАЙТУ

4.1 Опис інструментів реалізації

Вебсайт буде реалізовано з використанням мови програмування C#, додатковим інструментом буде використовуватися HTML та CSS. C# дозволяє розробникам створювати багато типів безпечних і надійних програм, які працюють на платформі .NET[7]. C# використовується для розробки бізнес-логіки та функціональності, CSS використовується для оформлення та стилізації веб-сторінок, а HTML - для створення структури та вмісту веб-сторінок. Ця комбінація мов дозволить створити функціональний та привабливий вебсайт згідно поставленого технічного завдання.

Оскільки Windows Server 2008 R2 підтримує лише .NET Framework 3.5, тож при розробці вебсайту необхідно використовувати цю версію фреймворку.

4.1.1 ASP.NET Framework 3.5

ASP.NET Web Forms – це потужний інструмент для створення динамічних веб-сайтів за допомогою знайомої моделі перетягування та скидання, керованої подіями. Поверхня дизайну та сотні елементів керування та компонентів дозволяють швидко створювати складні сайти на основі інтерфейсу користувача з доступом до даних[8].

Основними компонентами ASP.NET 3.5 Web Forms є сторінки, серверні елементи управління (server controls), події та код-бехінд (code-behind).

Сторінки у ASP.NET 3.5 Web Forms використовують розмітку на основі мови розмітки ASP.NET (ASPX), яка поєднує HTML код зі спеціальними елементами, такими як серверні елементи управління та вбудований код. Це дозволяє створювати веб-сторінки з динамічним вмістом та інтерактивними елементами.

Серверні елементи управління є ключовою складовою ASP.NET 3.5 Web Forms. Вони представляють собою різноманітні елементи, такі як кнопки, текстові поля, списки, таблиці та багато інших. Ці елементи можуть мати події, які відбуваються на серверній стороні, такі як натискання кнопки або введення тексту в поле. Вони також

дозволяють здійснювати зв'язування даних, що полегшує роботу з базами даних та іншими джерелами даних.

Події в ASP.NET 3.5 Web Forms дозволяють реагувати на дії користувачів або системи. Наприклад, можна створити обробники подій, які виконуються при натисканні кнопки або при завантаженні сторінки. Це дозволяє відреагувати на події та змінювати стан веб-сторінки або виконувати додаткові дії.

Код-бехінд використовується для обробки подій та виконання додаткового програмного коду на серверній стороні. Він дозволяє розділити логіку розробки веб-сторінки від її представлення (розмітки та візуального вигляду) та бізнес-логіки. Код-бехінд може містити функції, методи та змінні, які дозволяють взаємодіяти з сервером, базою даних та іншими компонентами.

Опишемо типовий сценарій взаємодії елементів вебпрограми один з одним і з клієнтом, що здійснює запит форми цієї програми.

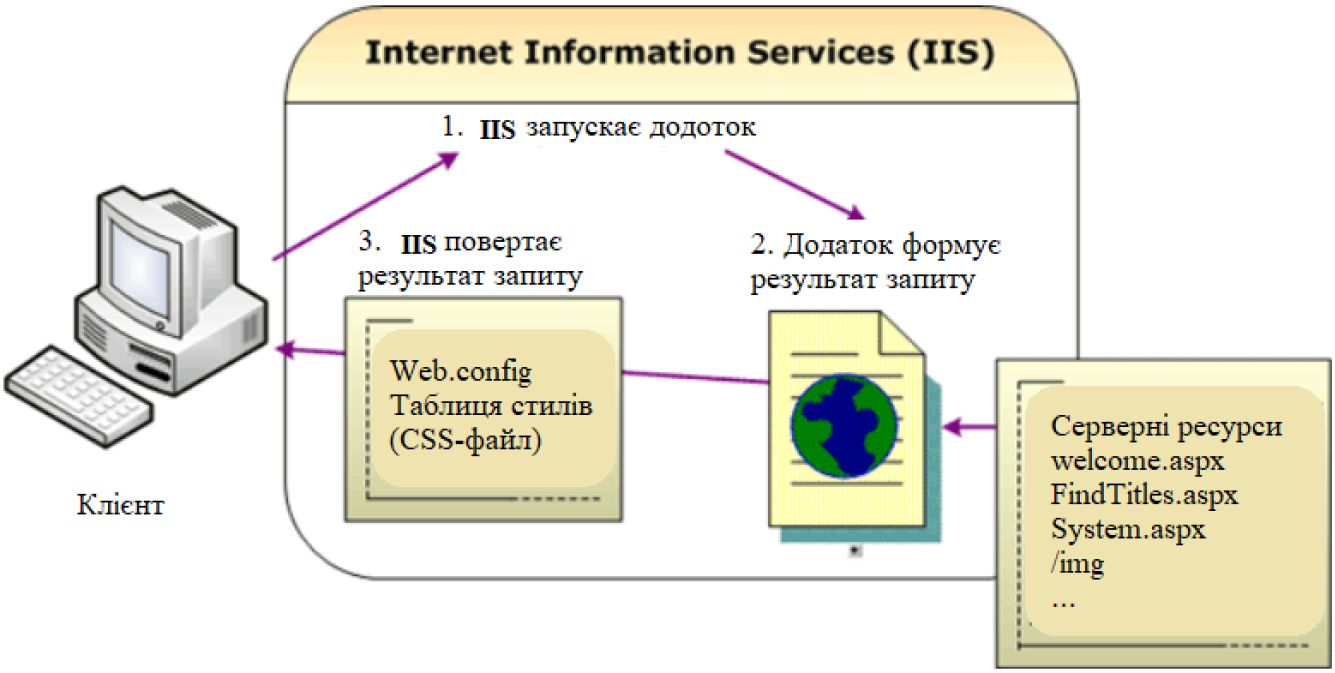


Рис. 4.1 Типовий сценарій взаємодії елементів вебпрограми з клієнтом.

Як видно із рис. 4.1 при зверненні клієнта до вебдодатку останній запускається на сервері IIS. Запущена програма формує відгук. Для цього на сервері створюється екземпляр запитаної вебформи, вона генерує HTML-текст відгуку, який передається

браузеру клієнта. Відразу після цього екземпляр вебформи знищується. Користувач, отримавши HTML-сторінку, згенеровану програмою, має можливість заповнювати різні поля форми (тестові поля, перемикачі тощо). Після заповнення всіх необхідних полів форми, користувач ініціює відправлення даних, введених ним у сторінку, назад на сервер. Це відбувається за рахунок використання технології зворотного відсилання, що викликається під час виконання певних дій (наприклад, натискання на кнопку). Отримавши дані від користувача, сервер створює новий екземпляр вебформи, заповнює його отриманими даними та обробляє всі необхідні події. Після закінчення обробки сервер формує HTML-код відповіді та відправляє його клієнту, а потім знищує екземпляр вебформи.

4.1.2 LDAP

LDAP (Lightweight Directory Access Protocol) - це відкритий і кросплатформний протокол, який використовується для автентифікації служб каталогів[9].

LDAP дозволяє програмам взаємодіяти з іншими серверами служб каталогів. Це важливо, тому що служби каталогів зберігають та передають важливу конфіденційну інформацію, пов'язану з користувачами, паролями та обліковими записами комп'ютерів. Запит LDAP - це команда, яка запитує у служби каталогів певну інформацію(див. рис. 4.2). Наприклад, якщо потрібно побачити, до яких груп входить конкретний користувач.

```
using (DirectoryEntry AD = new DirectoryEntry("LDAP://OU=Request,DC=PRISEPA,DC=NET,DC=UA"))
{
    using (DirectoryEntry u = AD.Children.Add("CN=" + full_name + " " + phone_number.ToString(), "contact"))
    {
```

Рис. 4.2 Запит LDAP з проекту.

4.1.3 Active Directory

Active Directory, який є службою каталогів, відіграє таку важливу роль у структурі IT-інфраструктури більшості організацій. Служба каталогів - це система програмного забезпечення, яка зберігає, організовує та надає доступ до інформації в каталозі операційної системи комп'ютера. Active Directory - це реалізація служб каталогів, яка надає всі види функцій, таких як автентифікація, управління групами

та користувачами, адміністрування політик та багато іншого. Active Directory служить єдиним сховищем даних для швидкого доступу до даних для всіх користувачів та контролює доступ для користувачів на основі політики безпеки каталогу.

Атрибут : атрибут — це елемент даних у записі, який пов’язує опис атрибута з набором значень.

Коротко кажучи: AD – це база даних служб каталогів, а LDAP – один із протоколів, які у даній роботі використовується для спілкування з нею. LDAP – це протокол, а Active Directory – це сервер.

4.2 Підготовчі процеси

У цьому розділі буде розглянуто налаштування середовища розробки, а також детальний опис реалізації коду і функцій, необхідних для правильної роботи системи відновлення облікових даних.

4.2.1 Розробка структури вебсайту

Перед початком розробки вебсайту важливо визначити його структуру. Структура сайту визначає схему розташування його сторінок і логічний зв'язок між ними. Вона відображає організацію контенту і навігаційну структуру сайту.

З точки зору технічної реалізації, структура сайту включає набір URL-адрес, які представляють різні сторінки та документи, розташовані на сайті. Ці URL-адреси організовані в певну ієрархію папок і каталогів, що спрощує управління та навігацію по сайту.

Використаємо деревоподібну структуру для побудови вебсайту (див. рис. 4.3). Деревоподібна структура присутня через наявність головної вебсторінки, яка є кореневим вузлом, а з неї можна перейти до інших сторінок. Наприклад, вебсторінка авторизації та вебсторінка подачі заявок є дочірніми вузлами, які виходять з головної сторінки. Це дозволяє створити ієрархічну структуру з кількома рівнями вкладеності.

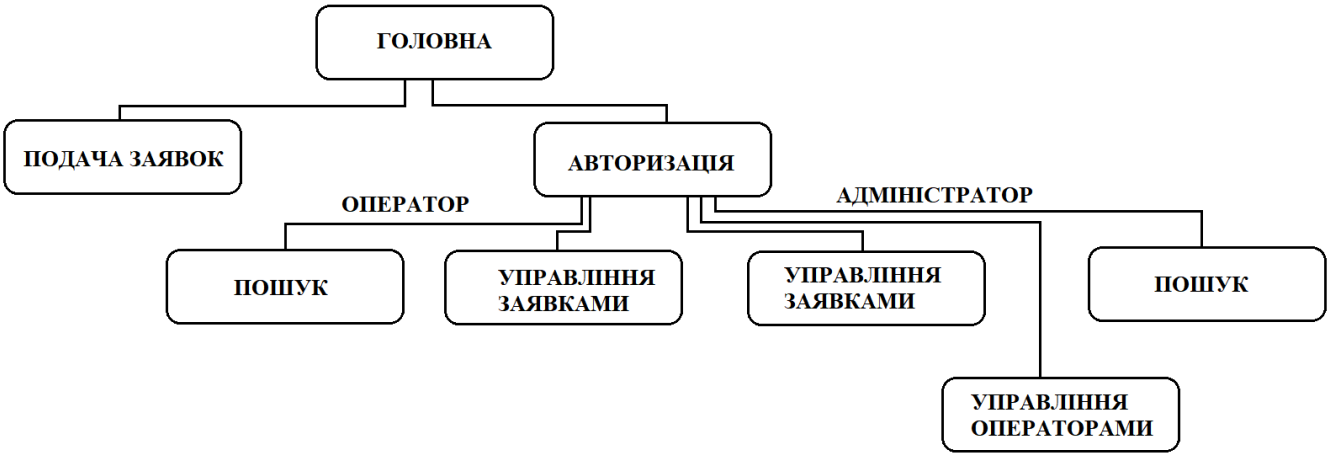


Рис 4.3. Блок-схема структури сайту.

4.2.2 Налаштування середовища розробки

Налаштування середовища розробки відіграє важливу роль у процесі розробки. В даному випадку, розробка функціоналу для відновлення логіну є прямим доповненням поштового серверу mail.univ.com.ua і має на меті подальшу інтеграцію в систему. З міркувань безпеки, на початковому етапі розробки неможливо використовувати актуальну базу даних та доменне ім'я, тому першим кроком налаштування середовища є встановлення віртуальної машини Oracle, на якій буде розгорнуто операційну систему Windows Server 2008 R2 та налаштовано DNS-суфікс PRISEPA.NET.UA (див. рис. 4.4).

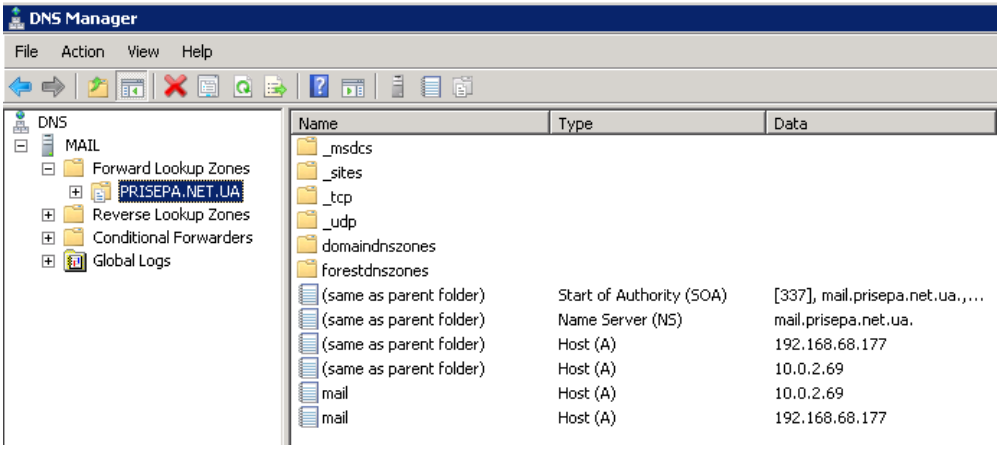


Рис. 4.4 Створене доменне ім’я PRISEPA.NET.UA на сервері.

Наступним етапом є встановлення та налаштування Active Directory на нашому сервері. Створення потрібних Organizational Unit(див. рис. 4.5):

- MAIL-Users - в даній OU зберігаються активні користувачі поштового серверу.
- Operators - в даній OU зберігаються створені оператори, які є відповідальними особами за відновлення логіну та паролю.

Зберігати заявки доцільно в Contact, а не в OU за наступних причин:

- Неактивні об'єкти: Заявки на відновлення логіну відносяться до неактивних об'єктів, оскільки це не користувачі, які активно використовуються в домені. Контакти, зокрема, призначені для зберігання інформації про неактивних осіб або ресурси, тому вони підходять для зберігання заявок на відновлення логіну.
- Контактні дані: Контакт може містити різноманітну інформацію про заявника, таку як його ім'я, контактні дані (телефон, електронна пошта) та інші важливі деталі. Це дозволяє зберігати заявку, яка і містить контактні дані заявника.

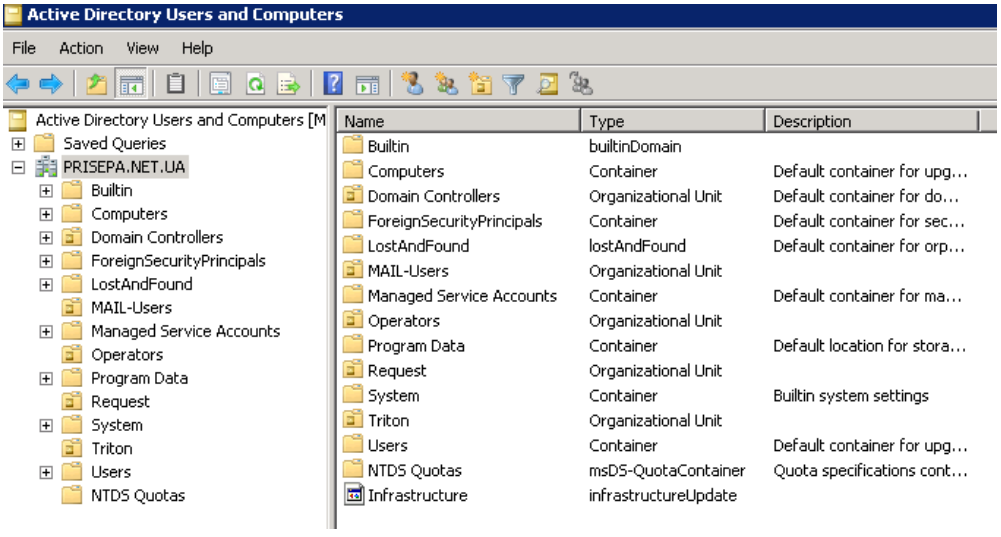


Рис. 4.5 AD зі створеними необхідними в проекті OU.

4.2.3 Структура проекту

1. Models - в цій папці знаходяться класи моделей(див. рис. 4.6).

2. Services - знаходяться класи, які містять бізнес-логіку додатку, такі як сервіси для взаємодії з базою даних.

3. wwwroot - ресурси, доступні для веб-сторінок, такі як зображення та файли CSS і JavaScript (main.css та main.js). Ці ресурси використовуються для візуального оформлення та функціональності веб-сторінок.

4. Web Forms - веб-форми, а саме Default.aspx, Application.aspx, Authorization.aspx, Request.aspx та Search.aspx, представляють різні сторінки вебсайту, на яких користувач може виконувати певні дії. Наприклад, сторінка Application.aspx створена для подання запиту на відновлення логіну.

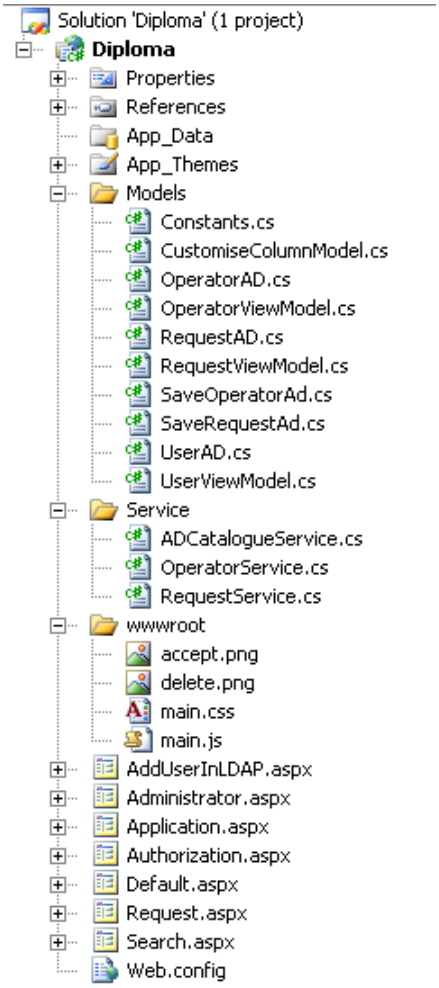


Рис. 4.6 Структура проекту.

Зауваження: веб форма AddUserInLDAP було створено виключно з метою заповнення даними AD. Вона не відображається для кінцевих користувачів і не має прямого впливу на функціональність основної системи. Її основна роль полягає в підготовці тестових даних для виконання різних сценаріїв та перевірки правильності роботи системи.

РОЗДІЛ 5. ПРОГРАМНА РЕАЛІЗАЦІЯ

5.1 Constants

Клас Constants було створено для полегшення впровадження і забезпечення централізованого керування значеннями констант по всьому проекту. Наприклад, Constants містить константу LDAPUserConnectionString (див. рис. 5.1). При впровадженні системи, коли зміниться з'єднання з LDAP-сервером, потрібно буде змінити значення константи LDAPUserConnectionString лише в класі Constants, і ця зміна автоматично відобразиться в усіх місцях, де вона використовується. Такий підхід спрощує обслуговування і уникнення помилок, пов'язаних з некоректними значеннями констант. Код класу Constants наведено в Додатку А.

```
public static class Constants
{
    public static string LDAPUserConnectionString = "LDAP://OU=MAIL-Users,DC=PRISEPA,DC=NET,DC=UA";
```

Рис. 5.1 Константа LDAPUserConnectionString.

5.2 Models

Папка "Models" використовується для збереження класів моделей. Модель в програмуванні використовується для представлення даних та бізнес-логіки в програмі. У проекті моделі містять властивості, які описують характеристики даних.

Принципова різниця між ViewModel та AdModel:

ViewModel виступає посередником між моделлю даних і представленням (користувацьким інтерфейсом) і містить дані, необхідні для відображення та обробки цих даних в інтерфейсі користувача. Наглядним прикладом є використання фото, як готовий рядок, де вже поєднано mimeType photo вже з готовим рядком Base64.

AdModel використовується для представлення даних або об'єктів, пов'язаних з AD, тобто об’єкти, а саме так, як вони зберігаються в базі. На прикладі того ж фото, mimeType зберігається в окремому полі, а решта коду зображення в іншому полі. Така дефрагментація дозволяє швидко відфільтрувати фото по його типу, не витягуючи всю фотографію загалом.

Опис всіх моделей проекту:

CustomiseColumnModel: модель використовується для налаштування стовпців в інтерфейсі. Вона містить властивості, такі як ProperityName (ім'я властивості), DisplayName (відображуване ім'я, тобто назва стовпця), Width (ширина стовпця).

SaveRequestAd: модель представляє дані для збереження запиту. Властивості: FirstName (ім'я), SecondName (прізвище), LastName (по-батькові), Email (електронна пошта), Faculty (факультет), PhoneNumber (номер телефону), Image (зображення, а саме фото студентського квитка) та ImageType (тип зображення).

RequestAD: модель представляє запит на відновлення логіну в AD. Властивості: Id, FirstName (ім'я), SecondName (прізвище), LastName (по-батькові), Email (електронна пошта), PhoneNumber (номер телефону), FacultyName (назва факультету), Image (зображення) та Status (статус запиту).

RequestViewModel: модель використовується для відображення запиту на відновлення логіну в інтерфейсі. Властивості: FullName (повне ім'я), SecondName (прізвище), FirstName (ім'я), LastName (по-батькові), PhoneNumber (номер телефону), Email (електронна пошта), FacultyName (назва факультету), ImageHtml (HTML-код зображення, фото студентського квитка) та ButtonsField (поле кнопок, а саме прийняти заявку чи відхилити).

UserAD: модель представляє дані про користувача з AD. Властивості: Id, FirstNameAndLastName (ім'я та по-батькові), SecondName (прізвище), FirstName (ім’я), LastName (по-батькові), UnivEmail (університетська електронна пошта, логін якої ми відновлюємо), AlternativeEmail (електронна пошта, а саме knu.ua), FacultyName (назва факультету).

UserViewModel: Ця модель використовується для представлення даних користувача в інтерфейсі. Властивості: FullName (повне ім'я), SecondName (прізвище), FirstName (ім'я), LastName (по-батькові), UnivEmail (університетська електронна пошта), AlternativeEmail (альтернативна електронна пошта, а саме knu.ua) та FacultyName (назва факультету).

SaveOperatorAd: модель використовується для збереження оператора в Active Directory. Властивості: Id, Logi, Password та FacultyName (назва факультету).

OperatorAD: модель представляє оператора в Active Directory. Властивості: Id, Login, Password , FacultyName (назва факультету) та Status (статус оператора).

OperatorViewModel: модель використовується для відображення оператора в таблиці. Властивості: Id, Login, Password, FacultyName та ButtonsField (поле кнопок, а саме кнопка видалити).

5.3 Services

Проект реалізовано з допомогою статичних класів(див. рис. 5.2). Основною перевагою є те, що не потрібно створювати об’єкт класу, щоб викликати його метод або отримати доступ до його полів, також методи статичного класу можна викликати по всьому проекту декілька разів. Це спрощує написання коду та використання функцій із статичного класу. Виклик статичного методу або доступ до статичного поля працює швидше, оскільки немає необхідності в управлінні пам'яттю для створення та знищення об'єктів.

```
namespace Diploma.Service
{
    public static class RequestService
    {
        public static bool SaveRequest(SaveRequestAd request)...

        private static string GetExtension(string mimeType) ...

        private static string GetMimeType(string extension)...

        public static RequestAD[] GetAllRequests(string facultyName)...

        public static ResultAccept AcceptRequest(Guid requestId)...

        public static ResultRemove RemoveRequest(Guid requestId)...

        private static void SetPassword(DirectoryEntry entry, string password)...

    }

    public class ResultAccept...

    public class ResultRemove ...

}
```

Рис. 5.2 Приклад вміст класу RequestService папки Service.

5.3.1 SaveRequest in AD

```
public static bool SaveRequest(SaveRequestAd request)
{
    try
    {
        using (DirectoryEntry AD = new DirectoryEntry(Constants.LDAPRequestConnectionString))
        {
            var fileId = Guid.NewGuid();
            var filePath = Constants.ImageFolderPath + fileId.ToString() + GetExtension(request.ImageType);
            using (DirectoryEntry u = AD.Children.Add("CN=" + fileId.ToString(), "contact"))
            {
                u.Properties["url"].Add(request.LastName);
                u.Properties["sn"].Add(request.SecondName);
                u.Properties["givenName"].Add(request.FirstName);
                u.Properties["description"].Add(request.Faculty);
                u.Properties["telephoneNumber"].Add(request.PhoneNumber);
                u.Properties["mail"].Add(request.Email);
                u.Properties["st"].Add("inProgress"); //string [deleted, acepted, inProgress]
                u.Properties["displayName"].Add(fileId.ToString() + GetExtension(request.ImageType));
                u.CommitChanges();
            }
            File.WriteAllBytes(filePath, request.Image);
        }
        return true;
    }
    catch(Exception e)
    {
        return false;
    }
}
```

Рис. 5.3 Метод збереження заявки в базу.

На рис. 5.3 код зберігає запит SaveRequestAd у AD та зберігає зображення, тобто фото студентського квитка у файлову систему, шлях винесено в константи(див. Додаток А). Зберігання фотографій у файловій системі дає більшу свободу щодо розміру та кількості фотографій, які можна зберегти, також це не впливає на розмір бази даних.

Основні етапи роботи коду:

1. Створюємо об'єкт AD для підключення до AD по рядку з'єднання Constants.LDAPRequestConnectionString;

2. Генеруємо fileId для запиту, використовуємо клас Guid, який генерує унікальний ідентифікатор.

Guid - це 128-бітне унікальний ідентифікатор, який зазвичай представляється у форматі строки складеної з 32 шістнадцяткових цифр, розділених дефісами. Особливістю Guid є те, що він є унікальним , і шанси, що він повториться є дуже малі,

адже він використовує алгоритми , які враховують різні фактори, наприклад час, мережеву адресу, унікальні індентифікатори пристроїв та інші.

3. Створення шляху для збереження зображення, шлях складається з Constants.ImageFolderPath, fileId.ToString() та розширення файлу, отриманого з GetExtension(request.ImageType), зберігаючи так фото в подальшому по унікальному fileId злегкістю можна буде витягнути його з бази в табличку.

4. Створення об'єкту u у дочірній папці AD з ім'ям "CN=" + fileId.ToString() і типом "contact", задається ім'я об'єкту з рядком "CN=" + fileId.ToString() та типом "contact", тобто кожна заявка є унікальною навіть якщо студенти подадуть 2 запити з однаковими полями, то заявки в БД збережуться.

5. Дадаємо значення властивостей запиту до відповідних властивостей об'єкту u та збереження в AD.

6. Зберігаємо фото в файлову систему.

5.3.2 GetAllRequests

Метод GetAllRequests отримує дані з AD та створює об'єкти RequestAD, які представляють заявки користувачів. Код наведено в Додатку Б. Пошук у директорії виконується за допомогою об'єкта DirectorySearcher (див. рис. 5.4).

```
DirectorySearcher directorySearcher = new DirectorySearcher(directoryEntry);
SearchResultCollection searchResultCollection = directorySearcher.FindAll();
```

Рис. 5.4 Використання DirectorySearcher в методі GetAllRequests

Метод перебирає результати пошуку у директорії та отримує необхідні дані про всі заявки, точніше інформацію про студента, який створив заявку, таку як ПІБ, електронна пошта, номер телефону, приналежність до факультету, статус заявки. Під час перебору також перевіряє певні умови, наприклад, порівнює ім'я факультету та перевіряє, чи авторизований користувач є глобальним адміністратором. Для кожної заявки створюється об'єкт RequestAD, який заповнюється отриманими даними. Якщо доступне зображення документу, воно завантажується та додається до об'єкту

RequestAD. Усі створені об'єкти RequestAD додаються до списку requests. На кінці методу список requests повертається як результат. Тобто в результаті, отриманий масив requests містить об'єкти RequestAD, які представляють дані про заявки з AD. Цей метод дозволяє отримати всі заявки, які наявні в каталозі і використовується для заповнення таблиці даними.

5.4 Gridview

Код наведений в Додатку В, відповідає за заповнення даними GridView на основі отриманих заявок (об'єкти RequestAD[], див. попередній підрозділ). Метод FillData приймає масив заявок requestAds.

1. Створюється порожній список requestViewModels, який міститиме об'єкти RequestViewModel - модель для відображення даних заявок.

2. Виконується ітерація через кожен об'єкт requestAd в масиві requestAds та для кожного об'єкта requestAd створюється новий об'єкт RequestViewModel.

3. Значення полів RequestViewModel заповнюються даними з відповідних полів requestAd.

4. Генерується HTML-код для відображення зображення заявки (поле ImageHtml) та HTML-код для кнопок дій над заявкою (поле ButtonsField). Залежно від статусу заявки, кнопки відображаються або ні. Використання HTML коду для відображення фото та кнопок має кілька переваг, наприклад:

- Простота у написанні: якщо відображати кнопки та фото за допомогою c#, а це ускладнює та збільшує код у декілька разів
- Легкість внесення змін: Використання HTML коду дозволяє легко модифікувати розмітку та структуру елементів, також дозволяє створювати розмітку з різними елементами та стилізувати їх за допомогою CSS, тобто легко змінити вигляд кнопок.

5. Створений об'єкт RequestViewModel додається до списку requestViewModels, потім список фільтрується, відсіюються записи з порожніми полями.

6. Налаштовується вигляд та структура гріда (GridView1), включаючи встановлення кількості та ширини стовпців.

7. Дані з списку requestViewModels присвоюються джерелу даних гріда (GridView1.DataSource), Грід оновлюється, щоб відображати нові дані (GridView1.DataBind()).

Метод CustomiseColumn (див. рис. 5.5) використовується для налаштування вигляду стовпця гріда. Він створює об'єкт BoundField для кожного стовпця та налаштовує його властивості, такі як ширина та текст заголовка. Оновлений стовпець додається до колекції стовпців гріда (GridView1.Columns.Add(column)).

```
private void CustomiseColumn(CustomiseColumnModel columnModel)
{
    BoundField column = new BoundField();
    column.ItemStyle.BorderStyle = BorderStyle.Double;
    if (columnModel.Width != null)
        column.ItemStyle.Width = new Unit(columnModel.Width.Value.ToString() + "%");
    column.DataField = columnModel.ProperityName;
    if (columnModel.ProperityName == "ImageHtml" || columnModel.ProperityName == "ButtonsField")
        column.HtmlEncode = false;
    column.HeaderText = columnModel.DisplayName;
    GridView1.Columns.Add(column);
}
```

Рис. 5.5 Метод CustomiseColumn.

5.5 Процеси перевірки даних

5.5.1 Валідація

Розглянемо реалізацію валідації в проекті на прикладі поданої заявки, весь код валідації (див. Додаток Г). Перш за все, встановлюємо значення властивостей Visible для прапорів видимості для кожного елемента Label, щоб приховати їх спочатку. Виконуємо перевірку наявності даних у текстових полях форми (наприклад _PhoneNumber, textBox_SecondName)(див. рис № 5.6). Якщо яке-небудь поле є порожнім, встановлюється success = false, відповідне повідомлення помилки встановлюється у відповідному Label і його властивість видимості встановлюється на true.

```
var success = true;
if (string.IsNullOrEmpty(textBox_PhoneNumber.Text))
{
    success = false;
    label7.Visible = true;
    label7.Text = "Це поле обов`язкове!";
}
```

Рис. 5.6 Перевірка наявності даних на прикладі textBox_PhoneNumber.

Також присутня перевірка кількості цифр у textBox_PhoneNumber, перевірка обраного значення в DropDownList1, тобто факультета, обраного файлу для завантаження FileUpload1.HasFile (фото студентського квитка). Всі поля заявки є обов’язковими(див. рис. 5.7).

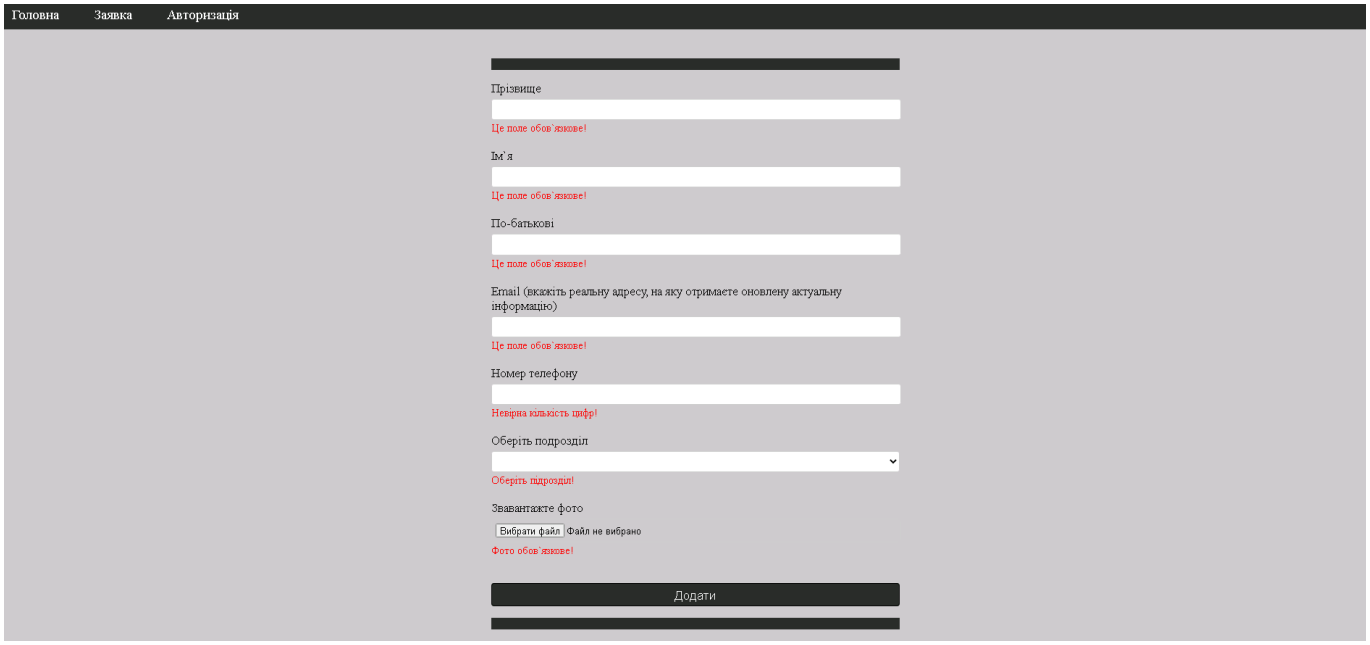


Рис. 5.7 Перевірка заповнення даних на сторінці заявки.

Після цього виконується перевірка значення success. Якщо воно дорівнює false, виконання методу зупиняється і виконується return, що означає, що немає додаткових дій. Якщо значення success дорівнює true, це означає, що всі перевірки пройшли успішно і дані у формі є валідними. Далі в коді створюється об'єкт SaveRequestAd, у якому зберігаються дані з текстових полів форми, після цього викликається метод RequestService.SaveRequest(saveRequest), який відправляє дані на сервер для збереження. Результат виконання цього методу зберігається у змінній result. Якщо

result дорівнює true, це означає, що заявка була успішно збережена. В такому випадку, відображається повідомлення успіху за допомогою ScriptManager.RegisterClientScriptBlock, що викликає JavaScript-код для відображення спливаючого повідомлення. Після цього відбувається перенаправлення на сторінку Application.aspx.

5.5.2 Обробка помилок

На прикладі прийняття заявки розглянемо реалізацію обробок помилок. У коді використовуються об'єкт класу ResultAccept, який містить різні властивості для відстеження можливих помилок (див. Додаток Д). В класі ResultAccept використовуються властивості з методами get і set для доступу до значень полів класу, окрім властивості Success, у неї використовується тільки метод get, оскільки вона обчислюється на підставі значень інших полів класу.

Основний принцип обробки помилок полягає в тому, що після виконання певних дій або операцій перевіряється стан об'єкта ResultAccept, щоб визначити, чи виникли помилки. Кожна властивість ResultAccept відповідає певному типу помилки, яку можна зустріти під час обробки запиту.

Властивості:

NotFoundRequest - запит не знайдено;

MoreThenOneUser - знайдено більше одного користувача, що відповідає запиту;

NotFoundUser – не знайдено користувача, який пов'язаний з запитом;

ErrorWhileProcessing - помилка під час обробки запиту;

RequestNotInProgress – заявка або вже прийнята або відхилена;

ErrorWhileSaveLDAP - помилка під час збереження даних у LDAP, а саме нового паролю;

Success – успішна операція.

Основні кроки обробки помилок в коді(див. Додаток Е):

1. Створюється об'єкт ResultAccept, який ініціалізується зі значеннями за замовчуванням (усі властивості мають значення false).

2. Після кожної важливої операції або перевірки встановлюються відповідні властивості ResultAccept, якщо виникає помилка.

3.Після цього перевіряється значення властивості Success об'єкта ResultAccept. Якщо вона дорівнює true, це означає, що немає жодних помилок, і виконується певна логіка або виводиться повідомлення про успіх.

4. Якщо значення властивості Success дорівнює false, перевіряються окремі властивості ResultAccept для визначення типу помилки, яка сталася. В залежності від цього встановлюється відповідне повідомлення про помилку, яке потім може використовуватися для відображення користувачеві(див. Додаток Є).

5.6 Використання js в проекті.

В проекті JavaScript використовується для різних функціональних можливостей (див. Додаток Ж). Наприклад:

1. Зміна стилів рядків таблиці

JavaScript використовуємо для зміни стилів рядків таблиці залежно від значення атрибуту кнопки. Наприклад, коли значення атрибуту є removed, тобто заявку відхилено, змінюється колір фону батьківського елемента рядка на сірий, а коли значення атрибуту є accepted, тобто прийнято, колір фону змінюється на блідо-зелений. Це дозволяє візуально позначати рядки таблиці відповідно до їх стану.

2. Обробка подій кліку на кнопки та зображення

Наприклад, коли користувач натискає на кнопку Accept Request (acceptRequest(el)), викликається функція, яка отримує значення елемента input, записує його в відповідний елемент на сторінці та викликає приховану кнопку для обробки подальших дій.

3. Показ повноекранного зображення

Коли користувач натискає на зображення, створюється новий елемент div з класом .fullscreen, який містить зображення. Цей елемент додається до document.body. При натисканні на елемент div, він видаляється, повертаючи користувача до попереднього вигляду.

4. Взаємодія з сервером

Наприклад, при натисканні на кнопку "Remove Request" (removeRequest(el)), викликається функція, яка отримує значення елемента input, передає його на сервер і викликає відповідний запит для видалення вибраного запиту. Те саме відбувається при натисканні кнопки "Remove Operator" (removeOperator(el)), де відповідний ідентифікатор передається на сервер для видалення оператора.

РОЗДІЛ 6. РЕЗУЛЬТАТИ

У цьому розділі будуть наведені докладні результати, які охоплюють кожен етап процесу відновлення логіну та паролю. Ви знайдете покроковий опис кожного етапу, супроводжений відповідними зображеннями, що ілюструють результати роботи. Цей аналіз та опис результатів сприятиме кращому розумінню процесу відновлення даних користувача, а також дозволить оцінити ефективність та надійність розробленого рішення. На даний момент вебсайт є у вільному доступі.

6.1 Вебсторінка подачі заявок

У процесі відновлення логіну та паролю користувача, який забув свої облікові дані, він спочатку переходить на сторінку подачі заявок(див. рис. 6.1). На цій сторінці йому потрібно заповнити всі поля, тобто правильно вказати своє ПІБ, актуальну поштову скриньку, на яку йому буде надіслано оновлену інформацію про його обліковий запис, номер телефону (номер телефону потрібен для зв’язку з користувачем у випадку, якщо у оператора виникнуть питання, які він зможе узгодити за допомогою дзвінка), обрати підрозділ, до якого він належить та завантажити фото для підтвердження своєї особистості, для студента передбачається студентський квиток.

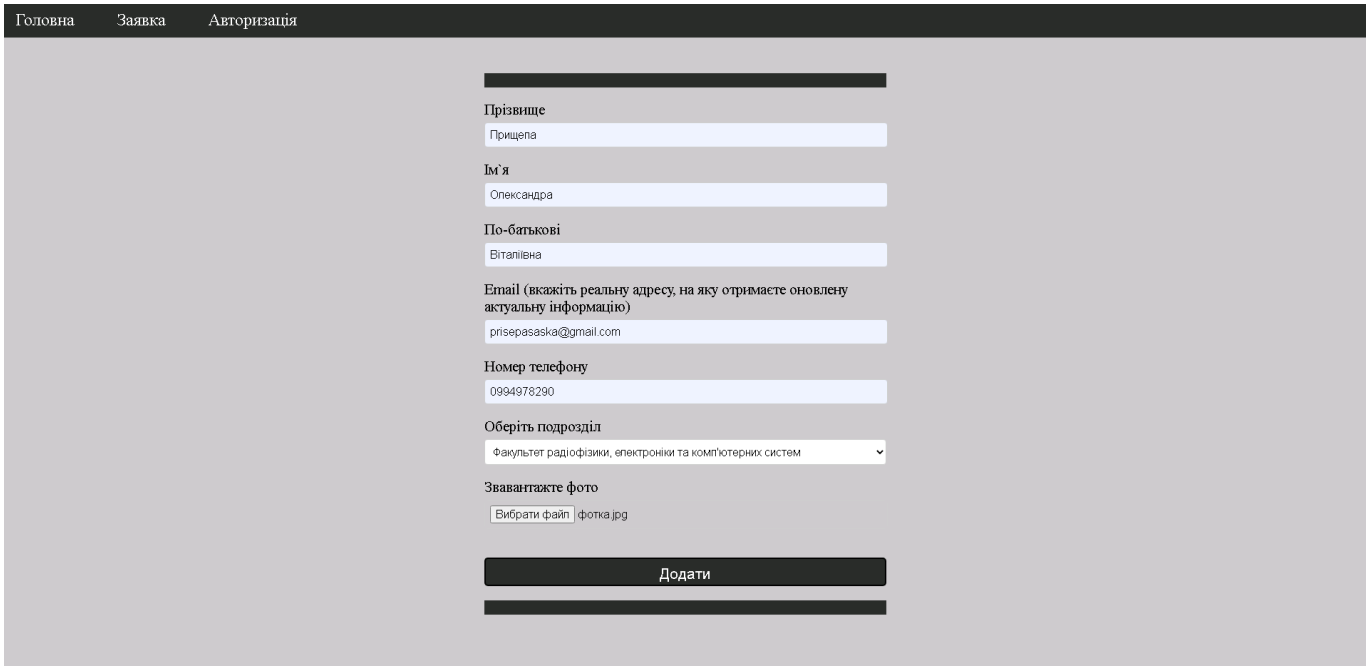


Рис. 6.1 Заповнена вебсторінка заявки на відновлення логіну та паролю.

6.2 Авторизація

Для підтвердження заявки, оператор, як відповідальна особа, повинен авторизуватися в особистому кабінеті оператора. Оператор повинен ввести свої облікові дані, такі як логін та пароль, на сторінці авторизації(див. рис. 6.2). Після успішної авторизації, він отримує доступ до особистого кабінету оператора, де може переглядати список заявок.

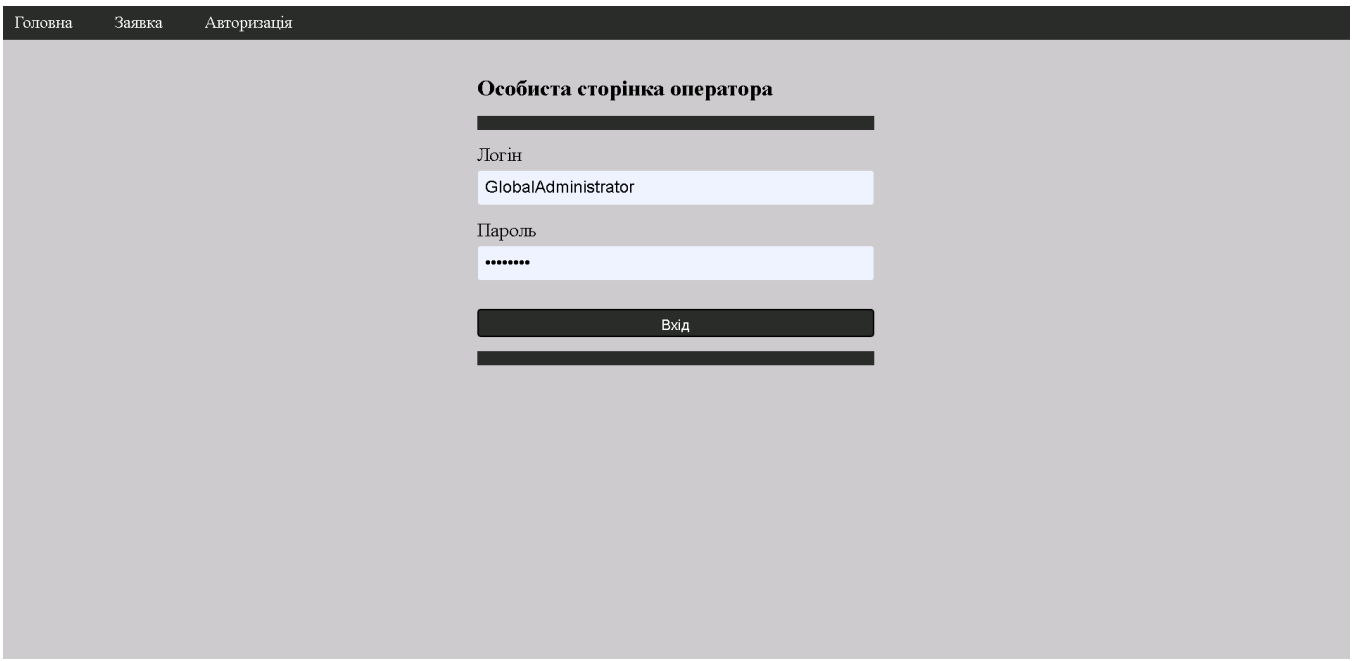


Рис. 6.2 Вебсторінка авторизації.

Зауваження: Зазвичай, у реальних системах перші адміністраторські облікові записи створюються під час процедури ініціалізації системи. У даному випадку, першого глобального адміністратора в базу даних додано безпосередньо в коді. Це дозволило розпочати використання системи авторизації та перевірити, чи працює вона належним чином, до того як будуть додані інші функціональності та реальні адміністратори.

6.3 Функціонал Глобального адміністратора

6.3.1 Вебсторінка запитів

Глобальний адміністратор володіє спеціальними привілеями та повноваженнями, що дозволяють йому переглядати та управляти заявками всіх

підрозділів (див. рис. 6.3). Він має повний доступ до інформації про всі заявки та може вживати відповідних дій щодо них, він може здійснювати різні дії щодо заявок, тобто прийняття або відхилення заявки.

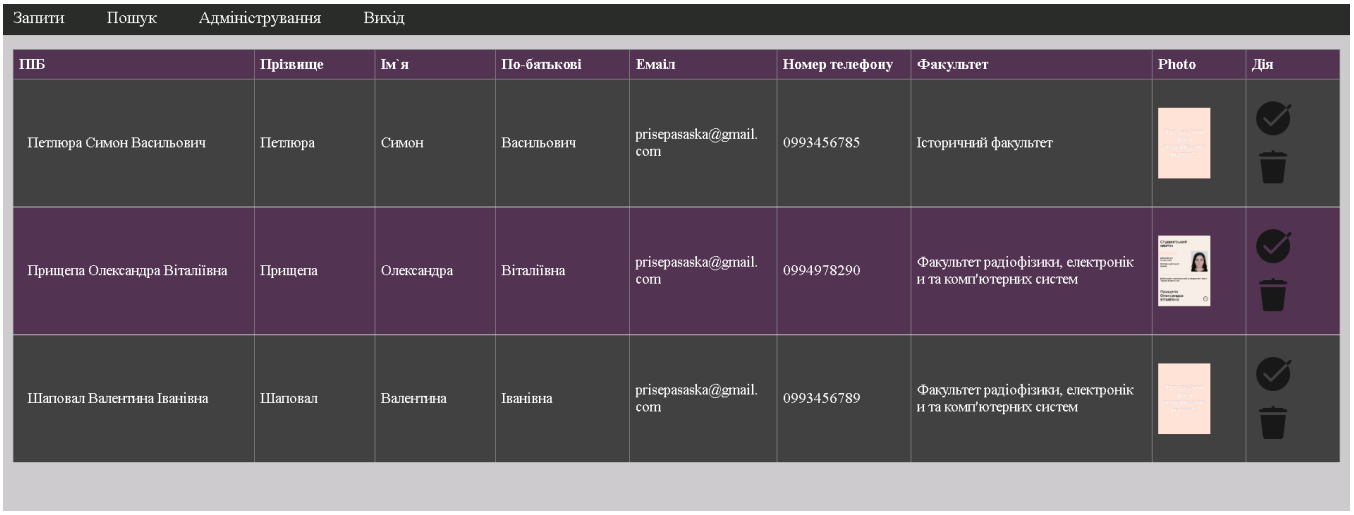


Рис. 6.3 Вебсторінка авторизації.

При відновленні логіну та паролю користувача, глобальний адміністратор або оператор має важливий обов’язок - ознайомитися з документом, який користувач надав для підтвердження своєї особистості, перш ніж приймати заявку. Ця додаткова перевірка документа має на меті забезпечити безпеку та достовірність процесу відновлення облікових даних.

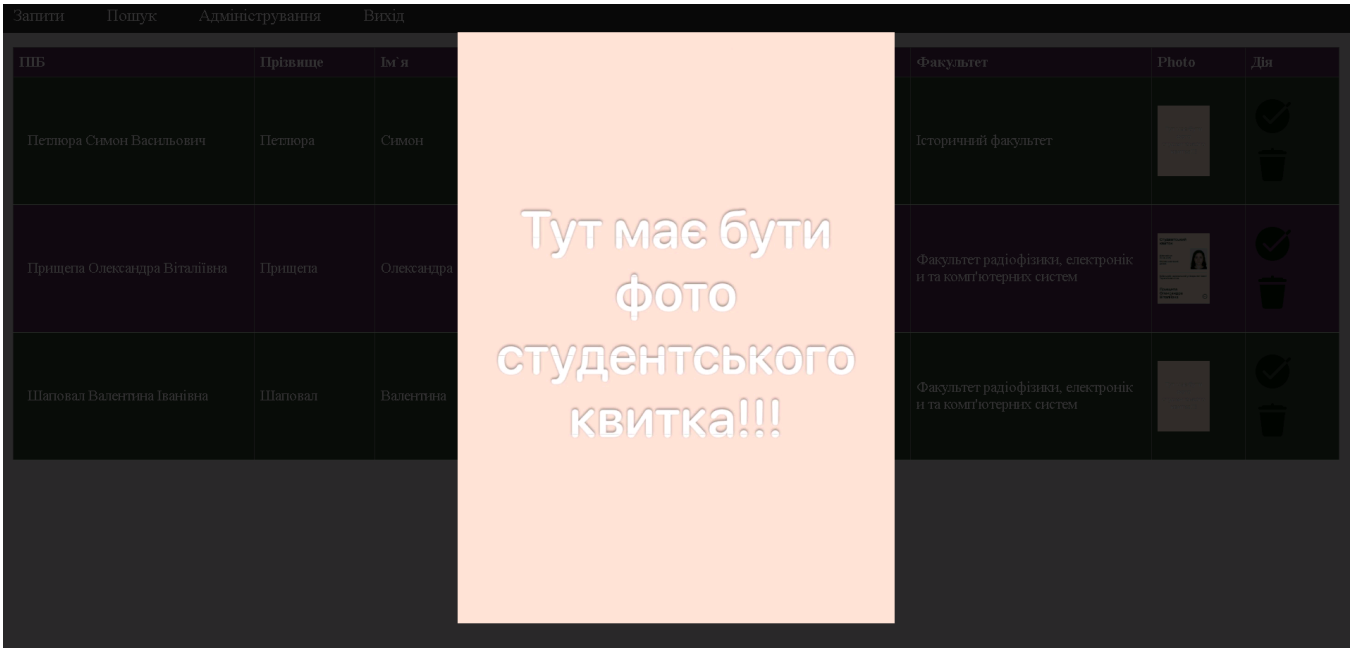


Рис. 6.4 Перегляд наданого користувачем документу.

Оператор повинен уважно переглянути наданий документ(див. рис. 6.4), який може бути фотографією або сканом особистого документа, такого як студентський квиток або посвідчення особи. Переглянути документ можна натиснувши на фотографію.

При натисканні на кнопку підтвердити користувачу, який лишив заявку відправляється лист, в якому вказується його логін та новий пароль(див. рис. 6.5).

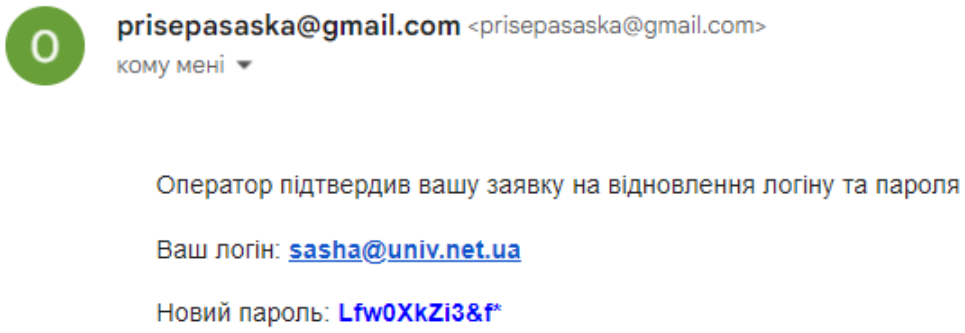


Рис. 6.5 Лист з логіном та паролем.

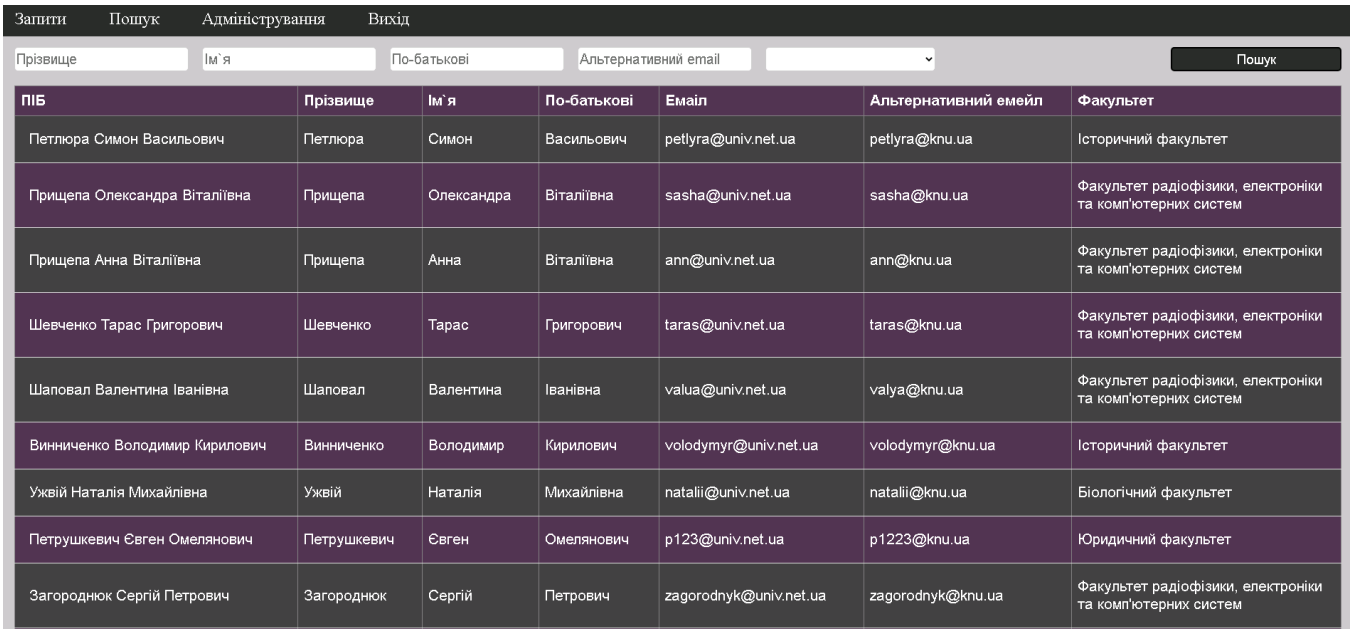
Глобальний адміністратор може бачити стан кожної заявки - чи вона була прийнята або відхилена. Для візуального позначення цих станів, використовується зміна кольору рядків заявок. Конкретно, прийнята заявка відображається з блідо-зеленим кольором, тоді як відхилена заявка має сірий колір(див. рис. 6.6).

Рис. 6.6 Відображення станів заявок.

Це візуальне позначення дозволяє глобальному адміністратору швидко розпізнавати статус кожної заявки, враховуючи її кольорове виділення. Такий підхід полегшує процес управління заявками та надає адміністратору зручний засіб візуалізації поточного стану системи.

6.3.2 Вебсторінка пошуку

Глобальний адміністратор має привілейований доступ до глобального пошуку (див. рис. 6.7), що дозволяє йому переглядати користувачів з усіх підрозділів системи, тобто всіх користувачів системи.



ПІБ	Прізвище	Ім`я	По-батькові	Емаіл	Альтернативний емейл	Факультет
Петлюра Симон Васильович	Петлюра	Симон	Васильович	petlyra@univ.net.ua	petlyra@knu.ua	Історичний факультет
Прищепа Олександра Віталіївна	Прищепа	Олександра	Віталіївна	sasha@univ.net.ua	sasha@knu.ua	Факультет радіофізики, електроніки та комп'ютерних систем
Прищепа Анна Віталіївна	Прищепа	Анна	Віталіївна	ann@univ.net.ua	ann@knu.ua	Факультет радіофізики, електроніки та комп'ютерних систем
Шевченко Тарас Григорович	Шевченко	Тарас	Григорович	taras@univ.net.ua	taras@knu.ua	Факультет радіофізики, електроніки та комп'ютерних систем
Шаповал Валентина Іванівна	Шаповал	Валентина	Іванівна	valua@univ.net.ua	valya@knu.ua	Факультет радіофізики, електроніки та комп'ютерних систем
Винниченко Володимир Кирилович	Винниченко	Володимир	Кирилович	volodymyr@univ.net.ua	volodymyr@knu.ua	Історичний факультет
Ужвій Наталія Михайлівна	Ужвій	Наталія	Михайлівна	natalii@univ.net.ua	natalii@knu.ua	Біологічний факультет
Петрушкевич Євген Омелянович	Петрушкевич	Євген	Омелянович	p123@univ.net.ua	p1223@knu.ua	Юридичний факультет
Загороднюк Сергій Петрович	Загороднюк	Сергій	Петрович	zagorodnyk@univ.net.ua	zagorodnyk@knu.ua	Факультет радіофізики, електроніки та комп'ютерних систем

Рис. 6.7 Вигляд вебсторінки пошуку

На сторінці глобального пошуку глобальний адміністратор може використовувати різні фільтри одразу для зручного та точного пошуку користувачів (див. рис. 3.8). Один з доступних фільтрів - факультет, що дозволяє адміністратору відфільтрувати користувачів за певним факультетом. Це особливо корисно оскільки база складається з великої кількісті користувачів, де зручне фільтрування допомагає знаходити необхідну інформацію швидко та ефективно.

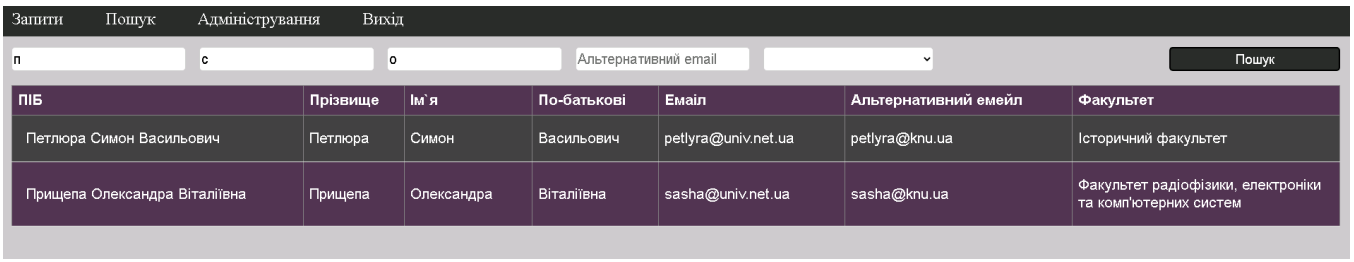


Рис. 6.8 Тестування роботи пошуку.

6.3.3 Вебсторінка керування операторами

Головна перевага у Глобального Адміністратора – це доступ до сторінки керування операторами. На цій сторінці глобальний адміністратор має можливість створити нового оператора(див рис. 6.9), який буде мати відповідальність за обробку заявок конкретного підрозділу або ще одного глобального адміністратора за потреби. Це дозволяє глобальному адміністратору розширити команду операторів для більш ефективного управління робочими процесами.

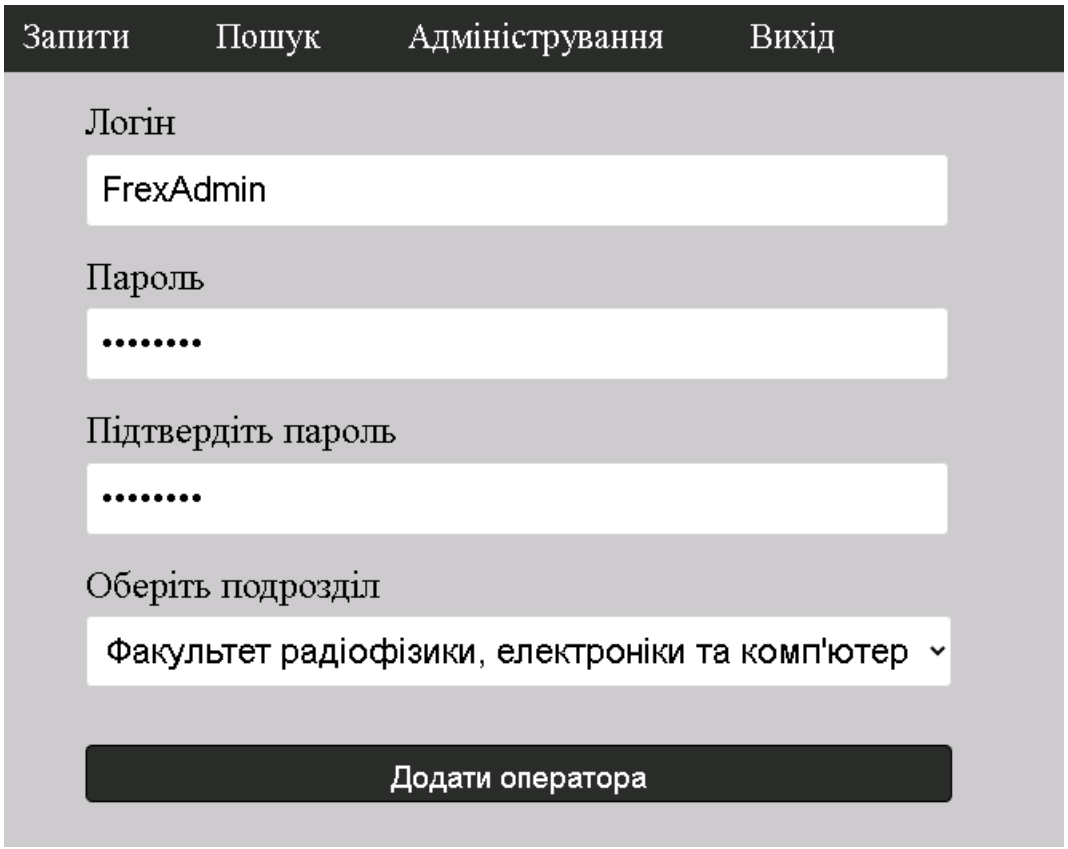


Рис 6.9 Створення нового оператора.

На сторінці адміністрування Глобальний адміністратор має доступ до перегляду існуючих операторів та їх відповідних даних, включаючи паролі. Переглянути пароль можна натиснувши на «Показати пароль» (див. рис. 6.10). Окрім перегляду, на сторінці також доступна функція видалення оператора.

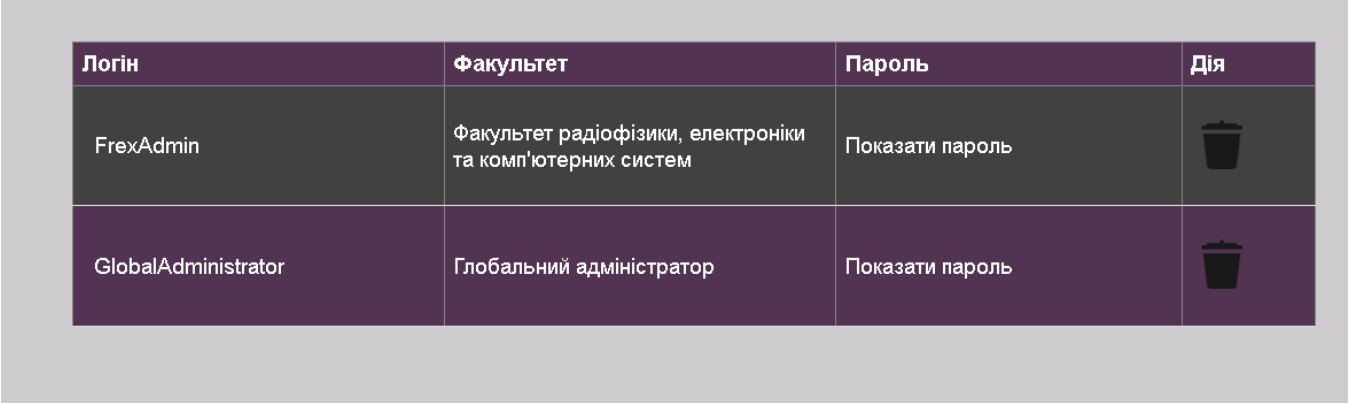


Рис. 6.10 Таблиця операторів.

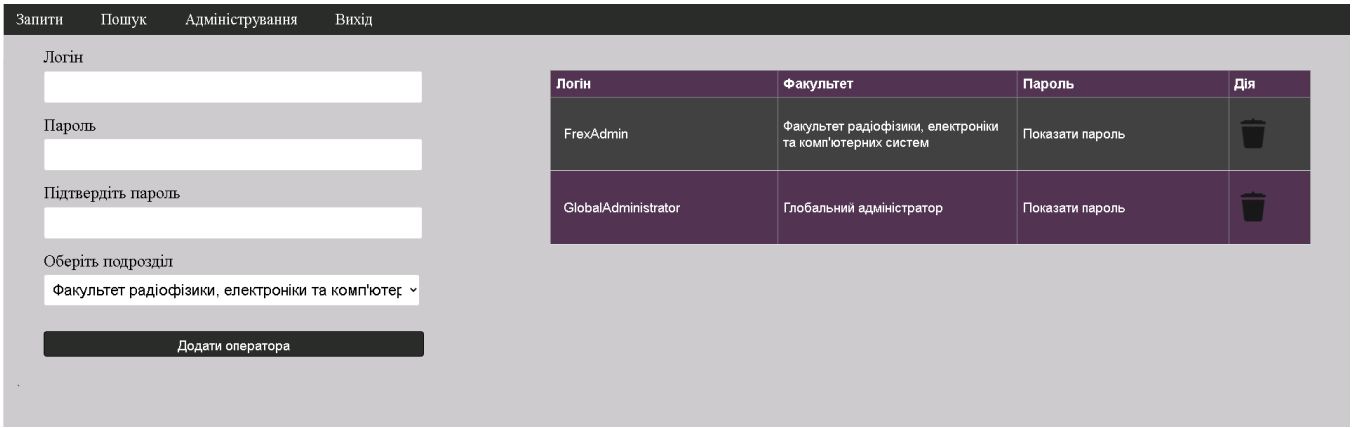


Рис. 6.11 Загальний вигляд вебсторінки адміністрування.

6.4 Функціонал Оператора

Для відображення функціоналу оператора авторизуємося під FrexAdmin – оператор відповідальний за підрозділ факультету РЕКС.

6.4.1 Вебсторінка запитів

Оператор, працює в рамках свого підрозділу та має доступ лише до заявок, які стосуються цього підрозділу та заявок, які знаходяться в прогресі (див. рис. 6.12). Він може переглядати інформацію про ці заявки, ознайомлюватися з документами, які

завантажили користувачі, та виконувати відповідні дії, такі як прийняття або відхилення заявок.

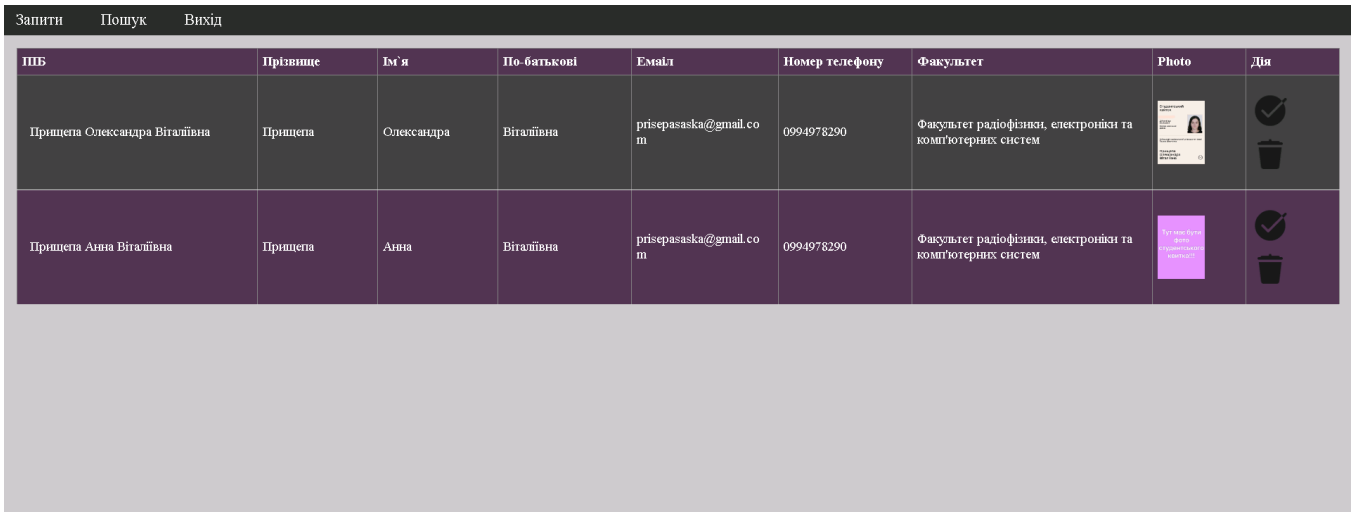


Рис. 3.12 Вебсторінка запитів оператора

6.4.2 Вебсторінка пошуку

Оператор FrexAdmin має доступ до пошуку користувачів тільки свого підрозділу(див. рис. 3.13). Оператор може використовувати фільтри для точнішого пошуку користувачів за прізвищем, іменем, по-батькові та альтернативною поштовою скринькою. Це дозволяє оператору знаходити необхідних користувачів зі свого підрозділу швидко та зручно.

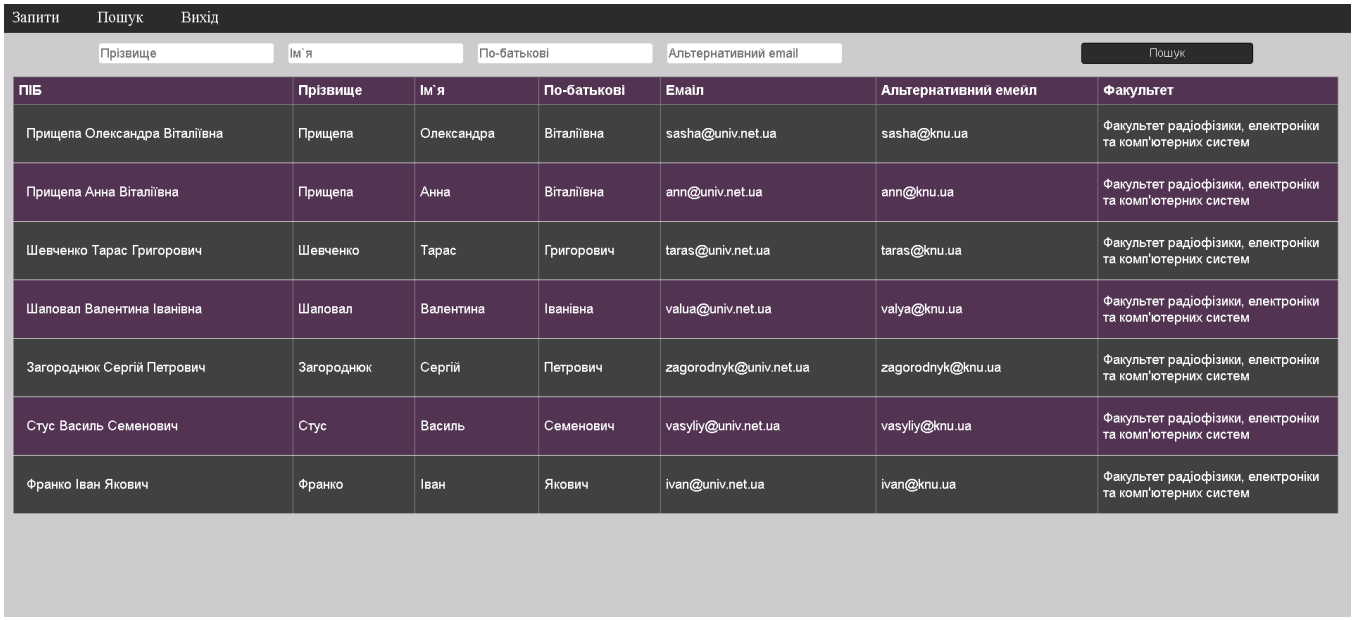


Рис. 6.13 Вебсторінка пошуку оператора.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено систему відновлення логіну та паролю до поштового серверу mail.univ.net.ua у вигляді вебсайту.

Механізм відновлення логіну та паролю є адаптованим під даний поштовий сервер і забезпечує можливість відновлення логіну у дистанційному форматі без особистої присутності на факультеті, а також надає можливість оператору підтвердити заявку на відновлення логіну також дистанційно.

Перш за все було проведено аналіз проблеми забування логінів та паролів, доведено, що дана проблема є поширеною та потребує нагального вирішення. У процесі розробки було також проведено аналіз вимог і визначено необхідні функціональні можливості системи. Були використані технології ASP.NET, Active Directory. Був розроблений відповідний функціонал подачі заявки на відновлення логіну та паролю, створено особистий кабінет оператора, додано функціонал для глобального адміністратора. В результаті відпрацювання всіх кроків користувач отримує лист на поштову скриньку зі вказаним логіном та новим паролем.

Подальший розвиток системи передбачає:

- Програмну інтеграцію на сервер університету;
- Тестування з використанням реальної бази даних;
- Впровадження в роботу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Password Security Report

https://www.cyclonis.com/report-83-percent-users-surveyed-use-same-password-multiple-sites/

2. 78% of people forgot a password in the past 90 days

https://www.helpnetsecurity.com/2019/12/11/forgot-password/

3. Воєнний стан в Україні

https://pon.org.ua/novyny/10333-voiennyi-stan-prodovzheno-shche-na-90-dib.html

4. Поради щодо відновлення облікового запису

https://support.google.com/accounts/answer/7682439?hl=uk

5. Не вдається ввійти в обліковий запис Google

https://support.google.com/accounts/troubleshooter/2402620?sjid=4092913496345635078-EU#ts=2402553

6. Втрата доступу до електронної адреси чи номеру телефону, пов’язаних з обліковим записом Instagram

https://help.instagram.com/358911864194456/?helpref=faq_content

7. A tour of the C# language
https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

8. ASP.NET overview https://learn.microsoft.com/en-us/aspnet/overview

9. Learn About LDAP https://ldap.com/learn-about-ldap/

10. Active Directory Domain Services Overview
https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

11. W3C Accessibility Standards Overview
https://www.w3.org/WAI/standards-guidelines/

ДОДАТКИ

ДОДАТОК А

```
public static class Constants
{
    public static string SenderEmail = "prisepasaska@gmail.com";
    public static string SenderPassword = "SomePassword";
    public static string SenderSmtpClient = "smtp.gmail.com";
    public static int SenderSmtpPort = 587;

    public static string AlternativeEmailPropertyName = "msExchExtensionAttribute45";

    public static string LDAPUserConnectionString = "LDAP://OU=MAIL-
Users,DC=PRISEPA,DC=NET,DC=UA";
    public static string LDAPOperatorConnectionString = "LDAP://OU=Operators,OU=MAIL-
Users,DC=PRISEPA,DC=NET,DC=UA";
    public static string LDAPRequestConnectionString =
"LDAP://OU=Request,DC=PRISEPA,DC=NET,DC=UA";
    public static string ImageFolderPath = @"C:\Images\";
    public static string[] Admins = new string[] { "Глобальний адміністратор" };
    public static string[] FacultyNames = new string[] {"",
            "Біологічний факультет",
            "Хімічний факультет",
            "Факультет кібернетики",
            "Географічний факультет",
            "Геологічний факультет",
            "Економічний факультет",
            "Історичний факультет",
            "Юридичний факультет",
            "Механіко-математичний факультет",
            "Інститут філології",
            "Філософський факультет",
            "Підготовчий факультет",
            "Фізичний факультет",
            "Радіофізичний факультет",
            "Факультет соціології та психології",
            "Інститут журналістики",
            "Військовий інститут",
            "Наукова бібліотека університету",
            "Червоний корпус",
            "Ректорат",
            "Лабораторія неруйнуючих досліджень",
            "Науково-дослідна частина",
            "Студмістечко університету",
            "Юнінет",
            "Інформаційно-обчислювальний центр",
            "Інститут адвокатури",
            "Астрономічна обсерваторія",
            "Клас червоного корпусу",
            "Ботанічний сад",
            "Фізико-математичний ліцей",
            "Фінансово-правовий колледж",
            "ІРУПЕМ Географічний ф-т",
            "Факультет соціології",
            "Факультет психології",
            "Інститут міжнародних відносин",
            "Інститут високих технологій",
            "Інститут післядипломної освіти",
            "Факультет інформаційних технологій",
            "Інститут психіатрії" };
    public static string EmailBody = "<div style=\"width: 100%;height: 200px;padding: 10px;
margin: 10px;background-color: black;color: white;\"><h3>Новий пароль</h3><p>Оператор
підтвердив вашу заявку на відновлення логіну та пароля</p><p class=\"p\">Ваш логін: <span
```

```
style=\"font-weight: 700;color: blue;\">LOGIN</span></p><p class=\"p\">Новий пароль: <span
style=\"font-weight: 700;color: blue;\">PASSWORD</span></p></div>";
}
```

ДОДАТОК Б

```
public static RequestAD[] GetAllRequests(string facultyName)
{

    var requests = new List<RequestAD>();
    DirectoryEntry directoryEntry = new
DirectoryEntry(Constants.LDAPRequestConnectionString);

    DirectorySearcher directorySearcher = new DirectorySearcher(directoryEntry);
    SearchResultCollection searchResultCollection = directorySearcher.FindAll();

    for (int i = 0; i < searchResultCollection.Count; i++)
    {

        SearchResult searchResult = searchResultCollection[i];
        DirectoryEntry userDirectoryEntry = searchResult.GetDirectoryEntry();
        var description = (string)userDirectoryEntry.Properties["description"].Value;

        if (!string.IsNullOrEmpty(facultyName)
            && !string.IsNullOrEmpty(description)
            && description != facultyName
            && !Constants.Admins.Contains(facultyName))
        {
            continue;
        }


        var request = new RequestAD
        {
            SecondName = (string)userDirectoryEntry.Properties["sn"].Value,
            Email = (string)userDirectoryEntry.Properties["mail"].Value,
            PhoneNumber = (string)userDirectoryEntry.Properties["telephoneNumber"].Value,
            FirstName = (string)userDirectoryEntry.Properties["givenName"].Value,
            Status = (string)userDirectoryEntry.Properties["st"].Value,
            FacultyName = description,
            LastName = (string)userDirectoryEntry.Properties["url"].Value,


        };
        var fileIdString = (string)userDirectoryEntry.Properties["displayName"].Value;
        if (string.IsNullOrEmpty(fileIdString) || !fileIdString.Contains("."))
        {
            continue;
        }
        request.Id = fileIdString.Split('.')[0];
        var extension = fileIdString.Split('.')[1];
        request.ImageType = GetMimeType(extension);

        var fileBytes = new byte[0];
        if (File.Exists(Constants.ImageFolderPath + fileIdString))
        {
            fileBytes = File.ReadAllBytes(Constants.ImageFolderPath + fileIdString);
        }
        if (fileBytes.Length == 0)
            continue;
        request.Image = Convert.ToBase64String(fileBytes);

        if (string.IsNullOrEmpty(request.SecondName) ||
            string.IsNullOrEmpty(request.Email) ||
            string.IsNullOrEmpty(request.PhoneNumber) ||
            string.IsNullOrEmpty(request.FirstName) ||
```

```
            string.IsNullOrEmpty(request.LastName) ||
            string.IsNullOrEmpty(request.FacultyName) ||
            string.IsNullOrEmpty(request.Status))
        {
            continue;
        }

        requests.Add(request);
    }

    directoryEntry.Close();

    return requests.ToArray();
}
```

ДОДАТОК В

```
private void FillData(RequestAD[] requestAds)
{
    var requestViewModels = new List<RequestViewModel>();

    foreach (var requestAd in requestAds.OrderBy(x => x.LastName))
    {
        var requestViewModel = new RequestViewModel();
        requestViewModel.SecondName = requestAd.SecondName;
        requestViewModel.PhoneNumber = requestAd.PhoneNumber;
        requestViewModel.Email = requestAd.Email;
        requestViewModel.FacultyName = requestAd.FacultyName;
        requestViewModel.LastName = requestAd.LastName;
        requestViewModel.FirstName = requestAd.FirstName;
        requestViewModel.ImageHtml = "<img class=\"requestImage\"
src=\"" + requestAd.Image + "\">";
        requestViewModel.ButtonsField = requestAd.Status == "inProgress" ?
            " <div value=\"" + requestAd.Status + "\" class=\"buttonContainer\"><input
class=\"id\" value=\"" + requestAd.Id + "\"></input><a class=\"tableBtn acceptBtn\"
onclick=\"acceptRequest(this)\"></a><a class=\"tableBtn removeBtn\"
onclick=\"removeRequest(this)\"></a></div>"
            : " <div value=\"" + requestAd.Status + "\" class=\"buttonContainer\"></div>";
        requestViewModels.Add(requestViewModel);

    }
    requestViewModels = requestViewModels
        .Where(x =>
            !string.IsNullOrEmpty(x.LastName)
            && !string.IsNullOrEmpty(x.FirstName)
            && !string.IsNullOrEmpty(x.SecondName)
            && !string.IsNullOrEmpty(x.FacultyName)
            && !string.IsNullOrEmpty(x.Email)
            && !string.IsNullOrEmpty(x.PhoneNumber))
        .ToList();

    GridView1.AutoGenerateColumns = false;
    GridView1.Columns.Clear();

    var columns = new CustomiseColumnModel[] {
                new CustomiseColumnModel
                {
                    ProperityName = "FullName",
                    DisplayName = "ПІБ",
                    Width = 18,
                },
                new CustomiseColumnModel
                {
                    ProperityName = "SecondName",
                    DisplayName = "Прізвище",
```

```
                    Width = 9,
                },
                 new CustomiseColumnModel
                {
                    ProperityName = "FirstName",
                    DisplayName = "Ім`я",
                     Width = 9,
                },
                new CustomiseColumnModel
                {
                    ProperityName = "LastName",
                    DisplayName = "По-батькові",
                    Width = 10,
                },
                 new CustomiseColumnModel
                {
                    ProperityName = "Email",
                    DisplayName = "Емаіл",
                     Width = 11,
                },
                new CustomiseColumnModel
                {
                    ProperityName = "PhoneNumber",
                    DisplayName = "Номер телефону",
                     Width = 10,
                },
                 new CustomiseColumnModel
                {
                    ProperityName = "FacultyName",
                    DisplayName = "Факультет",
                    Width = 18,
                },
                new CustomiseColumnModel
                {
                    ProperityName = "ImageHtml",
                    DisplayName = "Photo",
                    Width = 7,
                },
                new CustomiseColumnModel
                {
                    ProperityName = "ButtonsField",
                    DisplayName = "Дія",
                    Width = 7,
                }

            };
    foreach (var column in columns)
    {
        CustomiseColumn(column);
    }

    GridView1.DataSource = requestViewModels;
    GridView1.DataBind();
}
```

ДОДАТОК Г

```
protected void Button1_Click(object sender, EventArgs e)

{
    var label6 = this.FindControl("Label6") as Label;
    var label7 = this.FindControl("Label7") as Label;
    var label9 = this.FindControl("Label9") as Label;
    var label15 = this.FindControl("Label15") as Label;
    var label13 = this.FindControl("Label13") as Label;
    var label12 = this.FindControl("Label12") as Label;
```

```
var label14 = this.FindControl("Label14") as Label;
label9.Visible = false;
label7.Visible = false;
label6.Visible = false;
label15.Visible = false;
label13.Visible = false;
label12.Visible = false;
label14.Visible = false;

var textBox_SecondName = this.FindControl("TextBox_SecondName") as TextBox;
var textBox_FirstName = this.FindControl("TextBox_FirstName") as TextBox;
var textBox_LastName = this.FindControl("TextBox_LastName") as TextBox;
var textBox_Email = this.FindControl("TextBox_Email") as TextBox;
var textBox_PhoneNumber = this.FindControl("TextBox_PhoneNumber") as TextBox;

var success = true;
if (string.IsNullOrEmpty(textBox_PhoneNumber.Text))
{
    success = false;
    label7.Visible = true;
    label7.Text = "Це поле обов`язкове!";
}
if (string.IsNullOrEmpty(textBox_SecondName.Text))
{
    success = false;
    label6.Visible = true;
    label6.Text = "Це поле обов`язкове!";
}
if (string.IsNullOrEmpty(textBox_FirstName.Text))
{
    success = false;
    label13.Visible = true;
    label13.Text = "Це поле обов`язкове!";
}
if (string.IsNullOrEmpty(textBox_LastName.Text))
{
    success = false;
    label12.Visible = true;
    label12.Text = "Це поле обов`язкове!";
}
if (string.IsNullOrEmpty(textBox_Email.Text))
{
    success = false;
    label9.Visible = true;
    label9.Text = "Це поле обов`язкове!";
}
if (textBox_PhoneNumber.Text != null && textBox_PhoneNumber.Text.Length != 10)
{
    success = false;
    label7.Visible = true;
    label7.Text = "Невірна кількість цифр!";
}
if (DropDownList1.Text == "")
{
    success = false;
    label14.Visible = true;
    label14.Text = "Оберіть підрозділ!";
}
if (!FileUpload1.HasFile)
{
    success = false;
    label15.Visible = true;
    Label15.Text = "Фото обов`язкове!";
}
```

```
    if (!success)
        return;
    var saveRequest = new SaveRequestAd
    {
        FirstName = textBox_FirstName.Text,
        SecondName = textBox_SecondName.Text,
        LastName = textBox_LastName.Text,
        Email = textBox_Email.Text,
        Faculty = DropDownList1.Text,
        PhoneNumber = textBox_PhoneNumber.Text,
        Image = FileUpload1.FileBytes,
        ImageType = FileUpload1.PostedFile.ContentType
    };

    var result = RequestService.SaveRequest(saveRequest);
    if (result == true)
    {
        string resultMessage = "Заявку успішно створено!";
        ScriptManager.RegisterClientScriptBlock(this, this.GetType(), "alertMessage",
"alert('" + resultMessage + "'); setTimeout(function() { window.location.href =
'Application.aspx'; }, 2000);", true);
    }
    else
    {
        label15.Visible = true;
        Label15.Text = "Системна помилка!";
    }
```

ДОДАТОК Д

```
    public class ResultAccept
{
    public bool NotFoundRequest { get; set; }

    public bool MoreThenOneUser { get; set; }

    public bool NotFoundUser { get; set; }

    public bool ErrorWhileProcessing { get; set; }

    public bool RequestNotInProgress { get; set; }

    public bool ErrorWhileSaveLDAP { get; set; }

    public bool Success
    {
        get
        {
            return !(NotFoundRequest ||
                MoreThenOneUser ||
                ErrorWhileProcessing ||
                RequestNotInProgress ||
                ErrorWhileSaveLDAP ||
                NotFoundUser);
        }
    }
}
```

ДОДАТОК Е

```
        public static ResultAccept AcceptRequest(Guid requestId)
{

    var result = new ResultAccept();
    try
```

```
    {
        var request = GetAllRequests(Constants.Admins[0])
            .FirstOrDefault(x => x.Id == requestId.ToString());
        if (request == null)
        {
            result.NotFoundRequest = true;
            return result;
        }

        if (request.Status != "inProgress")
        {
            result.RequestNotInProgress = true;
            return result;
        }


        var users = ADCatalogueService.GetAllUsers(Constants.Admins[0])
            .Where(x => request.FirstName == x.FirstName
                && request.LastName == x.LastName
                && request.SecondName == x.SecondName)
                .ToArray();

        if (users.Length > 1)
        {
            result.MoreThenOneUser = true;
            return result;
        }

        if (users.Length == 0)
        {
            result.NotFoundUser = true;
            return result;
        }

        MailAddress from = new MailAddress(Constants.SenderEmail, Constants.SenderEmail);
        MailAddress to = new MailAddress(request.Email);
        MailMessage message = new MailMessage(from, to);

        var newPassword = ADCatalogueService.GenerateNewPassword();
        string messageBody = "";
        messageBody = "<html>";
        messageBody += "<html>";
        messageBody += "<head>";
        messageBody += "<meta charset=\"utf-8\">";
        messageBody += "</head>";
        messageBody += "<body>";
        messageBody = Constants.EmailBody.Replace("LOGIN",
users[0].UnivEmail).Replace("PASSWORD", newPassword);
        messageBody += "</body>";
        messageBody += "</html>";
        message.IsBodyHtml = true;
        message.BodyEncoding = Encoding.UTF8;
        message.Body = messageBody;

        SmtpClient smtp = new SmtpClient(Constants.SenderSmtpClient,
Constants.SenderSmtpPort);
        smtp.Credentials = new NetworkCredential(Constants.SenderEmail,
Constants.SenderPassword);
        smtp.EnableSsl = true;

        smtp.Send(message);
        result.ErrorWhileSaveLDAP = !ADCatalogueService.UpdateUser(users[0].UnivEmail,
newPassword, requestId.ToString());
        return result;
    }
```

```
    catch (Exception e)
    {
        result.ErrorWhileProcessing = true;
        return result;
    }
}
```

ДОДАТОК Є

```
protected void forIdButtonAcceptRequest_Click(object sender, EventArgs e)
{
    // var id = new Guid("64c48d3f-1f0d-40ea-866f-f66f2a8ecebc");
    var id = new Guid(forIdRequest.Text);

    var result = RequestService.AcceptRequest(id);
    var resultMessage = "";
    if (result.Success)
    {
        resultMessage = "Заявку успішно оброблено";
        ScriptManager.RegisterClientScriptBlock(this, this.GetType(), "alertMessage",
"alert('" + resultMessage + "'); setTimeout(function() { window.location.href =
'Request.aspx'; }, 2000);", true);

    }
    else
    {
        if (result.NotFoundUser)
            resultMessage = "Помилка: користувача не знайдено";
        if (result.MoreThenOneUser)
            resultMessage = "Помилка: знайдено більше одного користувача";
        if (result.NotFoundRequest)
            resultMessage = "Помилка: заявку не знайдено";
        if (result.ErrorWhileProcessing)
            resultMessage = "Сталася системна помилка";
        if (result.RequestNotInProgress)
            resultMessage = "Помилка: Неможливо підтвердити заяку, оскільки вона вже
підтверджена, або відхилена";
        if (result.ErrorWhileSaveLDAP)
            resultMessage = "Сталася помилка під час зберігання в базу";

        ScriptManager.RegisterClientScriptBlock(this, this.GetType(), "alertMessage",
"alert('" + resultMessage + "')", true);
    }
}
```

ДОДАТОК Ж

```
window.onload = function() {
    var requestButtons = document.querySelectorAll('.buttonContainer');

    requestButtons.forEach(function(element) {
        if (element.getAttribute('value') === 'removed')
        {
            var grandparent = element.parentNode.parentNode;
            grandparent.style.backgroundColor = '#9490a1';
        }
        else if (element.getAttribute('value') === 'accepted')
        {
            var grandparent = element.parentNode.parentNode;
            grandparent.style.backgroundColor = '#90a194';
        }
    });
```

```
    var operatorPasswords = document.querySelectorAll('.passwordContainer');
    operatorPasswords.forEach(function(element)
  {
        element.addEventListener('click', function() {
            var password = element.getAttribute('value');
            alert(password);
        });
    });

    const images = document.getElementsByClassName("requestImage");
    const imageArray = Array.from(images);

    imageArray.forEach(function(image) {
        image.addEventListener("click", function() {
            const fullscreenElement = document.createElement("div");
            fullscreenElement.className = "fullscreen";

            const fullscreenImage = document.createElement("img");
            fullscreenImage.src = image.src;
            fullscreenImage.className = "fullscreen-image";
            fullscreenElement.appendChild(fullscreenImage);
            document.body.appendChild(fullscreenElement);
            fullscreenElement.addEventListener("click", function() {
                document.body.removeChild(fullscreenElement);
            });
        });
    });
};


function acceptRequest(el)
{
    var parent = el.parentElement;
    debugger;
    var input = parent.querySelector('input');
    var id = input.value;
    document.getElementById('forIdRequest').value = id;
    document.getElementById("forIdButtonAcceptRequest").click();
};

function removeRequest(el)
{
    var parent = el.parentElement;
    var input = parent.querySelector('input');
    var id = input.value;

    document.getElementById('forIdRequest').value = id;
    document.getElementById("forIdButtonRemoveRequest").click();
    console.log(id);
};

function removeOperator(el)
{
    var parent = el.parentElement;
    var input = parent.querySelector('input');
    var id = input.value;
    document.getElementById('forOperatorId').value = id;
    document.getElementById("forIdButtonRemoveOperator").click();
    console.log(id);
};
```