Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Глоби Віталія Юрійовича*
(ПІБ)

академічної групи *121-18-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного додатку з побудови генеалогічного дерева за допомогою середовища розробки Unity*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.В.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-2* (група) *Глоби В.Ю.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного додатку з побудови генеалогічного дерева за допомогою середовища розробки Unity*

затверджена наказом ректора НТУ «ДП» від *18.05.2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Глоба В.Ю.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 62 с., 15 рис., 3 дод., 20 джерел.

Об’єкт розробки: програмний додаток генеалогічного дерева.

Мета кваліфікаційної роботи: метою є розробка програмного продукту, який надасть змогу користувачам створювати генеалогічні дерева за допомогою середовища розробки Unity.

У вступі розглядається аналіз сучасного стану проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведені обґрунтування актуальності теми та конкретизація постановки завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленого додатка, проведений підрахунок вартості роботи по створенню програми та розраховано час на її створення.

Практичне значення полягає у розробці програмного додатка, що надає можливість користувачам створювати генеалогічні дерева своїх родин, зберігати інформацію про своїх родичів у базі даних, створювати власний акаунт та здійснювати пошук людей у відкритій БД.

Актуальність даного програмного продукту визначається відсутністю подібних програм у мережі українського походження із простим та зрозумілим функціоналом, що оптимізують та спрощують дії щодо створення генеалогічного дерева, на відміну від його створення у фізичному вигляді, який легко загубити чи пошкодити; надають змогу зберігати інформацію про своїх родичів у зручному варіанті; розширюють функціонал звичайного генеалогічного дерева змогою додавання у нього зображень людини та різноманітної іншої корисної інформації.

Список ключових слів: DESKTOP, DESKTOP-ДОДАТОК, ПРОГРАМА, РОДИННЕ ДЕРЕВО, ГЕНЕАЛОЧІГНЕ ДЕРЕВО, БАЗА ДАНИХ, UNITY, КОМП’ЮТЕР.

ABSTRACT

Explanatory note: 62 p., 15 figs, 3 apps, 20 sources.

Object of development: family tree application

The purpose of the qualification work: development of a software application for building a family tree using the Unity development environment, which allows users to create family trees and store information about them in a database, as well as view other user's family trees if they are publicly available.

The introduction considers the analysis of the current state of the problem, specifies the purpose of the qualication work and the field of its application, provides justification for the relevance of the topic and the specification of the problem.

In the first section carries out the analysis of the subject area, determines the relevance of the task and the dedication of the development, creates task statemant, the software and hardware requirements of the product, specifies technologies and tools for developmet.

In the second section analyzes the existing solutions, chose the platform for developmet, creates design and finish development of the product, describes the algorithm, structure and architecture solutions in the system, defines the input and output data, describes characteristics of the technical resources used, describes how to run a program, features of user interaction, differences between serve and client parts of product.

The economic sections determines the complexity of the developed application, calculates the cost of work to create a program and defines the time for its creation.

The practical significance is the development of a software application that allows users to create family trees of their families, store information about their relatives in a database, create your own account and search for people in an open database.

The relevance of this software product is determined by the lack of similar programs in the network of Ukrainian origin with simple and clear functionality that optimizes and simplifies the process of creating a family tree, in contrast to its creation in physical form, which is easy to lose or damage; allow you to store information about your relatives in a convenient way; expand the functionality of an ordinary family tree by adding images of a person and a variety of other useful information.

List of keywords: DESKTOP, DESKTOP-APPLICATION, PROGRAM, FAMILY TREE, GENEALOGY TREE, DATABASE, UNITY, COMPUTER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СУБД – система управління базами даних;

ООП – об’єктно орієнтоване програмування;

ІСР – інтегрована середа розробки;

СР – середовище розробки.

ВСТУП

Тематика даної клаліфікаційної роботи присвячена розробці програмного додатка з побудови генеалогічного дерева на базі середовища розробки Unity.

Метою даної кваліфікаційної роботи є вивчення інструментальниих засобів об'єктно орієнтованої мови програмування C#, середовища розробки Unity та мови структурованих запитів SQL для розробки програмного забезпечення, яке надасть змогу користувачам дізнатися більше про свій родовід.

З давніх давен існує традиція шанувати та пам'ятати своїх предків, але з кожним новим поколінням люди все більше про неї забувають та втрачають розуміння нащо вона взагалі потрібна, а отже перестають цікавитися своїм корінням. Середньостатистична людина може назвати три або чотири покоління свого роду. Для чого ж нам потрібно знати свою генеалогію?

По-перше, це звичайна людська зацікавленість. Багатьом з нас хотілось би дізнатися, чим займалися наші предкі, які у них були ремесла, традиції, таланти або ж чи є серед них якісь знамениті та видатні люди. Вивчаючи цю інформацію це приносить не тільки захоплення та задоволення, але й навчає нас історії.

По-друге, знання власної генеалогії несе практичне значення, адже ми можемо дізнатися про схильність нашого роду до певних захворювань та виходячи з цього вести профілактику цих захворювань, чи проаналізувавши рід занять наших предків визначитися зі своїм професійним назначенням, або ж отримати громадянство, на під ставі коренів своїх предків.

Існує ще багато факторів пов'язаних з духовним зв'язком та цінностями зі своїми предками, вирішення сімейних проблем, психологічний та релігійний фактори.

На жаль, раніше у простих людей не було можловості вести свої родоводи. Відомості про пращурів передавалися з уст в уста та через деякий час за певних обставин просто безслідно губилися. Наш світ з кожним днем прогресує і зараз у нас з'явилося багато нових технологій, які полегшують нам життя та приносять нові можливості, про які ми раніше і уявити собі не могли.

Використовуючи ці можливості і технології можна створити тренд на знання свого родоводу, розробивши програмне забезпечення, яке буде містити всю інформацію про рід людини в одному місці.

Результатом виконання даної роботи буде програма, яка буде спрощувати та приносити різноманіття у створенні родового дерева, пришвидшить пошук довідок про людину та буде містити велику БД з різноманітною інформацією.

Саме тому завданням кваліфікаційної роботи було обрано “Розробка desktop додатка генеалогічного дерева за допомогою середовища розробки Unity”. Завдання даної кваліфікаційної роботи та об’єкт його діяльності безпосередньо пов’язані з напрямом підготовки, відповідає узагальненій тематиці кваліфікаційних робіт та переліку зазначених навичок та компетенцій, якими повинен володіти фахівець напряму “Інженерія програмного забезпечення”.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Генеалогічне дерево або сімейне – це схематичне представлення родових зв'язків, представлене переліком нащадків та їх подружжя у вигляді деревоподібної структури, із суворим дотриманням певних правил. Дані про родичів, як правило, зводяться в об'ємну таблицю з безліччю розбіжностей, схожих на гілки та коріння – саме тому її називають сімейним деревом. Спочатку сімейні дерева були представлені у вигляді таблиць, зверху таблиці розташовувалися пращури, а знизу нащадки. Сучасний же вид генеалогічних дерев пійшов від дерева Єссея, на якому зображений родовід Ісуса Христа у вигляді дерева, що проростає із постаті Єссея – батька царя Давида [6]. У генеалогії сімейне дерево є складовим поняттям.

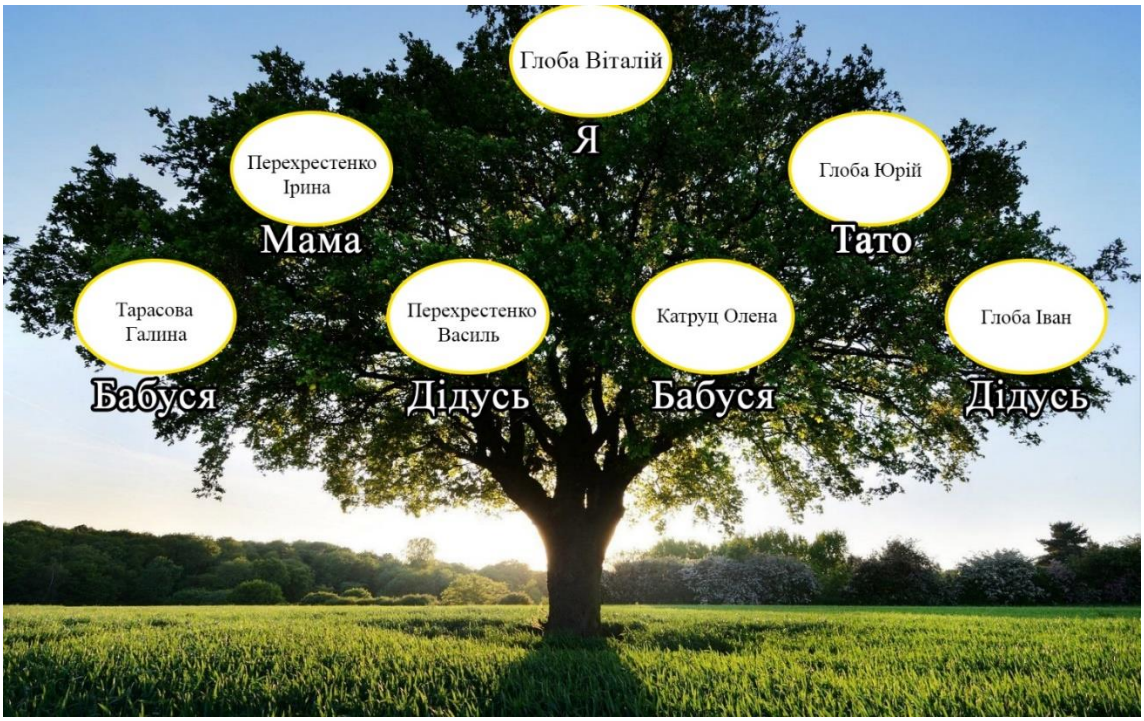

Рис. 1.1. Приклад типового генеалогічного дерева

Розрізняють два види генеалогічних дерев: висхідні та низхідні. Висхідне сімейне дерево – це дерево пращурів, коли старші покоління розташовуються знизу дерева, а молодші зверху [3], як зображено на рис. 1.1. Низхідне дерево – це дерево нащадків і старші покоління розташовуються вище на дереві, ніж молодші.

Хоча генеалогічні дерева зображуються у вигляді дерев, сімейні відносини в загальному випадку не утворюють дерева в строгому сенсі, оскільки далекі родичі можуть спаровуватися. Отже, людина може мати спільного предка як з боку матері, так і з боку батька. Однак, оскільки батько повинен народитися раніше за свою дитину, людина не може бути своїм власним предком, і отже, петель немає. У зв'язку з цим родовід утворює орієнтований ациклічний граф. Тим не менш, графіки, що зображають матрилінійне походження (стосунки матері та дочки) та батьківське походження (стосунки батька та сина), утворюють дерева [3]. Припускаючи відсутність загального предка, карта родоводу є ідеальним бінарним деревом, тому що у кожної людини одна мати і один батько, таким чином вони мають регулярну структуру. Дерево нащадків, з іншого боку, немає регулярної структури, оскільки в людини може бути будь-яка кількість дітей чи взагалі жодного [4,5].

Найдавнішим задокументованим сімейним деревом є дерево сім’ї Конг, які є нащадками Конфуція. У комітеті зі складання генеалогії Конфуція зареєстровано близько 3 мільйонів членів родини. У даний час сім’я Конг є 83-м поколінням роду Конфуція і вона занесена у Книгу рекордів Гіннеса. Існують і більш давніші родоводи, але за різних причин учені не можуть підтвердити їх достовірність [1,2].

Серед розробників ПЗ ідея програми генеалогічного дерева не дуже поширена та популярна, оскільки на стадії розробки можуть виникати проблеми із візуальною частиною програми, адже при створенні великих родових дерев, які будуть містити десятки або ж сотні людей, виникає проблема із тим, як саме розташувати всіх їх на екрані, аби звичайному користувачеві були зрозумілі усі зв’язки у дереві та він не заплутався.

Також складно монетизувати контент у подібного роду програм, тому половина із них або є платними, або ж містять безкоштовні підписки з урізаними можливостями.

Ще одною проблемою аналогічних програм є устаріле та просте оформлення. Із одного боку для когось це може бути плюсом, бо немає візуального нагромадження і нічого не відволікає, з іншого ж боку, це може відбити бажання користуватися такою програмою ще на стадії вибору, оскільки гарна картинка завжди приємніша людському оку.

1.2. Призначення розробки та галузь застосування

Повною назвою розробленої програми для кваліфікаційної роботи є “Розробка програмного додатка з побудови генеалогічного дерева за допомогою середовища розробки Unity”.

Ключові слова та термінологія:

Unity – безкоштовна кросплатформова середа розробки, розроблена американською компанією Unity Technologies. Дозволяє користувачам розробляти на своїй основі різноманітні програми, ігри та візуалізовувати математичні моделі. Найбільшої популярності набула серед розробників ігор, використовується як велики компаніями так і фріланс розробниками. Написана на мовах C++, C#, для написання скриптів використовує C#. Редактор Unity має простий Drag&Drop інтерфейс, що складається із різних вікон, завдяки чому можна проводити налагодження програми прямо в редакторі.

Desktop-додаток – програма, яка локально встановлюється на комп’ютер користувача за допомогою спеціального інсталятора і працює під управлінням ОС. Особливість такої програми в тому, що вона працює локально та автономно, у деяких випадках не потребує підключення до мережі Інтернет.

C# – об’єктно-орієнтована мова програмування зі сторогою статичною типізацією, була створена і на цей час підтримується компанією Microsoft. Підходить для розробки ігор, веб-сервісів, мобільних та desktop-додатків.

SQL – мова запитів, за допомогою якої у СУБД можна комунікувати із даними, які знаходяться у БД, тобто додавати, видаляти, редагувати та інше.

Розроблена програма може використовуватися в різноманітних галузях медицини, таких як, патологія, дерматологія, гематологія та інші, адже у родоводах можна простежувати ознаки патологічних захворювання, які передаються спадково. У школах в молодших класах часто дають завдання на створення сімейних дерев, саме тому програму можна впровадити у навчальні заклади аби зацікавити учнів досліджувати свій родовід. У більшій мірі програма буде використовуватися індивідуально користувачами у власних цілях та інтересах.

Призначення розробки – створення програмного продукту, який надасть змогу користувачам створювати родові дерева, наповнюючи тим самим велику базу даних, яка у майбутньому може бути корисною в різноманітних галузях.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

– освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;

– завдання на кваліфікаційну роботу на тему «Розробка програмного додатку з побудови генеалогічного дерева за допомогою середовища розробки Unity».

1.4. Постановка завдання

Завданням даної кваліфікаційної роботи є написання програмного продукту, який облегшить і надасть змогу створювати генеалогічне дерево, відносно створення физічного гееналогічного дерева, та шукати інформацію про певну людину, якщо вона є у відкритому доступі, за допомогою середовища розробки Unity.

Основними характеристиками розробки програми повинні бути:

- зареєструвати власний акаунт;
- можливість користувачеві самостійно створювати генеалогічне дерево;
- зберігати, видаляти та редагувати інформацію у своєму родоводі;
- продивлятися інформацію у відкритому доступі.

Умови, які потрібно виконати для вирішення поставленої задачі:

- вибір мови програмування та середовища розробки;
- детальне ознайомлення із галуззю розробки програмного продукту;
- ознайомлення із аналогами програми;
- структурування та оптимізація процесів розробки;
- написання програмного коду.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт розробки матиме чотири види доступу до свого функціонала:

- гість;
- користувач;
- адміністратор.

На рівні доступу “Гість” повинен бути реалізований наступний функціонал:

– можливість створити новий акаунт, або авторизуватися

На рівні доступу “Користувач” повинен бути реалізований наступний функціонал:

– можливість виходу із акаунта;
– можливість підвищити статус акаунта до “ Користувач Premium”;
– створювати генеалогічне дерево;
– видаляти генеалогічне дерево;
– редагувати генеалогічне дерево;
– зберігати генеалогічне дерево;
– можливість пошуку людини у БД.

На рівні доступу “Адміністратор” повинен бути реалізований наступний функціонал:

– можливість виходу із акаунта;
– створювати генеалогічне дерево;
– видаляти генеалогічне дерево;
– редагувати генеалогічне дерево;
– зберігати генеалогічне дерево;
– можливість продивлятися інформацію, яку інші користувачі надали у відкритий доступ;
– видаляти та редагувати генеалогічні дерева інших користувачі;
– блокувати користувачів.

Інтерфейс та оформлення програми повинні бути інтуїтивно зрозумілими користувачеві.

1.5.2. Вимоги до інформаційної безпеки

Основними вимогами до інформаційної безпеки є:

– конфіденційність даних;
– цілісність даних;
– безпечність передачі даних до БД
– використання надійного пароля.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для підтримки та коректної роботи із програмою рекомендуються такі параметри технічного засобу:

– Процесор : багатоядерний Intel або AMD із підтримкою набору інструкцій SSE.
– Відеокарта: з підтримкою DX10, DX11 або DX12.
– Оперативна пам’ять: 4 ГБ.
– ОС: Windows 10, 64-розрядної версії.

1.5.4. Вимоги до інформаційної та програмної сумісності

Розроблюваний програмний додаток локально встановлюється на комп'ютер користувача і не потребує встановлення інших програм. Аби використовувати програмний продукт потрібно мати комп'ютер із встановленою на ньому операційною системою Windows 10 64-х розрядної версії та доступ до мережі Інтернет, аби завантажити пограму через будь-який браузер.

Основною мовою програмування була C#, структура програми була розроблена у середовищі розробки Unity, база даних була розроблена у середовищі MySQL.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом виконання даної кваліфікаційної роботи повинен бути додаток, за допомогою якого користувач зможе створити генеалогічне дерево своєї родини.

Додаток буде сумісний тільки з ОС Windows 10 та буде містити 4 рівні доступу до функціоналу:

- рівень доступу “Гість”;
- рівень доступу “Користувач”;
- рівень доступу “Адміністратор”.

На рівні доступу “Гість” повинен бути реалізований наступний функціонал:

- можливість створити новий акаунт, або авторизуватися.

На рівні доступу “Користувач” повинен бути реалізований наступний функціонал:

- можливість виходу із акаунта;
- можливість підвищити статус акаунта до “ Користувач Premium”;
- створювати генеалогічне дерево;
- видаляти генеалогічне дерево;
- редагувати генеалогічне дерево;
- зберігати генеалогічне дерево.

На рівні доступу “Адміністратор” повинен бути реалізований наступний функціонал:

- можливість виходу із акаунта;
- створювати генеалогічне дерево;
- видаляти генеалогічне дерево;
- редагувати генеалогічне дерево;

– зберігати генеалогічне дерево;
– можливість продивлятися інформацію, яку інші користувачі надали у відкритий доступ;
– видаляти та редагувати генеалогічні дерева інших користувачі;
– блокувати користувачів.

2.2. Опис застосованих математичних методів

При проектуванні та розробці даного додатку використовувалися тільки прості математичні дії. Ніякі математичні методи не були використані.

2.3. Опис використаної архітектури та шаблонів проєктування

Структура розробленого додатку є схожою зі структурою виду Model-View-Controller (MVC, “Модель–Вид–Контролер” або ж “Модель-Представлення-Контролер”). MVC структура являє собою схему розподілення даних додатку і керуючої логіки на три окремі компоненти: модель, вид, контролер [7]. Приклад типової MVC структури можна побачити на рис.2.1.

Рис. 2.1. Типове представлення структури MVC

Модель – це динамічна структура даних програми, незалежна від інтерфейсу користувача. Являється цетральним компонентом структури. Вона безпосередньо керує даними, логікою та правилами програми, отримуючи зміни, які хоче внести користувач у модель, від контролера [7,8].

Вид (Представлення) – будь-яке візуальне представлення інформації, яке надходить від моделі, наприклад: схеми, рисунки, таблиці і т.д.. Можливе існування декількох видів представлення інформації, але контролер обирай той, який більш підходить у даній ситуації [7,8].

Контролер – отримує набір даних від користувача, які він хоче змінити, і потім перетворює ці дані на команди для моделі даних або для представлення. У залежності від потребностей контролер може перевіряти вхідні дані, а вже потім передати їх моделі чи виду [7,8].

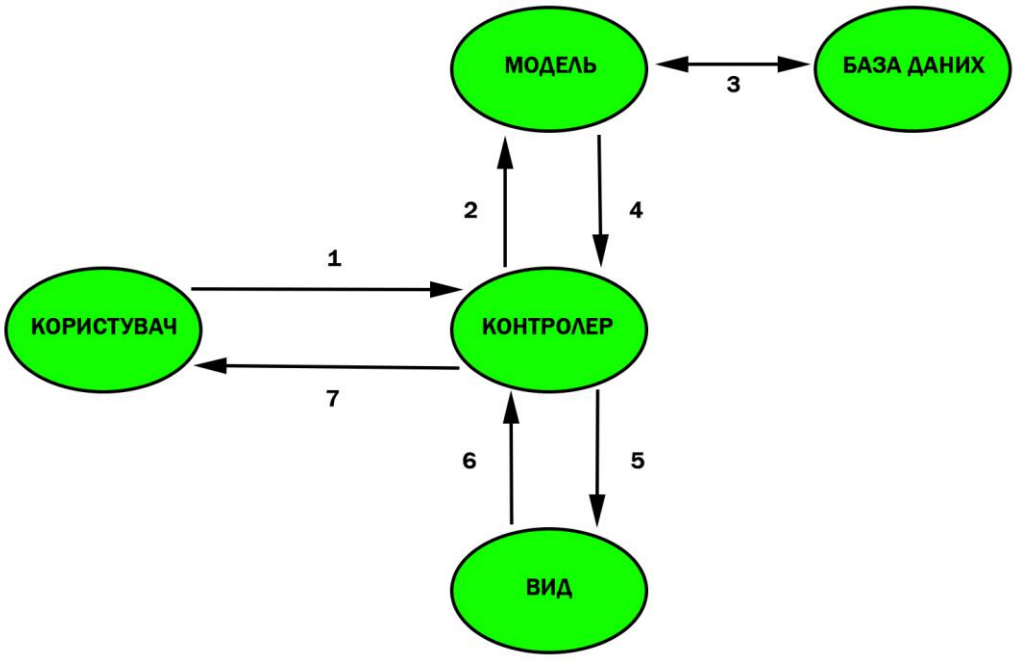

Рис. 2.2. Структура розробленого додатка

Розглянемо рис.2.2., на якому зображена структура розробленого додатку для кваліфікаційної роботи. Користувач даного додатка вносить змінні, тобто створює генеалогічне дерево, або заносить туди дані, які відправляються на

обробку до контролера (1), контролер перевіряє запит користувача і після цього визиває модель аби змінити дані (2), внесені зміни заносяться до БД, яка звітує моделі, що зміни пройшли успішно (3), модель, маючи змінену інформацію передає її контролеру (4), аби той підібрав підходячий вигляд для ної інформації (5), і після цього виводить дану інформацію користувачу , коли користувач закінчує заносити нові дані.

2.4. Опис використаних технологій та мов програмування

При розробці даного додатка були використані наступні технології та мови програмування:

- мова програмування C#;
- структурована мова запитів SQL;
- бібліотека MySQL Connector/Net;

Бібліотека MySQL Connector/Net забезпечує підключення додатка з СУБД MySQL для подальшої передачі даних до БД. Вона є дуже зручною у використанні, містить усі потрібні методи для взаємодії розробника з базою даних, які можна додати в окремий клас і використовувати у себе при розробці. Даний конектор забезпечує надійну та швидку передачу даних між додатком і БД.

C# – це сучасна мова програмування, розроблена компанією Microsoft, яка користується великим попитом і входить у топ найпопулрніших мов програмування уже на протязі багатьох років. Вона є об’єктно-орієнтованою та безпечною для програмування, що дозволяє створювати різноманітні додатки які працюють на платформі Microsoft .NET [10]. Мова C# розроблена на структурі C++ і містить у собі функції подібних до тих, що є у мові Java та JavaScript, тому розробникам, що володіють цими мовами C# буде легка в освоєнні.

C# був розроблений для полегшення обміну інформації та різноматніних послуг в Інтернеті. Він використовує XML – Extensible Markup Languege і SOAP

– Simple Object Acccess Protocol, що спрощує процес програмування для розробників, адже завдяки ним можна отримати доступ до певного елемента програмування без написання розробником додаткового коду для кожного свого кроку, що підшвидшить роботу та зробить її менш дорогою. Компанія Microsoft співпрацювала з компанією ECMA, що займається стандартизацією інформаційних технологій, аби створити власний стандарт для мови C#, що спонукало б інші компанії розробляти свої версії мови [13]. На данний момент відомо про 3 незалежні реалізації мови C#, що базуються на її специфікації:

- Mono – комп'ютерна платформа, до основних компонентів якої входять компілятор C#, віртуальна машина для загальномовної інфраструктури та бібліотеки основних класів. Вони мають стандартизацію ECMA-334 і ECMA-335, що дозволило фреймворку надати безкоштовну машину для віртуалізації CLI із вихідним кодом, який є у відкритому доступі, що відповідає стандартам.
- dotGNU є набором інструментальних засобів, що проєктувався як аналог Microsoft .NET, та ціллю якого є надання розробникам можливості розробляти проєкти на різних мовах програмування, але набір мов був обмежений і підтримуються тільки C#, C, C++ та Visual Basic .NET. DotGNU включає в себе 3 основні компоненти: dotGNU Portable .NET, phpGroupWare та DGEE.
- Portable.NET – вільне ПЗ, яке є частиною dotGNU, що надає переносний набір інструментів та системи виконання для Common Language Infrastructure (фреймворки загальномовної структури).

Серед причин, які роблять мову C# популярною у розробників можна виділити такі:

- сучасна мова загального призначення;
- легка у вивченні;
- об'єктно-орієнтована;
- компілюється на різних комп'ютерних платформах;
- структурована та компонентно-орієнтована;

– являється частиною .NET Framework.

Ключовими організаційними концепціями C# є програми, типи, збірки, простори імен та члени. Програми оголошують типи, у яких мостяться члени, і ці типи організовуються у простори імен. До типів можуть відноситись класи, інтерфейси і структури, а до членів – методи, властивості та поля. Під час компілювання програми, компілятор фізично збирає її в збірку. Зазвичай вона має одне із розширенн: .exe або ж .dll, якщо .exe, то збірка реалізує програму, а .dll реалізує бібліотеку [15].

Нижче наведені важливі функції, які підтримує C#:

– автоматичний збір сміття;
– bool-еві умови;
– стандартна бібліотека;
– індексатори;
– умовна компіляція;
– версійна збірка;
– властивості та події;
– делегати та управління подіями;
– проста багатопоточність;
– LINQ і лямбда-вирази;
– інтеграція з Windows.

Найкращими ІСР для програмування на C# є:

– Visual Studio;
– Project Rider;
– Eclipse;
– Visual Studio Code;
– MonoDevelop;
– Code::Blocks.

При розробці додатка для кваліфікаційної роботи було обране інтегроване середовище розробки Visual Studio. Це ІСР є продуктом розробки компанії

Microsoft та є найпопулярнішим серед розробників, які обрали мову програмування C# для своїх проєктів.

SQL (Structured Query Language) – структурована мова в програмуванні, що предназначена для управління даними в реляйних базах даних. Простими словами можна сказати, що SQL допомагає розробнику комунікувати з даними, які знаходяться у БД, тобто продивлятися, видаляти, додавати, редагувати та інше [9]. За допомогою SQL дуже зручно працювати зі структурованими даними, тобто такими даними, які мають зв'язки між деякими сутностями або змінними.

Історія SQL розпочинається на початку 70-х років XX-го століття, коли компанії IBM стало відомо про реляційну модель Едгарда Кодда. Тоді вона називалася SEQUEL (Structured English Query Language) і призначалася для комунікування даними в оригінальній системі куревання БД IBM. Після цього була розроблена SQUARE – перший приклад мови релаційної БД, але вона була складною у використанні, тому в 1973-му році, після переїзду головного департамену компанії у дослідницький центр була розпочата робота над продовженням мови SQUARE, яка була переіменована у SEQUAL, пізніше і ця назва була змінена на відому нам уже SQL. У 1986 році за допомогою організацій ANSI та ISO, що займаються стандартизацією технологій абревіатура SQL стала для визначення структурованої мови програмування [17].

На відміну від своїх попередників серед API читання-запису, таких як ISAM та VSAM, SQL має дві великі переваги над ними:

– По-перше, вона має концепцію, при якій розробник може отримати доступ до багатьох записів у БД лише за допомогою однієї команди, а не прописувати для кожного запису окрему команду.

– По-друге, не потрібно описувати як саме досягти цих записів, наприклад з індексом або ж без нього.

Мову SQL можно розподілити на декілька елементів:

- речення, які є складовими поняттями запитів, але не є обов'язковими;
- вирази, які можуть складатися із таблиць та скалярних значень;
- предикати, які переносять значення істинності;

– запити, які є найважливішим елементом в SQL;

– заяви, які мають постійний вплив на схеми та дані.

Також стандартним елементом граматики синитаксису мови SQL є крапка з комою в операціях, хоч вона і є не обов’язковою на кожній платформі [14,18]. Пробіли ігноруються у запитах SQL, що полегшує форматування та написання коду.

Безперечними плюсами мови SQL є:

– швидкість обробки запитів та доступу до даних;

– стандартизований синтаксис мови SQL, який використовується у різніх СУБД;

– стандартизація мови від ANSI та ISO;

– гарна сумісність із реляційними БД.

Але через декілька причин можуть виникати несумістність систем баз даних:

– розміри стандарту SQL, через це не всі реалізатори можуть підтримувати увесь стандарт;

– стандарт не описує поведінку БД в окремих областях, через що реалізація сама повинна вирішувати як поводитися.

Серед найпопулярніших СУБД SQL можна виділити такі:

– Microsoft SQL Server;

– Oracle SQL Developer;

– MySQL;

– PosgreSQL;

– MongoDB;

– DB2;

– Misrosoft Access.

2.5. Опис структури програми та алгоритмів її функціонування

Для опису структури програми та алгоритмів її функціонування були використані UML діаграми.

UML (Unified Modeling Language) – мова для графічного опису програмного продукту, його процесів, структур та інших елементів.

Інструментарій графічної мови UML складається із набору різних модельних елементів таких як геометричні фігури, стрілки, таблиці, написи значки та інше для позначення окремих структур у певному процесі або програмному додатку.

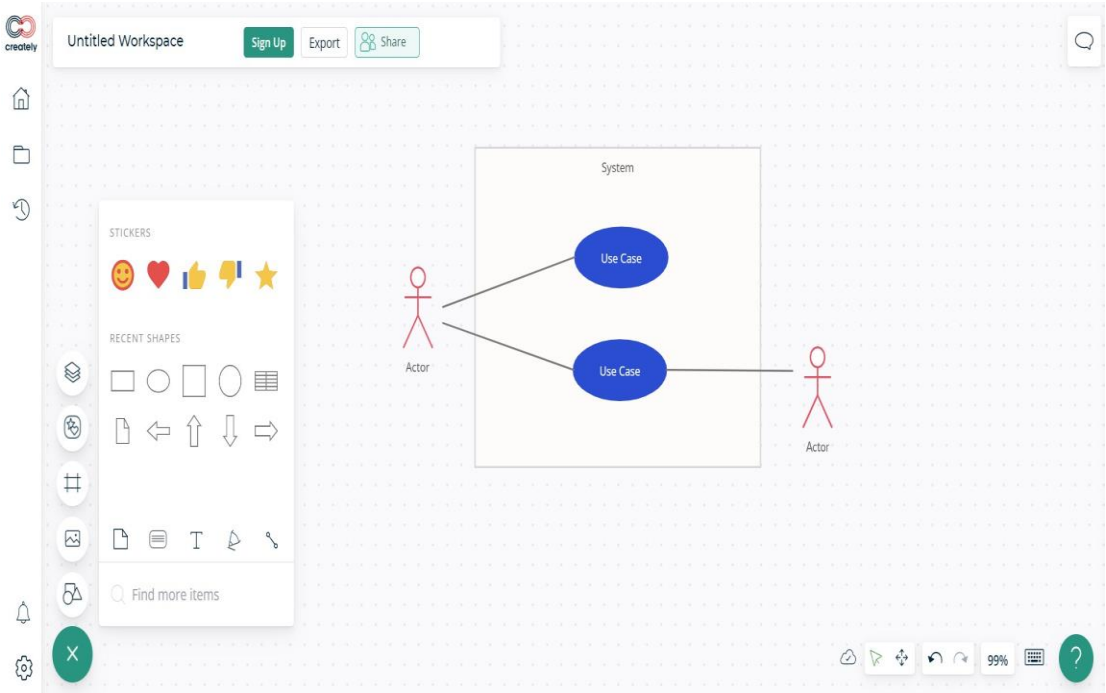

Рис. 2.3. Приклад сервісу з побудови UML діаграм

На рис. 2.3 можна побачити типовий сервіс, який надає можливість користувачам створювати різноманітні діаграми. Саме на зображенні показаний шаблон UML діаграми Use Case із набором графічних елементів.

UML діаграми бувають таких типів:

– діаграма випадків використання – показує користувачів системи та їх взаємодію з нею;

– діаграма класів – показує взаємодію класів;
– діаграма послідовностей – показує об’єкти процесу або системи, їх взаємодію та виклики;
– діаграма стану – показує стан об’єктів, його зміну та події;
– діаграма діяльності – показує дії, що впливають на інші дії, що були у певній частині системи;
– діаграма взаємозв’язку сутностей – показує сутність взаємозв’язків та обмежень між даними;

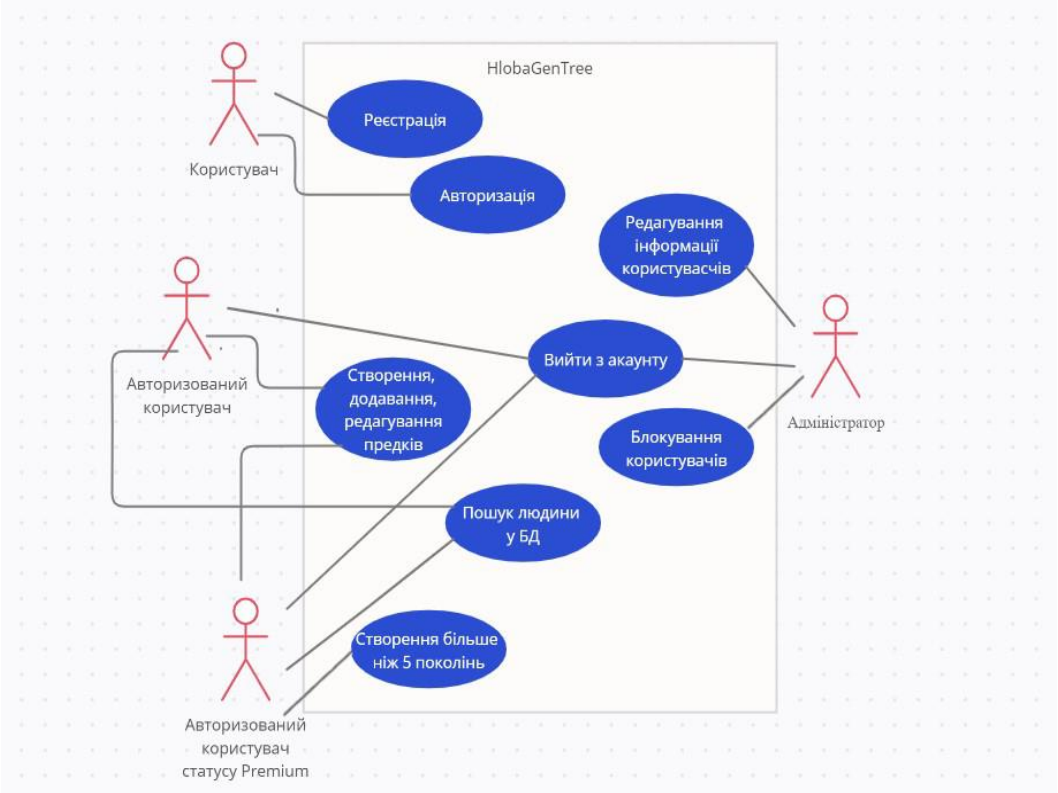

Рис. 2.4. Діаграма випадків використання розробленого додатку

На рис. 2.4 показана діаграма випадків використання (Use Case). Червоними чоловічками в цій діаграмі показуються актори, тобто користувачі програмного додатку. На схемі зображено 4 актори: користувач, авторизованний користувач, користувач Premium, адміністратор. Овалами показуються дії, які актори можуть виконати у додатку.

Актору з назвою користувач доступні тільки дві дії у додатку: реєстрація або ж авторизація. Тобто коли, користувач запускає програму йому буде

запропоновано зареєструватися, якщо у нього ще не має акаунту в у системі, або ж авторизуватися, якщо він уже зараєстрований.

Після авторизації користувач може стати авторизованим користувачем або ж адміністратором, у залежності хто саме авторизується. Авторизований користувач також може бути статусу Premium. У всіх цих акторів свій набір дій, які вони можуть виконувати.

Актор під назвою авторизований користувач може створювати, видаляти нових предків, редагувати інформацію про них, шукати людей у відкритій БД та вийти зі свого акаунту.

Адміністратор може редагувати інформацію про користувачів та їх родину, блокувати користувачів та вийти з акаунту.

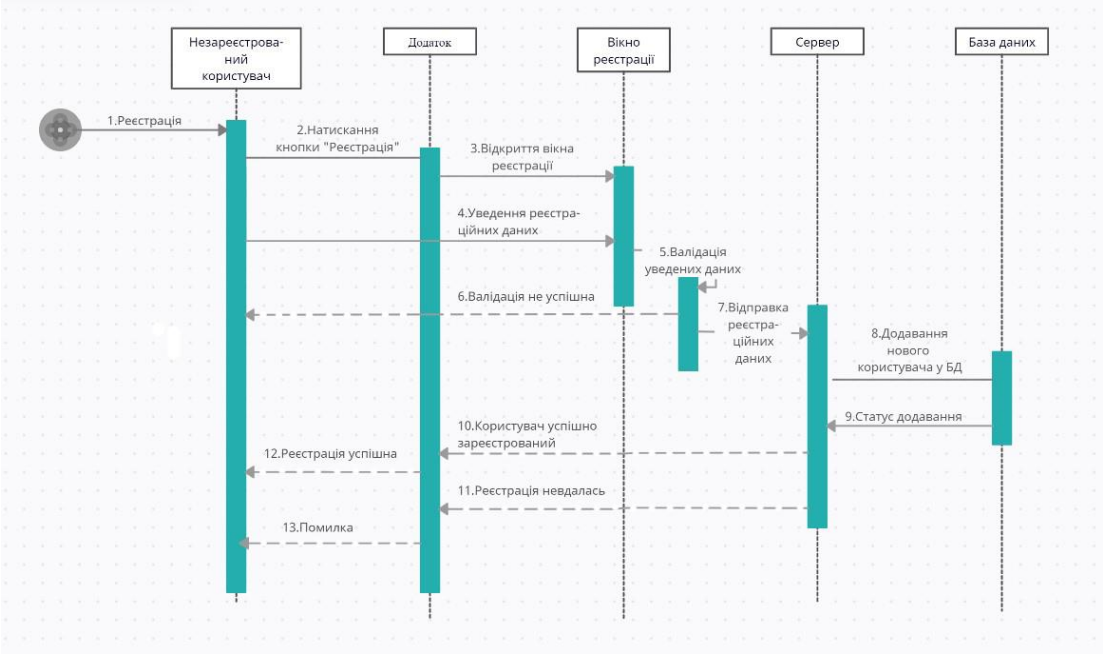

Рис. 2.5. Діаграма послідовностей (процес реєстрації)

На рис. 2.5 можна побачити процес реєстрації за допомогою діаграми послідовностей з нумерацією послідовних дій. Якщо користувач ще не має акаунту в системі для користування нею він повинен зареєструватися (1). Для цього він натискає кнопку “Реєстрація” (2) після чого відкривається вікно із реєстраційною формою (3). Користувач заповнює туди дані (4), якщо дані не

проходять валідацію (5), то користувач отримує повідомлення, що дані введено не коректно (6), якщо ж дані уведено коректно, то вони відправляються на сервер (7), після чого сервер відсилає запит у БД для додавання нового користувача у систему (8). Статус додавання нового користувача у БД надсилається на сервер (9), може бути 2 статуси, або ж користувача було додано (10), або ж щось пішло не так і додати його не вдалося (11). У випадку успішного додавання користувачу висвітиться повідомлення з текстом, що реєстрація пройшла успішно (12), якщо ж навпаки, то напише, що сталася помилка.

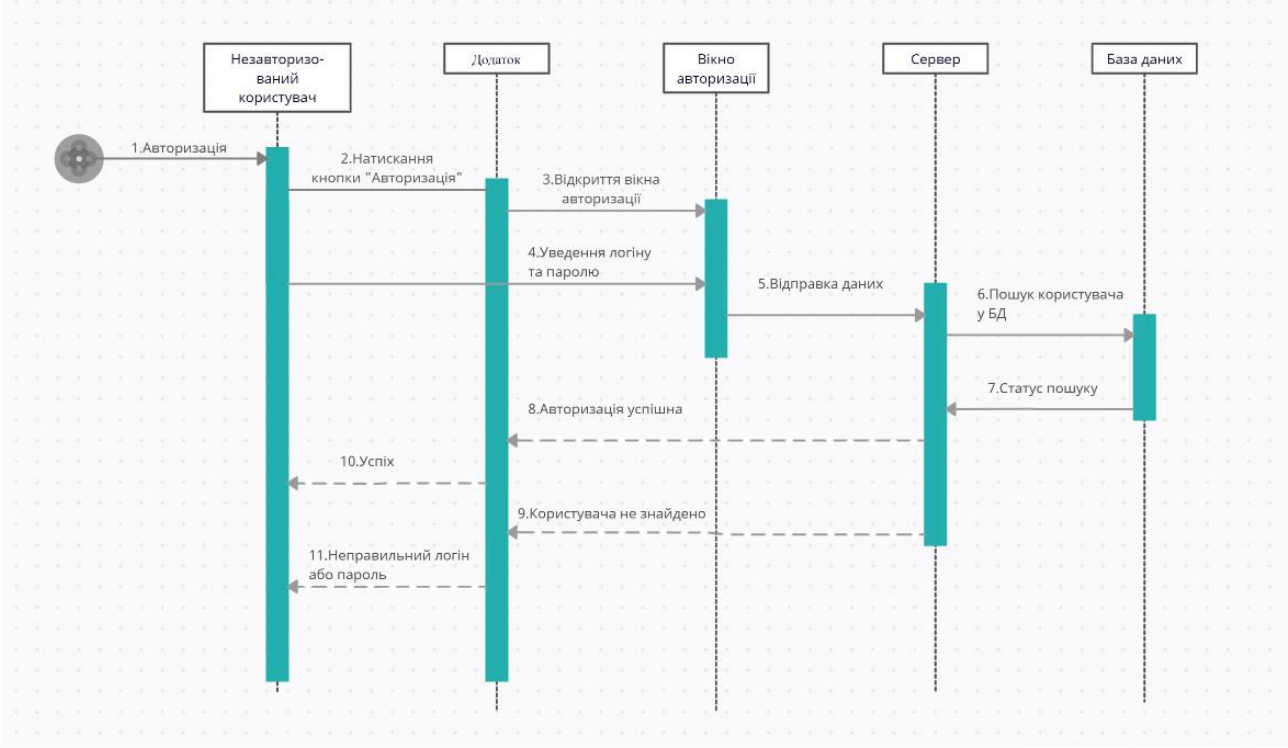

Рис. 2.6. Діаграма послідовностей (процес авторизації)

Процес авторизації користувача можна побачити на рис. 2.6 також за допомогою діаграми послідовностей. Щоб зайти у додаток, уже зареєстрованому користувачеві потрібно пройти етап авторизації (1). Для цього користувач нажимає на кнопку “Авторизуватися” (2), після чого відкриється вікно авторизації (3), в якому користувач уведе дані від свого акаунту (4). Потім ці дані будуть перенаправлені серверу (5), щоб той надіслав запит у БД для перевірки цих даних (6). Статус перевірки даних користувача передається серверу (7) ,

якщо акаунт із такими логіном та паролем було знайдено, то авторизація пройде успішно (8), якщо ж навпаки, то авторизація провалиться (9). У успішному проходженні авторизації користувача пустить у систему, у інакшому випадку висвітиться повідомлення із текстом, що логін або пароль неправильний.

На рис. 2.7, що наведений нижче, представлений процес додавання нового предка у генеалогічне дерева за допомогою діаграми послідовностей.

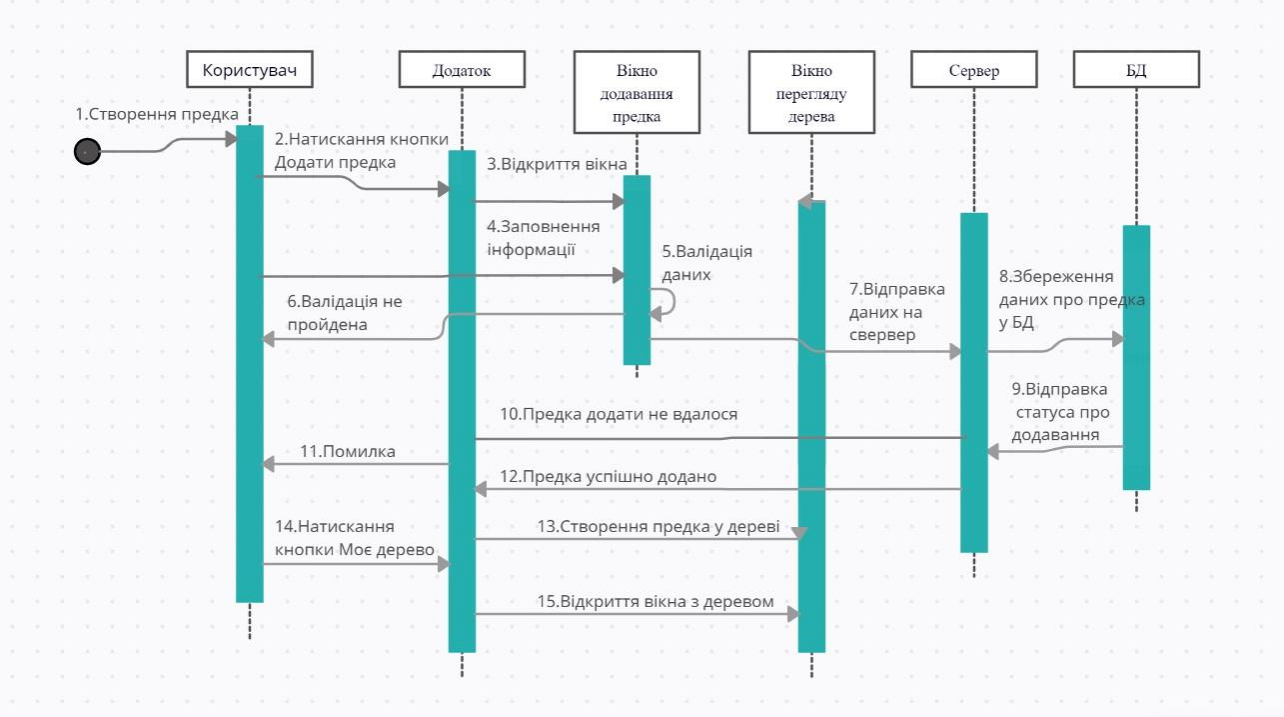

Рис. 2.7. Діаграма послідовностей (додавання предка)

Щоб додати нового предка у своє генеалогічне дерево (1), користувачу потрібно натиснути кнопку “Додати предка” (2) після чого відкриється вікно з формою для заповнення інформації про родича (3). Користувач заповнює ці дані (4) , під час заповнення проводиться валідація введених даних (5), якщо введені дані не пройшли валідацію, то користувачу висвічується про це повідомлення(6), після чого дані про родича відправляються на сервер (7) , сервер робить запрос і додає інформацію у БД (8), а БД відправляє статус додавання назад (9) , якщо предка із якохсь причини не вдалося додати до БД(10), користувач отримає про

це повідомлення (11), якщо ж усе пройшло успішно (12), то предка буде додано до генеалогічного дерева (13). Після всього цього користувачу всього лише потрібно натиснути кнопку “Моє дерево” (14), після чого відкриється вікно з ним (15) і там буде тільки но створений предок.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними розробленого програмного продукту є дані, які охарактеризовують людину, здебільшого вони представляють із себе дані текстового типу, дати та зображення, які заповнюються користувачем на спеціальній для цього формі. Також до вхідних даних можна віднести дані, які користувач уводить при реєстрації у додатку логін, пароль та інформацію про себе. Ще одним типом вхідних даних будуть запроси користувачів на пошук деякої людини у БД.

Вихідні дані представляють собою зручне графічне відображення інформації, яка зберігається у базі дних, здебільшого генеалогічне дерево, форми з відображенням інформації про людину. Також до вихідних даних можна віднести системні повідомлення користувачеві про успішну або провальну операцію, наприклад при реєстрації, авторизації чи додаванні нового предка.

Обмін даними між додатком та базою даних ведеться за допомогою локального серверу MAMP, та MySQL connector, за допомогою якого БД була підключена до додатку і відкрилася можливість обміну даних.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Під час розробки програмного продукту для кваліфікаційної роботи був використаний настільний ПК з наступники технічними характеристиками:

– Процесор: AMD Ryzen 5 3600x 3.8GHz.
– Відеокарта: Geforce GTX 1660 Super 6 GB.
– ОЗУ: 16GB.
– Пам’ять: 1TB.
– Комп’ютерна миш.
– Клавіатура.

2.7.2. Використані програмні засоби

Під час розробки даного програмного продукту для кваліфікаційної роботи були використані наступні середовища розробки:

– Visual Studio Community – безкоштовне інтегроване середовище розробки випущене компанією Microsoft, що містить у собі інструментальний набір для розробки різних програмних продуктів під різні платформи. Написана на мові C++ та C#. Забезпечує програмування на C#, C++,Visual Basic .NET, Visual J Sharp. Містить у собі стандартний редактор, відкладач, компілятор, засоби автозавершення коду, графічні конструктури та багато інших корисних функцій, які облегшують розробнику процес створення програмного продукту.
– Unity – крос-платформенне середовище розробки для створення програм, які зможуть працювати більш ніж на 25 платформах [16]. Основними перевагами даного середовища розробки є наява візуального СР та зручний drug&drop інтерфейс, за допомогою яких розробнику буде дуже зручно взаємодіяти зі своїм проєктом, адже простим перетаскуванням елемента на сцену можна зразу з ним взаємодіяти, задавати йому параметрів та змінювати його, що набагато зменшує час розробки [12]. Також очевидними перевагами є кросплатформеність та модульність системи компонентів.

Для створення локального серверу на власному комп’ютері, аби взаємодіяти з БД було використане безкоштовне програмне забезпечення MAMP, яке містить

у собі вже налаштовані екземпляри роботи із СУБД MySQL. Дане ПЗ створює “міст”, аби розробник міг додавати, видаляти та редагувати інформацію, яка знаходиться у базі даних. Інтерфейс програми можна побачити на рис.2.8.

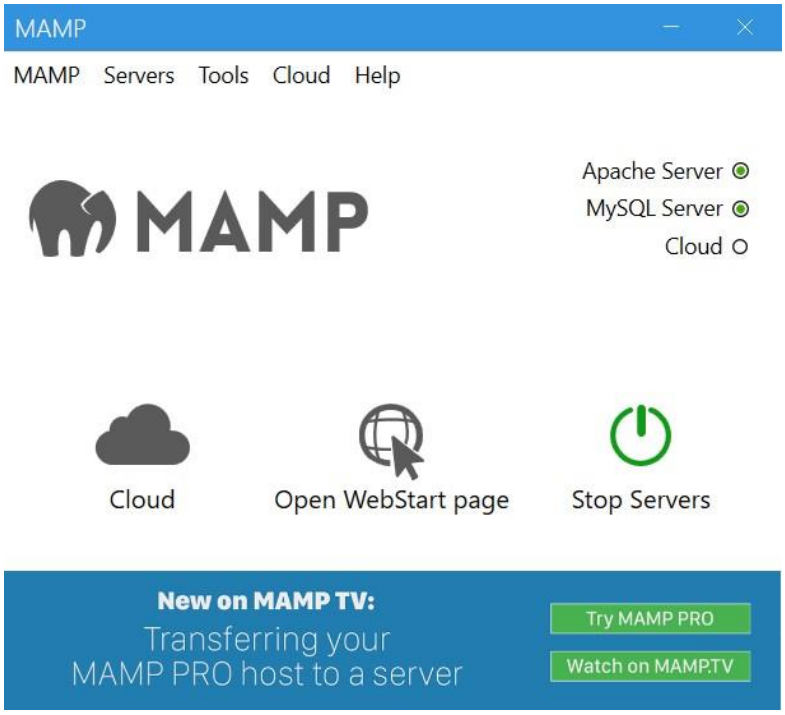

Рис.2.8. Інтерфейс ПЗ MAMP

Для адміністрування СУБД MySQL був використаний безкоштовний веб-додаток phpMyAdmin, за допомогою якого можна без встановлення додаткового програмного забезпечення, а лиш зайшовши на сайт і підключивши свою БД взаємодіяти з нею та продивлятися її вміст.

2.7.3. Виклик та завантаження програми

Для роботи з додатком досить запустити файл HlobaGenTree.exe, після чого програма буде запущена і після вікна завантаження користувач зможе зареєструватися, авторизуватися та після цього успішно користуватися програмним додатком.

2.7.4. Опис інтерфейсу користувача

На рис.2.9 ми можемо побачити завантаження програми. Під час відкриття додатку відбувається вивід на екран процента завантаження програми разом з анімацією проростання дерева. Завантаження відбувається синхронно з завантаженням ресурсів програми.

Рис.2.9. Завантаження додатку

Після завантаження користувач потрапляє на вікно авторизації, де йому потрібно ввести логін та пароль від свого акаунта. Данне вікно авторизації ми можемо побачити на рис.2.10 .

Рис.2.10. Вікно авторизації

Якщо у користувача ще не має зареєстрованого акаунту в додатку, то йому пропонується натиснути на гіперпосилання “Не маєте акаунта? Тоді зареєструйте його!”, після чого відкриється вікно реєстрації.

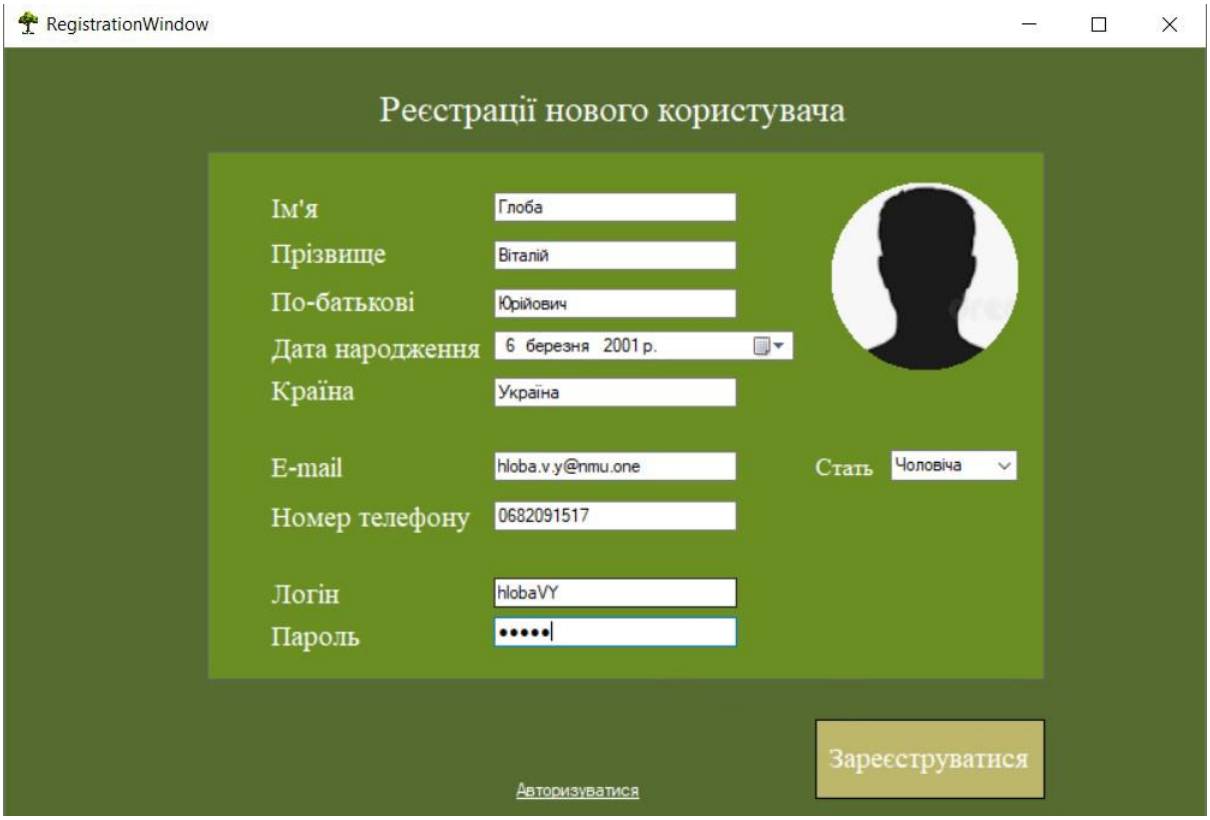

Рис.2.11. Вікно реєстрації

На рис.2.11 зображене вікно реєстрації, увівши всі дані і натиснувши кнопку “Зареєструватися” дані про користувача будуть додані у БД. Якщо ж користувач залишить якесь поле вільним, йому висвітиться текст, що потрібно заповнити всі поля. При зміні статі змінюється фотографія користувача. Потім фото можна буде загрузити своє.

Після успішної реєстрації і авторизації користувач потрапляє на головну сторінку додатка, на якій є логотип, привітальний текст та відео, яке містить інструкцію по використанню додатка. Головну сторінку можна побачити на рис.2.12.

Рис.2.12. Головна сторінка додатку

Щоб додати інформацію про нового родича у базу даних та потім додати його на генеалогічне дерево треба натиснути на кнопку “Додати родича” після чого відкриється вікно, що зображене на рис.2.13.

Рис.2.13. Форма додавання родича

На формі, що зображена на рис.2.13 ми можемо заповнити різноманітну інформацію про свого родича та додати його фото.

Щоб створити генеалогічне дерево треба натиснути на кнопку “Моє дерево” після чого відкриється вікно з інструментарієм для його створення. Нам потрібно натиснути кнопку редагувати, після чого нам стануть активні кнопки “Додати” та “Зв’язки”. Потім нам потрібно вибрати зі списку родича, якого ми попередньо додали у базу даних та натиснути кнопку “Додати” після чого створиться об'єкт із фотографією та ім'я та прізвищем обраного родича. Кожен користувач може переміщувати об'єкти, як йому заманеться, адже чіткої форми дерева немає. Щоб додати зв'язки між родичами потрібно натиснути кнопку “Зв’язки” і після цього ми зможемо намалювати зв’язки.

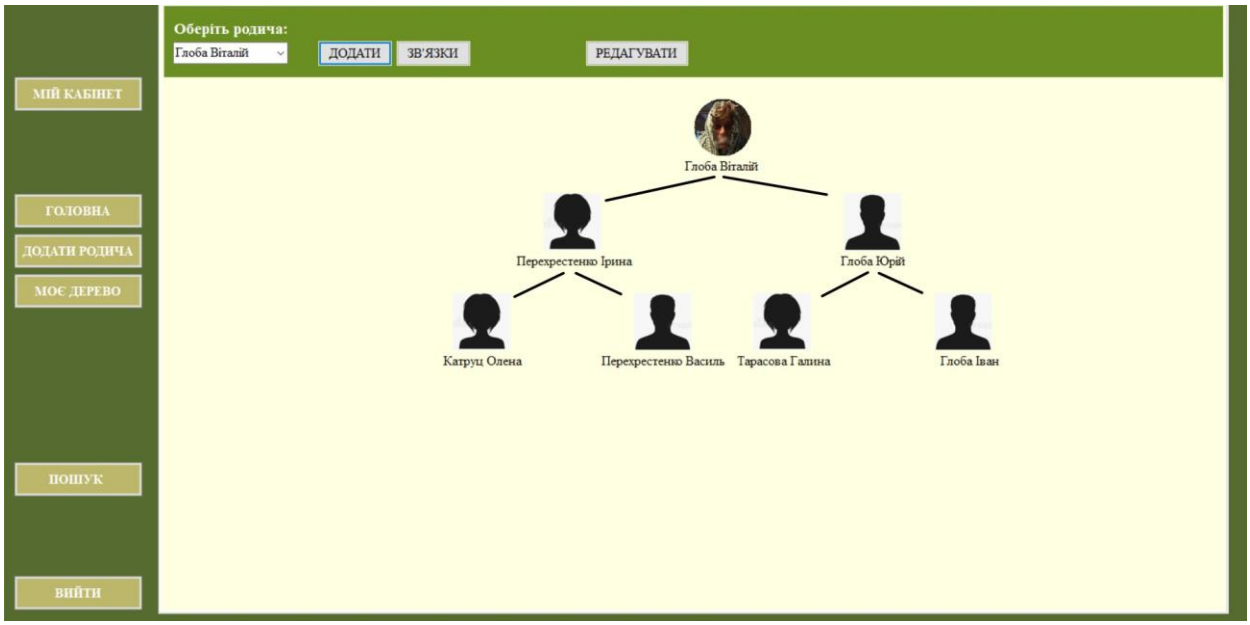

Рис.2.14. Приклад вигляду дерева

РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 1121;
2. Коефіцієнт складності програми – 1,45;
3. Коефіцієнт корекції програми в ході її розробки – 0,08;
4. Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,25;
5. Проаналізувавши інформацію на сайтах з пошуку роботи, таких як Work.ua, Rabota.ua та сайту спільноти програмістів DOU.ua середня заробітна плата програміста на Unity зі знанням БД та стажем роботи у цій сфері до 1-го року становить на місяць від 600$ до 900$. На 12.06.2022 року офіціальний курс валюти НБУ становить 29,2549 грн. Увізьмемо середнє значення з діапазону заробітних плат (600 – 900 американських доларів) та розрахуємо середню місячну заробітну плату у гривнях: ((900+600)/2) * 29,2549 = 21 941,175 грн. Згідно інформації із сайту dtkt.ua, де було прораховано кількість робочих годин на 2022 рік при 40-годинному робочому тижні, урахувавши всі вихідні та святкові дні, місячний фонд робочого часу становить – 166 годин [20]. Знаючи цю інформацію середня годинна заробітна плата програміста на Unity становить – 132 грн;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8;
7. Число виконавців – 1;
8. Вартість машино–години ЕОМ – 14 грн/год;

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей із різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації;

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт корекції програми в ході її розробки.

$$Q = 1121 \cdot 1{,}45 \cdot (1 + 0{,}08) = 1755$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{1755 \cdot 1{,}25}{81 \cdot 0{,}8} = 138 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25) \cdot k}, \text{ людино-годин.} \qquad (3.4)$$

$$t_a = \frac{1755}{25 \cdot 0{,}8} = 87{,}75 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок–схемі:

$$t_n = \frac{Q}{(20..25) \cdot k}, \text{ людино–годин.} \qquad (3.5)$$

$$t_n = \frac{1755}{21 \cdot 0{,}8} = 104{,}5 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \text{ людино-годин.} \qquad (3.6)$$

$$t_{отл} = \frac{1755}{5 \cdot 0{,}8} = 438{,}75 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{дp} + t_{до}\text{ , людино-годин,} \qquad (3.7)$$

де $t_{дp}$ – трудомісткість підготовки матеріалів і рукопису:

$$t_{дp} = \frac{Q}{(15..20)\cdot k}\text{ , людино-годин.} \qquad (3.8)$$

$$t_{дp} = \frac{1755}{(20)\cdot 0{,}8} = 109{,}7 \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{дp}\text{ , людино-годин.} \qquad (3.9)$$

$$t_{до} = 0{,}75 \cdot 109{,}7 = 82{,}3\text{ , людино-годин.}$$

Повернувшись до формули (3.7) розрахуємо витрати праці на підготовку документації:

$$t_д = 109{,}7 + 82{,}3 = 192 \text{ людино-години.}$$

Отримавши всі змінні можемо розрахувати трудомісткість розробки програмного забезпечення за формулою (3.1):

$$t = 50 + 138 + 87{,}75 + 104{,}5 + 438{,}75 + 192 = 1\,011$$ людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}\text{ , грн.} \qquad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}\text{ , грн,} \qquad (3.11)$$

де t – загальна трудомісткість, людино–годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1011 \cdot 132 = 133\,452 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{МЧ}\text{ , грн,} \qquad (3.12)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$С_{МЧ}$ – вартість машино–години ЕОМ, грн/год.

$$З_{МВ} = 438{,}75 \cdot 14 = 6\,142{,}5 \text{ грн.}$$

Повертаємось до формули (3.10) і розраховуємо загальні витрати на створення програмного забезпечення:

$$К_{ПО} = 133\,452 + 6\,142{,}5 = 139\,594{,}5 \text{ грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ , місяців,}$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40-годинному робочому тижні рік F_p = 176 годин).

$$T = \frac{1011}{1 \cdot 176} = 5{,}7 \text{ місяців.}$$

Висновки. На розробку програмного додатку генеалогічного дерева для кваліфікаційної роботи піде 1 011 годин. При стандартному 40-годинному робочому тижні і 176-годинному робочому місяці розробка даного ПЗ займе у програміста 5,7 місяців. Такий час розробки обумовлений кількістю операторів та використаних технологій і постійних корректив у програмі під час розробки. Розрахувавши витрати на заробітну плату виконавцю та вартість машинного часу на розробку програмного забезпечення прийдеться витратити 139 594,5 грн.

ВИСНОВКИ

У даній кваліфікаційній роботі був розроблений програмний додаток з побудови генеалогічного дерева за допомогою середовища розробки Unity.

Розроблений програмний продукт дозволяє звичайному користувачеві зареєструвавши акаунт у системі та авторизувавшись побудувати власне родове дерево, додавши до нього своїх предків або нащадків та заповнивши про них детальну інформацію. Даний додаток має у себе на меті пришвидшити процес створення та урізноманітнити функціонал генеалогічного дерева, у порівнянні зі звичайним генеалогічним деревом у фізичному вигляді.

На практиці розроблений програмний продукт у більшості випадків буде використаний звичайними користувачами у власних цілях, але його можна інтегрувати у навчальні заклади, адже більшість людей створюють свої перші генеалогічні дерева саме у школах, і він буде застосований за призначенням. Також за допомогою родового дерева можна прослідкувати спадкові хвороби, які простежуються у сім'ї та бути пильним.

Актуальність розробленого програмного продукту в кваліфікаційній роботі обґрунтовується відсутністю на українському ринкові програм із подібним простим та зрозумілим функціоналом та безкоштовним доступом до нього.

Під час виконання даної кваліфікаційної роботи були виконані наступні задачі:

- детально проаналізована предметна галузь задачі, що розв'язується;
- проаналізовані інструментальні засоби використаних технологій;
- проаналізовані аналоги програми у мережі;
- з’ясовані вимоги та підстави для розробки даного програмного продукту;
- написаний код програмного продукту;
- обрані раціональна структура та технології при створення додатку.

Програмний продукт розроблено за допомогою середовищ розробки Unity та Visual Studio Community, мови програмування C# та мови запитів SQL. Була створена реляційна база даних за допомогою веб додатку phpMyAdmin, яка була

підключена до середовища розробки за допомогою бібліотеки MySQL Connector/NET. Локальний сервер, який забезпечує взаємодію між розробленим додатком та БД було створено за допомогою програмного забезпечення MAMP.

При написанні кваліфікаційної роботи була прорахована трудомісткість розробленого додатку генеалогічного дерева (1 011 люд-год), проведений підрахунок витрат на розробку програмного додатку (139 594,5 грн.) та разраховано час на його створення (5,7 міс.)

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 8 Oldest Family Trees Ever. Access mode: URL: https://www.oldest.org/culture/family-trees/. date of access: 10.04.2022.;
2. The 7 Oldest Bloodlines in the World Might Surprise You. Access mode: URL: https://genealogyyou.com/oldest-bloodlines-in-the-world/#7_The_Lurie_Family_Bloodline. date of access: 10.04.2022.;
3. Томазов В.В. Родовід. Науково-методичний посібник. – К., 2001. – 52 с.;
4. Томазов В.В. Генеалогія козацько-старшинських родів: історіографія та джерела (друга половина XVII – початок XXI ст.). – К., 2006. – 282 с.;
5. Томазов В.В. Генеалогія історична // Спеціальні історичні дисципліни: довідник: навч. посіб. для студ. вищ. навч. закл. / І.Н. Войцехівська (кер. авт. кол.), В.В. Томазов, М.Ф. Дмитрієнко та ін. – К., 2008. – С. 123–131.;
6. The Tree of Jesse: The “Roots” of Christian Faith. Access mode: URL: https://stm.yale.edu/the-tree-of-jesse. date of access: 11.04.2022.;
7. MVC Framework Tutorial for Beginners: What is, Architecture & Example. Access mode: URL: https://www.guru99.com/mvc-tutorial.html. date of access: 15.05.2022.;
8. MVC Framework - Introduction. Access mode: URL: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm. date of access: 15.05.2022.;
9. Beaulieu A. Learning SQL. 2nd ed. Beijing : O'Reilly, 2009. - 98 p.;
10. Petzold C. Programming Microsoft Windows with C♯. Redmond, Wash : Microsoft Press, 2002. - 356 p.;
11. Okita A. Learning C# Programming with Unity 3D. Routledge, 2014. - 124 p.;
12. Blackman S. Unity for Absolute Beginners. Berkeley, CA : Apress, 2014. Access mode: URL: https://doi.org/10.1007/978-1-4302-6778-2. date of access: 05.05.2022.;

13. Фрімен, Адам ASP.NET MVC 3 Framework з прикладами на C# для професіоналів / Адам Фрімен , Стівен Сандерсон. - М.: Вільямс, 2011. - 672 с;

14. Software W. Learning Sql/Book and Disk. Prentice Hall, 1991. - 135 p.;

15. McGrath M. C# Programming in Easy Steps 2nd Edition. Taylor & Francis Group, 2020. - 87 p.;

16. Hocking J. Unity in Action: Multiplatform game development in C#. Manning Publications, 2018. - 263 p.;

17. Silberschatz A. Database system concepts. 6th ed. Dubuque, IA : McGraw-Hill Companies, 2011. - 112 p.;

18. Eckstein J., Schultz B. R. Introductory Relational Database Design for Business, with Microsoft Access. Wiley, 2018. - 157 p.;

19. Garcia-Molina H., Widom J. D., Ullman J. D. Database system implementation. Prentice Hall, 1999. 456 p.;

20. Норми тривалості робочого часу на 2022 рік. Режим доступу: URL: https://services.dtkt.ua/catalogues/worktime/124. дата звернення: 08.06.2022.

ДОДАТОК А

КОД ПРОГРАМИ

LoadingScene.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class LoadingScene : MonoBehaviour
{
    public int idScene; // айді завантажуємої сцени

    //Завантажувані елементи на сцені:
    public Image imageLoading;
    public Text textLoading;
    public Text pressAnyKey;

    // Start is called before the first frame update
    void Start()
    {
        StartCoroutine(Load());
    }

    // Update is called once per frame
    void Update()
    {

    }

    IEnumerator Load()
    {
        AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(idScene);

        asyncLoad.allowSceneActivation = false;

        pressAnyKey.enabled = false;

        while (!asyncLoad.isDone)
        {
```

```
            float progress = asyncLoad.progress / 0.9f;

            imageLoading.fillAmount = asyncLoad.progress;

            textLoading.text = string.Format("{0:0}%", progress * 100);

            if (asyncLoad.progress >= .9f && !asyncLoad.allowSceneActivation)
            {
               pressAnyKey.enabled = true;
               if (Input.anyKeyDown)
               {
                  asyncLoad.allowSceneActivation = true;
               }
            }
            yield return null;
         }
      }
   }
```

DataBase.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HlobaTreeGen
{
   class DataBase
   {
      MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=root;database=hlobatreegendb");
      public void openConnection()
      {
         if (connection.State == System.Data.ConnectionState.Closed)
            connection.Open();
      }

      public void closeConnection()
      {
         if (connection.State == System.Data.ConnectionState.Open)
```

```
            connection.Close();
        }

        public MySqlConnection getConnection()
        {
            return connection;
        }
    }
}
```

Authorization.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace HlobaTreeGen
{
    public partial class HlobaTreeGen : Form
    {
        public HlobaTreeGen()
        {
            InitializeComponent();

            validLoginText.Visible = false;
            validPasswordText.Visible = false;
            authorizationError.Visible = false;
        }

        private void loginBox_Click(object sender, EventArgs e)
        {
            if (loginField.Text == "Уведіть логін")
            {
                loginField.Text = "";
                loginField.ForeColor = Color.Black;
            }
```

```
    if (passwordField.Text == "")
    {
        passwordField.Text = "Уведіть пароль";
        passwordField.ForeColor = Color.Gray;
    }

}

private void passwordBox_Click(object sender, EventArgs e)
{
    if (passwordField.Text == "Уведіть пароль")
    {
        passwordField.UseSystemPasswordChar = true;
        passwordField.Text = "";
        passwordField.ForeColor = Color.Black;
    }
    if (loginField.Text == "")
    {
        loginField.Text = "Уведіть логін";
        loginField.ForeColor = Color.Gray;
    }
}

private void authorizationButton_Click(object sender, EventArgs e)
{
    if (loginField.Text == "")
    {
        loginField.Text = "Уведіть логін";
        loginField.ForeColor = Color.Gray;

        return;
    }

    if (passwordField.Text == "")
    {
        passwordField.Text = "Уведіть пароль";
        passwordField.ForeColor = Color.Gray;

        return;
    }
```

```
            if (loginField.Text == "Уведіть логін")
            {
               validLoginText.Visible = true;

               return;
            }

            else validLoginText.Visible = false;

            if (passwordField.Text == "Уведіть пароль")
            {
               validPasswordText.Visible = true;

               return;
            }

            else validPasswordText.Visible = false;
            DataBase dataBase = new DataBase();
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            MainWindow main = new MainWindow();

            String userLogin = loginField.Text;
            String userPassword = passwordField.Text;
            string id;
            MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE `login` =
@userLogin and `password` = @userPassword", dataBase.getConnection());
            MySqlCommand commandID = new MySqlCommand("SELECT `id_user` FROM `users` WHERE
`login` = @userLogin and `password` = @userPassword", dataBase.getConnection());
            command.Parameters.Add("@userLogin", MySqlDbType.VarChar).Value = userLogin;
            command.Parameters.Add("@userPassword", MySqlDbType.VarChar).Value = userPassword;
            commandID.Parameters.Add("@userLogin", MySqlDbType.VarChar).Value = userLogin;
            commandID.Parameters.Add("@userPassword", MySqlDbType.VarChar).Value = userPassword;

            dataBase.openConnection();
            id = commandID.ExecuteScalar().ToString();
            adapter.SelectCommand = command;
            adapter.Fill(table);

            if (table.Rows.Count > 0)
            {
               main.Show();
```

```
            this.Hide();
        }
        else
            MessageBox.Show("Логін або пароль не правильний!");
    }

    private void registrationLink_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        RegistrationWindows registration = new RegistrationWindows();
         registration.Show();
    }

    private void passwordBox_TextChanged(object sender, EventArgs e)
    {

    }
  }
}
```

Registration.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace HlobaTreeGen
{
  public partial class RegistrationWindows : Form
  {
    string selectedSex;
    public RegistrationWindows()
    {
      InitializeComponent();

      sexBox.SelectedIndexChanged += sexBox_SelectedIndexChanged;
      passwordReg.UseSystemPasswordChar = true;
    }

    private void sexBox_SelectedIndexChanged(object sender, EventArgs e)
    {
      selectedSex = sexBox.SelectedItem.ToString();
```

```
    if (selectedSex == "Чоловіча")
    {
        sexPicture.Image = Properties.Resources.icoMan;
    }
    else sexPicture.Image = Properties.Resources.icoWomen;

}

private void RegistrationWindows_Load(object sender, EventArgs e)
{
    System.Drawing.Drawing2D.GraphicsPath path = new System.Drawing.Drawing2D.GraphicsPath();
    path.AddEllipse(10, 0, 125, 125);
    Region rgn = new Region(path);
    sexPicture.Region = rgn;
    sexPicture.BackColor = System.Drawing.SystemColors.ActiveCaption;
}

private void regButton_Click(object sender, EventArgs e)
{
    if (nameReg.Text == "")
    {
        errorText.Visible = true;
        return;
    }
    if (surnameReg.Text == "")
    {
        errorText.Visible = true;
        return;
    }
    if (patronymicReg.Text == "")
    {
        errorText.Visible = true;
        return;
    }
    if (countryReg.Text == "")
    {
        errorText.Visible = true;
        return;
    }
    if (emailReg.Text == "")
    {
        errorText.Visible = true;
```

```
            return;
        }
        if (phoneReg.Text == "")
        {
            errorText.Visible = true;
            return;
        }
        if (loginReg.Text == "")
        {
            errorText.Visible = true;
            return;
        }
        if (passwordReg.Text == "")
        {
            errorText.Visible = true;
            return;
        }
        if (checkUser())
            return;

        DataBase dataBase = new DataBase();
        MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`name`,`surname`,`patronymic`,`sex`,`birthday`,`country`,`e-mail`,`phone`,`login`,`password`) VALUES (@name, @surname, @patronymic, @sex, @birthday, @country, @email, @phone, @login, @password)", dataBase.getConnection());

        command.Parameters.Add("@name", MySqlDbType.VarChar).Value = nameReg.Text;
        command.Parameters.Add("@surname", MySqlDbType.VarChar).Value = surnameReg.Text;
        command.Parameters.Add("@patronymic", MySqlDbType.VarChar).Value = patronymicReg.Text;
        command.Parameters.Add("@sex", MySqlDbType.VarChar).Value = sexBox.Text;
        command.Parameters.Add("@birthday", MySqlDbType.Date).Value = birthdayReg.Value;
        command.Parameters.Add("@country", MySqlDbType.VarChar).Value = countryReg.Text;
        command.Parameters.Add("@email", MySqlDbType.VarChar).Value = emailReg.Text;
        command.Parameters.Add("@phone", MySqlDbType.Int32).Value = phoneReg.Text;
        command.Parameters.Add("@login", MySqlDbType.VarChar).Value = loginReg.Text;
        command.Parameters.Add("@password", MySqlDbType.VarChar).Value = passwordReg.Text;

        dataBase.openConnection();

        if (command.ExecuteNonQuery() == 1)
            MessageBox.Show("Користувач успішно створений");
        else
```

```
                MessageBox.Show("Виникла помилка");

            dataBase.closeConnection();
        }
        public Boolean checkUser()
        {
            DataBase dataBase = new DataBase();
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();

            MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE `login` =
@userLogin", dataBase.getConnection());
            command.Parameters.Add("@userLogin", MySqlDbType.VarChar).Value = loginReg.Text;

            adapter.SelectCommand = command;
            adapter.Fill(table);

            if (table.Rows.Count > 0)
            {
                MessageBox.Show("Користувач із таким логіном уже зареєстрований!");
                return true;
            }
            else
                return false;
        }
    }
}
```

Main.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HlobaTreeGen
{
   public partial class MainWindow : Form
   {
      string userid;
      string relative;
      private PictureBox picture = new PictureBox();
      private Label label = new Label();
      public bool status = false;
      public int i = 0;

      public MainWindow()
      {
         InitializeComponent();
      }

      private void mainButton_Click(object sender, EventArgs e)
      {
         tabControl1.SelectTab(mainPage);
      }

      private void addButton_Click(object sender, EventArgs e)
      {
         tabControl1.SelectTab(addPage);
      }

      private void treeButton_Click(object sender, EventArgs e)
      {
         tabControl1.SelectTab(treePage);
      }

      private void findButton_Click(object sender, EventArgs e)
      {
         tabControl1.SelectTab(findPage);
      }

      private void exitButton_Click(object sender, EventArgs e)
      {
         Application.Exit();
```

```
}

private void button1_Click(object sender, EventArgs e)
{

}

private void MainWindow_Load(object sender, EventArgs e)
{
   System.Drawing.Drawing2D.GraphicsPath path = new System.Drawing.Drawing2D.GraphicsPath();

   path.AddEllipse(0, 0, 70, 70);
   Region rgn = new Region(path);
   mePic.Region = rgn;
   mePic.BackColor = System.Drawing.SystemColors.ActiveCaption;
}

private void addRelativeButton_Click(object sender, EventArgs e)
{
   if (nameAdd.Text == "")
   {
      errorAdd.Visible = true;

      return;
   }
   if (surnameAdd.Text == "")
   {
      errorAdd.Visible = true;

      return;
   }
   if (patronymicAdd.Text == "")
   {
      errorAdd.Visible = true;

      return;
   }
   if (placeAdd.Text == "")
   {
      errorAdd.Visible = true;

      return;
```

```
        }
        if (professionAdd.Text == "")
        {
            errorAdd.Visible = true;

            return;
        }
        DataBase dataBase = new DataBase();

        MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`name`,`surname`,`patronymic`,`sex`,`birthday`,`place`,`education`,`blood_type`,`profession`) VALUES (@name, @surname, @patronymic, @sex, @birthday, @place, @education, @blood, @profession)", dataBase.getConnection());

        command.Parameters.Add("@name", MySqlDbType.VarChar).Value = nameAdd.Text;
        command.Parameters.Add("@surname", MySqlDbType.VarChar).Value = surnameAdd.Text;
        command.Parameters.Add("@patronymic", MySqlDbType.VarChar).Value = patronymicAdd.Text;
        command.Parameters.Add("@sex", MySqlDbType.VarChar).Value = sexAdd.Text;
        command.Parameters.Add("@birthday", MySqlDbType.Date).Value = dateAdd.Value;
        command.Parameters.Add("@place", MySqlDbType.VarChar).Value = placeAdd.Text;
        command.Parameters.Add("@education", MySqlDbType.VarChar).Value = educationAdd.Text;
        command.Parameters.Add("@blood", MySqlDbType.VarChar).Value = bloodAdd.Text;
        command.Parameters.Add("@profession", MySqlDbType.VarChar).Value = professionAdd.Text;

        dataBase.openConnection();

        if (command.ExecuteNonQuery() == 1)
            MessageBox.Show("Родич успішно доданий");

        else
            MessageBox.Show("Виникла помилка");

        dataBase.closeConnection();
    }

    private void addPhotoButton_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();

        ofd.Filter = "Image Files(*.BMP; *.JPG; *.GIF; *.PNG)|*.BMP; *.JPG; *.GIF; *.PNG|All files (*.*)|*.*";

        if(ofd.ShowDialog() == DialogResult.OK)
        {
```

```
                try
                {
                    relativePicture.Image = new Bitmap(ofd.FileName);
                }

                catch
                {
                    MessageBox.Show("Не вдається відкрити обраний файл", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }
        }

        private void addRelButton_Click(object sender, EventArgs e)
        {
            relative = chooseRelative.Text;

            picture = new PictureBox();
            label = new Label();

            this.SuspendLayout();

            picture.Location = new Point(160 * i, 130);
            picture.Size = new Size(100, 100);
            picture.BackColor = new Color();
            picture.Image = Properties.Resources._1;

            label.Location = new Point(190 * i, 200);
            label.Size = new Size(150, 20);
            label.ForeColor = Color.Black;
            label.Text = relative;

            picture.MouseMove += Picture_MouseMove;
            picture.MouseUp += Picture_MouseUp;
            picture.MouseDown += Picture_MouseDown;

            label.MouseMove += Label_MouseMove;
            label.MouseUp += Label_MouseUp;
            label.MouseDown += Label_MouseDown;


            this.Controls.Add(picture);
```

```
            this.ResumeLayout();

            i++;
        }

        private void Label_MouseDown(object sender, MouseEventArgs e)
        {
            status = true;
        }

        private void Label_MouseUp(object sender, MouseEventArgs e)
        {
            status = false;
        }

        private void Label_MouseMove(object sender, MouseEventArgs e)
        {
            label.Location = new Point((Cursor.Position.X - this.Location.X), (Cursor.Position.Y - this.Location.Y));
        }

        private void Picture_MouseDown(object sender, MouseEventArgs e)
        {
            status = false;
        }

        private void Picture_MouseUp(object sender, MouseEventArgs e)
        {
            status = true;
        }

        private void Picture_MouseMove(object sender, MouseEventArgs e)
        {
            picture.Location = new Point((Cursor.Position.X - this.Location.X), (Cursor.Position.Y - this.Location.Y));
        }
    }
}
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему:

"Розробка програмного додатку з побудови генеалогічного дерева за допомогою середовища розробки Unity"

студента групи 121-18-2 Глоби Віталія Юрійовича

Керівник економічного розділу

доцент каф. ПЕП та ПУ, к.е.н **Л.В.Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Глоба.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Глоба.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Глоба.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Глоба.ppt	Презентація кваліфікаційної роботи