**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО**

Факультет обчислювальної техніки, інтелектуальних та управляючих систем

Кафедра інформаційних технологій

Напрям підготовки 6.050101 – Комп’ютерні науки

До захисту допускаю

Завідувач кафедри

Авраменко В.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, дата)

**ДИПЛОМНА РОБОТА**

освітнього ступеня БАКАЛАВР

Тема роботи

**ІНФОРМАЦІЙНА СИСТЕМА «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ»**

Виконавець Савченко В.М.

Керівник к.ф.т.н, доцент

Косенюк Г. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Черкаси – 2017**

**АНОТАЦІЯ**

У даній дипломній роботі описано етапи розробки інформаційної системи «Генеалогічне дерево родини», котра реалізована на мові програмування C#. Розглянуто аналоги, поставлено технічне завдання для розроблюваної системи, визначено основні технології для розробки. Детально описано головні класи програмного продукту та представлено основні діаграми.

Розроблений додаток дозволяє формувати генеалогічне дерево родини, та визначати ймовірність захворювання нащадків на генетичні хвороби, що сформує уявлення про можливі захворювання у майбутнього покоління.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, ГЕНЕАЛОГІЯ, ПРОГРАМА, ПОБУДОВА ГЕНЕАЛОГІЧНОГО ДЕРЕВА, РОДОВІД, ГЕНЕТИЧНІ ЗАХВОРЮВАННЯ, АНАЛІЗ РОДОВОДУ, НАЩАДКИ, ВІДНОСИНИ.

**ANNOTATION**

This research paper describes the stages of the information system "Family Tree", which is implemented in the programming language C #. Considered analogs delivered technical specification for the system being developed, the basic technology for development. Described in detail the major classes of product and presents basic diagram.

The developed application allows you to create a family tree of the family, and to determine the likelihood of disease in offspring genetic disease that generate ideas about possible disease in the next generation.

Keywords: Information system, database, genealogy, program, construction of a family tree, genealogy, genetic disorder ANALYSIS ancestry, descendants, relationships

ЗМІСТ

[ВСТУП 8](#_Toc485679482)

[1 ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ» 11](#_Toc485679483)

[1.1 Призначення та область застосування інформаційної системи «Генеалогічне дерево родини» 11](#_Toc485679484)

[1.2 Огляд способів і засобів вирішення поставлених задач 13](#_Toc485679485)

[1.2.1 Особливості виконання генеалогічного дослідження 13](#_Toc485679486)

[1.2.2 Закономірності успадкування генетичних захворювань за законами Менделя 14](#_Toc485679487)

[1.3 Аналіз переваг та недоліків існуючих аналогів інформаційної системи «Генеалогічне дерево родини» 17](#_Toc485679488)

[1.3.1 Програмний продукт «Family Tree Builder» 17](#_Toc485679489)

[1.3.2 Програмний продукт «SimTree» 19](#_Toc485679490)

[1.3.3 Програмний продукт «Genopro» 20](#_Toc485679491)

[1.3.4 Програмний продукт «RootsMagic Essentials» 22](#_Toc485679492)

[1.4 Постановка задачі на розробку 23](#_Toc485679493)

[Висновки до першого розділу 24](#_Toc485679494)

[2 АНАЛІЗ І ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ» 26](#_Toc485679495)

[2.1 Моделювання предметної області системи 26](#_Toc485679496)

[2.2 Формулювання та аналіз вимог до інформаційної системи 28](#_Toc485679497)

[2.2.1 Формування вимог за допомогою діаграми прецедентів 30](#_Toc485679498)

[2.2.2 Формування вимог за допомогою діаграми діяльності 32](#_Toc485679499)

[2.2.3 Аналіз вимог за допомогою діаграми послідовностей 34](#_Toc485679500)

[2.2.4 Функціонування системи за допомогою діаграми комунікації 36](#_Toc485679501)

[2.4. Об’єктно-орієнтоване проектування інформаційної системи 37](#_Toc485679502)

[2.4.1 Архітектурне проектування 38](#_Toc485679503)

[2.4.2 Детальне проектування 41](#_Toc485679504)

[2.4.3 Розгортання програмної системи на апаратних засобах 43](#_Toc485679505)

[2.5 Модель поведінки інформаційної системи «Генеалогічне дерево родини» 44](#_Toc485679506)

[Висновки до другого розділу 46](#_Toc485679507)

[3 РОЗРОБКА І ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ» 47](#_Toc485679508)

[3.1 Огляд засобів реалізації розробки інформаційної системи «Генеалогічне дерево родини» 47](#_Toc485679509)

[3.1.1 Архітектурний шаблон MVC 47](#_Toc485679510)

[3.1.2 Мова програмування C# 49](#_Toc485679511)

[3.1.3 Microsoft Visual Studio 2017 50](#_Toc485679512)

[3.1.4 Фреймворк ASP.NET 51](#_Toc485679513)

[3.1.5 Використання JSON для передачі даних 52](#_Toc485679514)

[3.2 Функціональний опис інформаційної системи «Генеалогічне дерево родини» 52](#_Toc485679515)

[3.3 Розробка бази даних інформаційної системи «Генеалогічне дерево родини» 54](#_Toc485679516)

[3.3.1 Концептуальна модель даних 54](#_Toc485679517)

[3.3.2 Логічна модель даних 56](#_Toc485679518)

[3.3.3 Фізична модель даних 57](#_Toc485679519)

[3.4 Розробка інтерфейсу користувача 62](#_Toc485679520)

[3.5 Опис розробки програмних компонентів 65](#_Toc485679521)

[3.6 Тестування інформаційної системи «Генеалогічне дерево родини» 66](#_Toc485679522)

[Висновки до третього розділу 69](#_Toc485679523)

[ВИСНОВКИ 71](#_Toc485679524)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 74](#_Toc485679525)

[ДОДАТОК А 75](#_Toc485679526)

# ВСТУП

Майже у кожної особи нашого суспільства виникало прагнення до сімейної історії. Із приходом інформаційних технологій це прагнення стає значно доступнішим, адже з’являються можливості легкого збору, зберігання та аналізування історичних даних за допомогою існуючих програмних продуктів.

Питання використання інформаційних систем для створення генеалогічного дерева родини є цілком актуальним. Проведення всіх етапів збору та аналізу інформації без використання сучасної обчислювальної техніки вимагає кропіткої праці, а також великої кількості часу у відповідного спеціаліста. На практиці часто виникають помилки у неправильному знаходженні та зберіганні інформації про родовід. Тому на сьогоднішній день програмні продукти, що пов’язані з побудовою генеалогічних дерев, є досить важливими в практичному застосуванні, але вони через велику вартість є не зовсім доступними.

Для того щоб інформаційна система мала широке використання, в основу програмного продукту закладають велику базу даних та базу знань. За допомогою даної технології можна отримати змогу збирати та впорядковувати величезну кількість даних, а також проводити різноманітні операції з ними.

**Актуальність теми** полягає у тому, що дана інформаційна система дозволить покращити зберігання історичної інформації про сім’ю. Є можливість визначення ймовірності захворювання генетичними хворобами від пращурів до нащадків, побудова графіків за масивом інформаційних даних від усіх родичів. Також актуальним є реалізація онлайн мережі у якій є можливість додавання користувачів до друзів з функцією перегляду, та копіювання інформації про рідню. Створення бази знань генетичних захворювань, що можуть проявлятися у людей. Все це надасть можливість зберігати більш різноманітнішу інформацію про свій родовід.

**Мета даної роботи** полягає у створенні інформаційної системи «Генеалогічне дерево родини», що полегшує формування загальної картини роду. Дана система розробляється для роботи з великим потоком даних, а також з метою скоротити витрати часових ресурсів користувачів у цілому. Для досягнення цієї мети потрібно виконати ряд **завдань**:

1. Дослідити поняття генеалогічного дерева родини та його основних методів побудови.
2. Дослідити структуру існуючих інформаційних систем побудови генеалогічного дерева родини, а також визначити їх головні переваги та недоліки.
3. Розробити теоретичну частину інформаційної системи «Генеалогічне дерево родини».
4. Розробити практичну частину інформаційної системи «Генеалогічне дерево родини».
5. Розробити простий інтерфейс, який буде доступний для розуміння та використання не досвідченим користувачам.
6. Розробити програмний продукт, що відповідатиме вимогам поставленої задачі на розробку та дотримуватиметься ідеології актуальності дипломної роботи.

**Об’єкт дослідження** – методи та засоби створення інформаційної системи генеалогічного дерева родини.

**Предмет дослідження** – це програмні засоби для автоматизації зберігання та аналізу інформації про родовід, а також можливість знаходження ймовірності передачі генетичних захворювань від пращурів до нащадків .

**Область застосування –** інформаційна система є корисною для користувачів, які поставили собі за мету сформувати своє генеалогічне дерево, що міститиме всю відому інформацію про предків та дає можливість передбачити генетичні захворювання у нащадків.

**Практичне значення** одержаних результатів полягає в формуванні генеалогічного дерева родини, визначення ймовірності прояву генетичних захворювань у нащадків. Автоматизації розрахункових задач даної предметної області.

Дипломна робота складається із вступу,трьохрозділів, висновків, списку використаної літератури (дев’ятнадцять найменувань). Робота містить десять таблиць, двадцять вісім рисунків. Роботу викладено на семидесяти п’яти сторінках друкованого тексту.

# 1 ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ»

## 1.1 Призначення та область застосування інформаційної системи «Генеалогічне дерево родини»

Генеалогічне (родовідне) дерево – схематичне представлення родинних зв'язків у вигляді умовно-символічного дерева, в коренях якого знаходиться родозасновник, а на кроні та вітках – представники різних ліній родоводу[1]. Таке схематичне зображення надає користувачеві більш зручний вигляд представлення родоводу.

Користувач має можливість пошуку та зберігання різноманітної інформації про своїх предків, але робота із великою кількістю даних одночасно, не є досить зручною та займає велику кількість часу. Тому доцільно використовувати інформаційну систему генеалогічного дерева родини.

Інформаційна система – це сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів[2].

Інформаційна система генеалогічного дерева має можливість визначення ймовірності прояву генетичних захворювань, що передались від предків до нащадків, це стає можливим завдяки генетичним алгоритмам, зокрема за допомогою законів Менделя, що становлять основу класичної генетики спадковості. Генетичні захворювання це захворювання, обумовлені порушеннями в процесах збереження, передачі та реалізації генетичної інформації. В основі спадкових захворювань лежать мутації: генні, хромосомні та геномні. Відповідно до цього всі спадкові хвороби людини можна об’єднати в 4 великі групи:

* генні (зміни на рівні окремих нуклеотидів);
* геномні (зміни кількості цілих хромосом);
* хромосомні (внутрішньо - і міжхромосомні перебудови);
* мультифакторіальні (на розвиток хвороби впливають і гени, і фактори навколишнього середовища).

Спадкові захворювання і захворювання із спадковою схильністю є основною причиною смертності, особливо в економічно розвинутих країнах. На підставі великого статистичного матеріалу встановлено, що через генетичні порушення 1 із 130 зародків гине вже в перші дні, 25 % припиняють своє існування на пізніших строках вагітності. 40 % дитячої смертності зумовлено спадковими дефектами, 5-12 % новонароджених мають природжені генетичні дефекти, які зумовлені дефектом хромосом або мутаціями окремих генів. Кожна людина має 5-10 потенціально шкідливих генів, які передаються нащадкам разом з нормальними генами.[3].

Основним призначенням продукту проектування є побудова генеалогічного дерева родини, яке представлене у класичній схемі, формування графіків на основі відомих даних та знаходження ймовірності прояву спадкових генетичних захворювань у майбутніх нащадків.

Основною областю застосування даної інформаційної системи «генеалогічне дерево родини» є діяльність користувачів, які прагнуть дізнатися більше про власний родовід та про його подальший розвиток.

## 1.2 Огляд способів і засобів вирішення поставлених задач

### 1.2.1 Особливості виконання генеалогічного дослідження

Генеалогічне дослідження передбачає пошук інформації у архівах щодо окремої родини на протязі кількох поколінь. Для виконання дослідження необхідна точна біографічна інформація: імена, прізвища та по батькові, місце та роки народження, місце проживання, здобутки відомих членів родини.

Повнота і чіткість відомостей є гарантією правильного пошуку необхідних даних.

Основними джерелами генеалогічних досліджень та історії родини є метричні книги, реєстрації актів громадянського стану. Іншими, також важливими матеріалами є документи посімейно-майнового обліку (для України це наприклад подвірні, посімейні (погосподарчі) списки, ревізькі «сказки», списки платників податку на прибуток, платників обов'язкового страхування, виборців та громадян, що були позбавлені виборчого права; справи про опікунство, межові книги, документація переписів 1917, 1921, 1926, 1939 рр.), джерела біографічного характеру (послужні (формулярні) списки та особові справи, переліки працівників підприємств, установ, та організацій, учасників релігійних спільнот, учнів, документація про стосунок до військової служби, фільтраційні і архівно-слідчі справи). Опис цих джерел подамо далі.

Виконання генеалогічного дослідження потрібно починати з наявної інформації, це такі дані як рік, дата народження одного з членів сім’ї. Спершу шукаємо актовий запис про народження особи в метричній книзі за місцем її народження.

У разі віднайдення цієї інформації можна дізнатися імена та по батькові батьків. Для того, щоби віднайти інших членів родини (сестер та братів) потрібний уважний перегляд рік за роком актових матеріалів про народження та смерті. Актові записи про шлюб потрібно переглянути для визначення років народження та місця приписки (можливого місця народження) батьків або дівочого прізвища матері.

Потрібно звертати увагу на всіх людей з таким самим прізвищем, оскільки вони можуть виявитися родичами).

Значно прискорить та полегшить виконання дослідження перегляд посімейних списків та ревізьких «сказок». За наведеними у цих документах роками смерті та народження можливо, уникаючи повного перегляду усіх наявних метричних книг, знайти потрібні актові записи. У разі, якщо деякі дані. потрібні для проведення дослідження відсутні, то можна звернутися до інших документів. Характеристику цих документів подамо у наступних статтях.

Для полегшення побудови родоводу рекомендується складати генеалогічні картки на кожну знайдену людину[4].

### 1.2.2 Закономірності успадкування генетичних захворювань за законами Менделя

У дослідах Менделя при схрещуванні сортів гороху, які мали жовті й зелені насіння, все потомство (тобто гібриди першого покоління) виявилося з жовтим насінням. При цьому не мало значення, з якого саме насіння (жовтого або зеленого) виросли материнські (батьківські) рослини. Отже, батько й мати в рівній мірі здатні передавати свої ознаки потомству.

Аналогічні результати були виявлені і в дослідах, в яких до уваги бралися інші ознаки. Так, при схрещуванні рослин з гладкими і зморшкуватим насінням все потомство мало гладкі насіння. При схрещуванні рослин з пурпуровими і білими квітками у всіх гібридів виявилися лише пурпурові пелюстки квіток.

Виявлена ​​закономірність отримала назву перший закон Менделя, або закон одноманітності гібридів першого покоління. Стан (алель) ознаки, який проявляється в першому поколінні, отримав назву домінантного, а стан (алель), який в першому поколінні гібридів не проявляється, називається рецесивним. «Задатки» ознак (за сучасною термінологією - гени) Г. Мендель запропонував позначати буквами латинського алфавіту. Стану, що належать до однієї пари ознак, позначають однією і тією ж буквою, домінантний алель великою, а рецесивний маленькою.

При схрещуванні гетерозиготних гібридів першого покоління між собою (самозапилення або споріднене схрещування) у другому поколінні з'являються особини як з домінантними, так і з рецесивними станами ознак, тобто виникає розщеплення, яке відбувається в певних відносинах. Так, в дослідах Менделя на 929 рослин другого покоління виявилося 705 з пурпуровими квітками і 224 з білими. Під час експерименту, в якому враховувався колір насіння, із 8023 насінин гороху, отриманих у другому поколінні, було 6022 жовтих і 2001 зелених, а з 7324 насіння, щодо яких враховувалася форма насінини, було отримано 5474 гладких і 1850 зморшкуватих. Виходячи з отриманих результатів, Мендель прийшов до висновку, що в другому поколінні 75% особин мають домінантний стан ознаки, а 25% - рецесивний (розщеплення 3: 1). Ця закономірність отримала назву другого закону Менделя, або закону розщеплення.

Згідно з цим законом і висловлюючись сучасною термінологією, можна зробити наступні висновки:

* алелі гена, перебуваючи в гетерозиготному стані, не змінюють структуру один одного;
* при дозріванні гамет у гібридів утворюється приблизно однакове число гамет з домінантними і рецесивними алелями;
* при заплідненні чоловічі і жіночі гамети, що несуть домінантні і рецесивні алелі, вільно комбінуються.

При схрещуванні двох гетерозигот («Аа»), у кожній з яких утворюється два типи гамет, необхідно очікувати чотири можливі поєднання. Яйцеклітина з алелем «А» може бути запліднена з однаковою часткою ймовірності як сперматозоїдом з алелем «А», так і сперматозоїдом з алелем «а»; і яйцеклітина з алелем «а» - сперматозоїдом або з алелем «А», або алелем «а». У результаті виходять зиготи «АА», «Аа», «Аа», «аа» або «АА», «2Аа», «аа».

За зовнішнім виглядом (фенотипом) особини «АА» і «Аа» не відрізняються, тому розщеплення виходить у співвідношенні 3: 1. За генотипу особини розподіляються у співвідношенні «1АА»: «2Аа»: «аа». Зрозуміло, що якщо від кожної групи особин другого покоління отримувати потомство лише самозапиленням, то перша «АА» і остання «аа» групи (вони гомозиготні) будуть давати лише одноманітне потомство (без розщеплення), а гетерозиготні «Аа» форми будуть давати розщеплення у співвідношенні 3: 1.

Таким чином, другий закон Менделя, або закон розщеплення, формулюється так: при схрещуванні двох гібридів першого покоління, які аналізуються за однією альтернативною парі станів ознаки, у потомстві спостерігається розщеплення за фенотипом у співвідношенні 3: 1 і за генотипом у співвідношенні 1: 2: 1.

Вивчаючи розщеплення при дигібридному схрещуванні, Мендель звернув увагу на таку обставину. При схрещуванні рослин з жовтим гладким (ААВВ) і зеленим зморшкуватим (ааbb) насінням у другому поколінні з'являлися нові комбінації ознак: жовті зморшкуваті (ааbb) і зелені гладкі (АаВb), що не зустрічалися в початкових формах. З цього спостереження Мендель зробив висновок, що розщеплення за кожною ознакою відбувається незалежно від другої ознаки. У цьому прикладі форма насіння успадковувалась незалежно від їх забарвлення. Ця закономірність отримала назву третього закону Менделя, або закону незалежного розподілу генів.

Третій закон Менделя формулюється так: при схрещуванні гомозиготних особин, які відрізняються за двома (або більше) ознаками, у другому поколінні спостерігається незалежне успадкування і комбінування станів ознак, якщо гени, які їх визначають, розташовані в різних парах хромосом. Це можливо тому, що під час мейозу розподіл (комбінування) хромосом у статевих клітинах при їх дозріванні йде незалежно і може привести до появи потомства з комбінацією ознак, відмінних від батьківських і прабатьківських особин[5].

## 1.3 Аналіз переваг та недоліків існуючих аналогів інформаційної системи «Генеалогічне дерево родини»

### 1.3.1 Програмний продукт «Family Tree Builder»

«Family Tree Builder» − проста зручна у використанні программа, російськомовна. Дозволяє вибудувати особисте генеалогічне дерево та має можливість збереження його в зручних графіках або звітах. Можна додавати до кожного учасника сімейного дерева дату народження або смерті, його фотографії, контактні дані, стать та інші важливі дані. До кожної персони можна додати чоловіка/дружину, дітей, батьків, братів або сестер. Створене дерево можна зберегти у вигляді графіка або звіту з подальшою можливістю друку для зручного перегляду[6]. Інтерфейс програми показана на рисунку 1 та 2.

Можливості програмного продукту:

* експорт свого дерева в Excel;
* створення графіків і звітів;
* збереження графіків і звітів в зручних форматах і можливість друку;
* синхронізація з сайтом розробника для публікації родоводу;
* пошук родичів з понад 65 мільйонів зареєстрованих користувачів;
* додавання точної інформації до кожної персони (ПІБ, дата народження або смерті, інформація про подружжя або дітей, фотографії та ін.).

Переваги:

* потужна програма, що дозволяє створювати сімейне дерево;
* простий і зручний інтерфейс;
* можливість скачати безкоштовно Family Tree Builder;
* наявність російськомовної локалізації.

Недоліки:

* одномовний інтерфейс;
* безкоштовна версія має обмежений функціонал.

Інтерфейс програмного продукту зображено на рисунку 1.1 та рисунку 1.2.

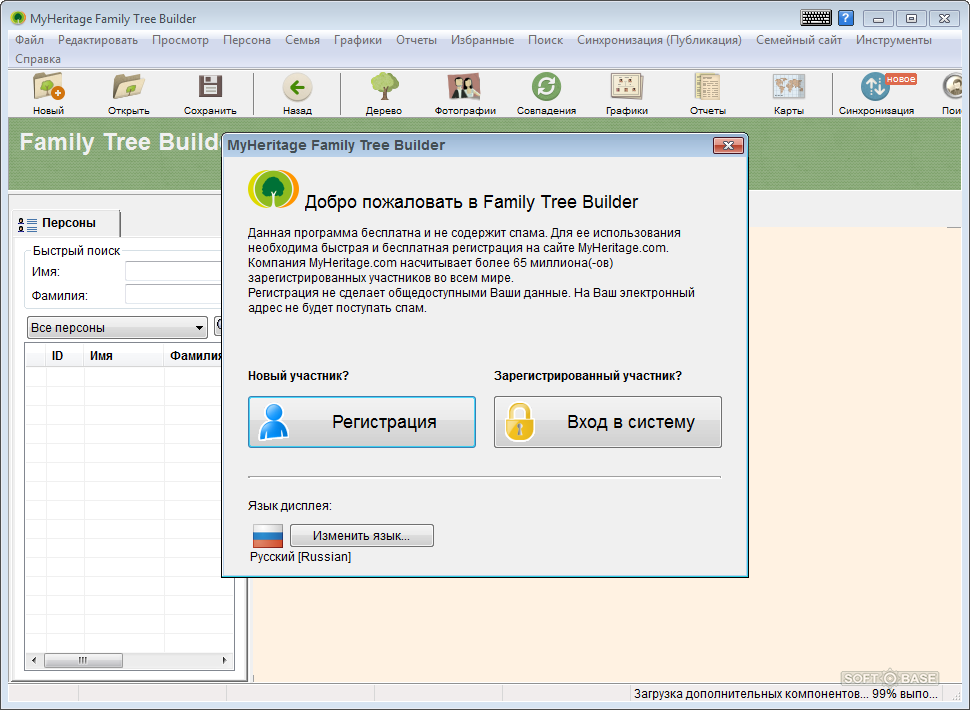


Рисунок 1.1 – Інтерфейс програми «Family Tree Builder»

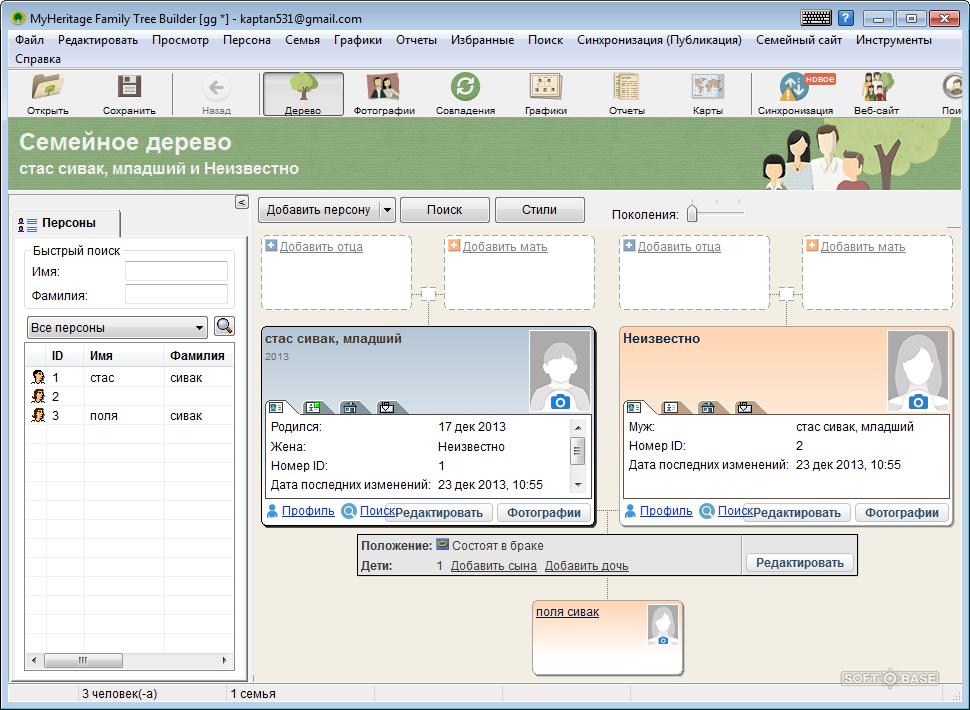


Рисунок 1.2 – Інтерфейс програми «Family Tree Builder»

### 1.3.2 Програмний продукт «SimTree»

Додаток допоможе візуально уявити генеалогічне дерево і внести коротку інформацію про кожну персони, включаючи фотографії. Інтерфейс простий і зрозумілий користувачам. Саме тому SimTree можна використовувати спільно з дітьми в навчальних цілях[7]. Інтерфейс програми зображено на рисунку 1.3.

Можливості програмного продукту:

* побудова генеалогічного дерева;
* табличне відображення інформації про кожну особу;
* додавати особисті дані до карток;
* автоматичне визначення родинних зв'язків при додаванні нових членів сім'ї.

Переваги:

* простий інтерфейс;
* зберігання практично необмеженого кількість даних;
* можливість прикріплювати файли.

Недоліки:

* обмежений функціонал;
* єдина схема візуального відображення спорідненості.

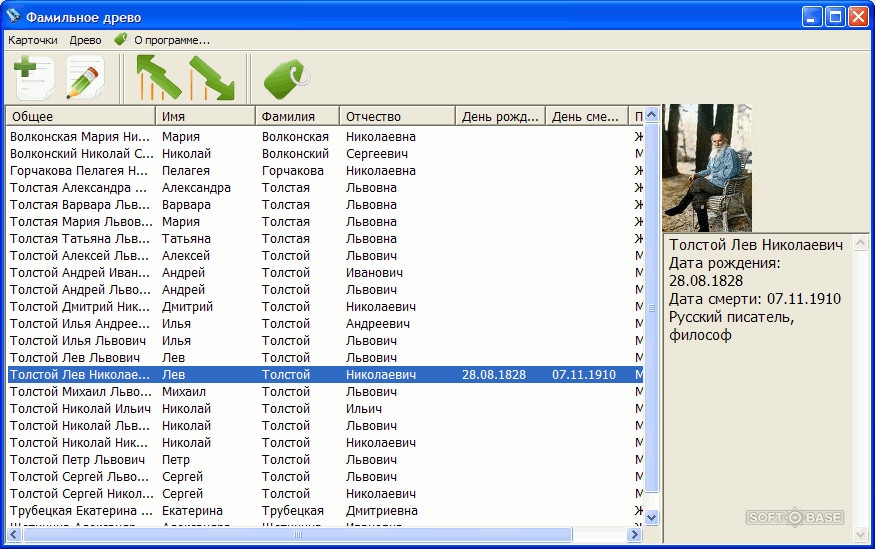


Рисунок 1.3 – Інтерфейс програми «SimTree»

### 1.3.3 Програмний продукт «Genopro»

Програма дозволяє швидко створити сімейне дерево навіть не дуже досвідченим користувачам. Завдяки наявності «гарячих клавіш» ви можете легко додати потрібних вам людей і інформацію про них. Програма умовно-безкоштовна: безкоштовна версія має обмежений термін дії (15 днів) і обмежена кількість персон для додавання (до 25 осіб). Щоб підвищити функціонал програми, необхідно буде купити ліцензійний ключ[8]. Інтерфейс програми вказаний на рисунках 1.4 та 1.5.

Можливості програмного продукту:

* наявність вбудованого генератора звітів;
* всі дані доступні для перегляду і редагування в табличному форматі;
* створення призначених для користувача вкладок даних для введення додаткової інформації;
* додавання інформації про відносини між людьми (наприклад, закоханість або конфлікт у відносинах);
* збереження резервної копії онлайн;
* відправка створеної родоводу іншим родичам по e-mail;
* друк створеної родоводу.

Переваги:

* програма має російськомовну локалізацію;
* наявність «гарячих клавіш»;
* можливість безкоштовного завантаження «Genopro»;
* експорт в різні формати, наприклад XML, Gedcom або TXT;
* імпорт сімейного дерева в форматі Gedcom, створеного в аналогічних програмах.

Недоліки:

* обмежений термін дії безкоштовної версії (15 днів);
* обмежена кількість створених персон в пробної версії (25 осіб).

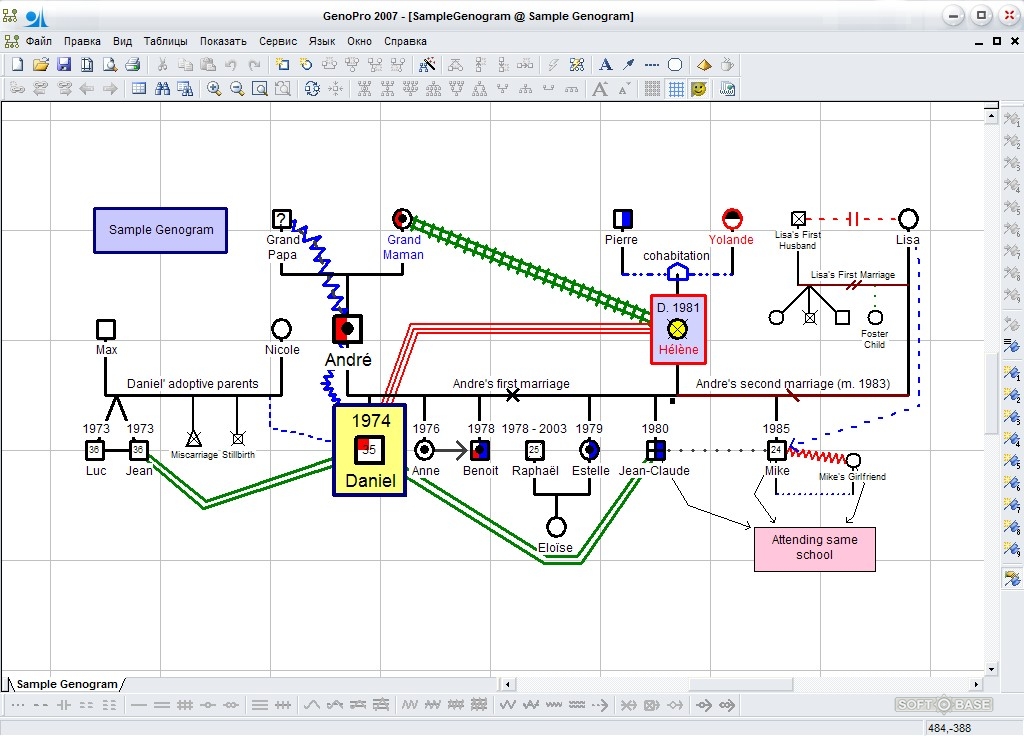


Рисунок 1.4 – Інтерфейс програми «Genopro»

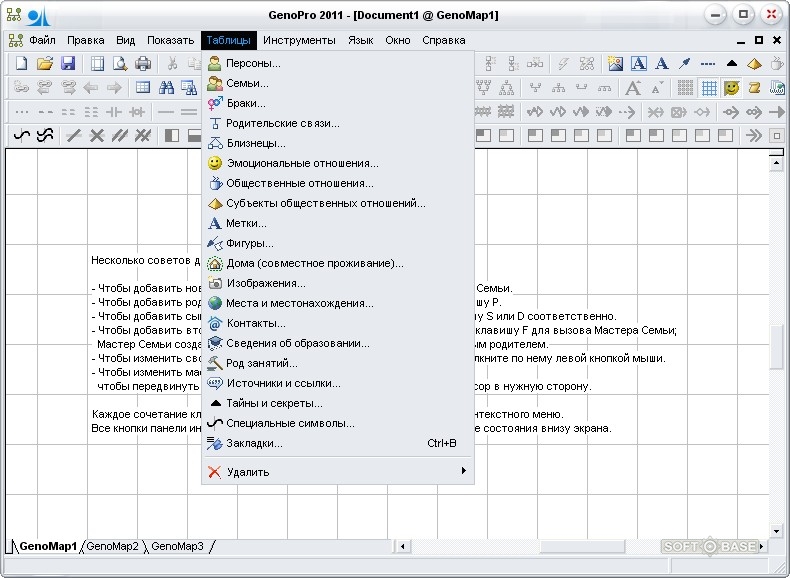


Рисунок 1.5 – Інтерфейс програми «Genopro»

### 1.3.4 Програмний продукт «RootsMagic Essentials»

Програма «RootsMagic Essentials» одна з найстаріших, її творець Bruce Buzbee написав першу версію ще в далекому 1987 році. З тих пір вона звичайно неодноразово змінювалася і допрацьовувалася і на даний момент представляє дуже симпатичну і функціональну платформу, що володіє масою переваг[8]. Інтерфейс програми показана на рисунку 1.6 та 1.7.

Можливості програмного продукту:

* складання генеалогічного дерева за допомогою зручної візуальної схеми;
* дослідження предків за допомогою спеціальних алгоритмів;
* прив’язку людей до місцевості;
* пошук інформації про осіб у відкритих джерелах.

Переваги:

* простий, красивий і доступний інтерфейс;
* широкий функціонал;
* можливість публікувати своє дерево онлайн і пов'язувати його з іншими.

Недоліки:

* програма не підтримує інші мови;
* багато важливих функцій доступні тільки в платній версії.

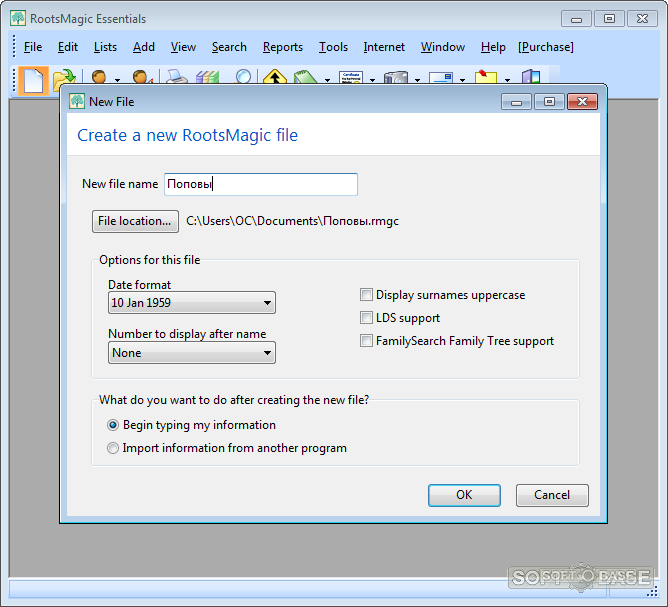


Рисунок 1.6 – Інтерфейс програми «RootsMagic Essentials»

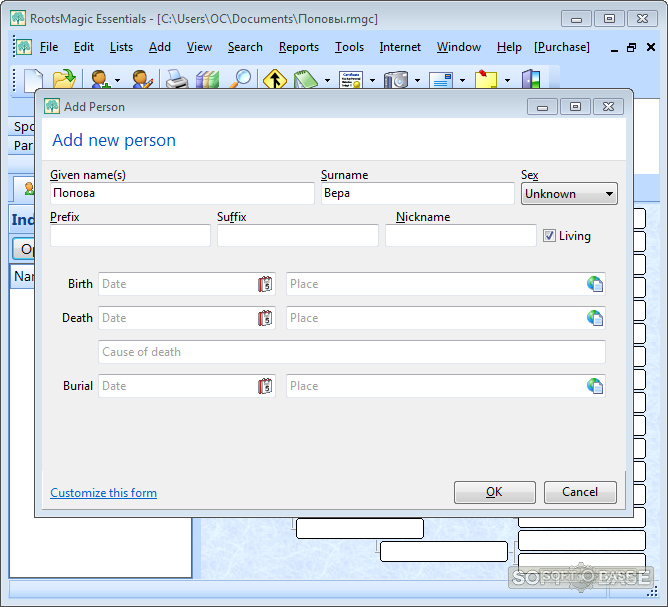


Рисунок 1.7 – Інтерфейс програми «RootsMagic Essentials»

## 1.4 Постановка задачі на розробку

Інформаційна система генеалогічного дерева родини повинна відповідати вимогам та виконувати такі функції, як:

* реєстрація нового користувача;
* вхід зареєстрованого користувача;
* створення генеалогічного дерева для користувача;
* додавання родичів до дерева користувача;
* можливість доручення до друзів;
* можливість дозволу перегляду дерева іншим користувачам;
* автоматичне формування генеалогічного дерева;
* ведення статистичного аналізу даних по родичам;
* можливість копіювання інформації із інших дерев;
* додавання нових генетичних захворювань;
* можливість знаходження ймовірності захворювання спадковою хворобою нащадка.

Також вся система повинна працювати завдяки Web-API, адже це дозволить поширити її на всі існуючі платформи. Всі дані які повертатимуться із сервера повинні бути у форматі JSON. Для забезпечення захисту особистих даних користувача, потрібно реалізувати шифрування його паролю для входу.

Для правильного визначення ймовірності прояву спадкових захворювань необхідно вказати достовірну інформацію про хворобу таку, як можливість передачі її жінкам або чоловікам, а також її форму прояву у пращура домінантна чи рецесивна.

## Висновки до першого розділу

Було вивчено конкретну область, для врахування особливостей, якими буде володіти дана інформаційна система.

Проаналізувавши опрацьовану літературу, можна сказати, що генеалогія та генетика посідає досить значне місце у розвитку сімейної історії. Використання даного програмного продукту дозволить скласти умови для більш ефективного проведення збору інформацій про власну родину.

У першому розділі розглянуто аналоги для розроблюваної інформаційної системи. Існує велика кількість програмного забезпечення для побудови генеалогічного дерева родини, але немає такої системи , яка передбачає наявність у осіб певного спадкового захворювання. Тому розробка даної інформаційної системи є досить актуальна.

Також під час написання першого розділу було проаналізовано фактичний стан задачі, що розв’язується, на основі матеріалів, які характеризують об’єкт проектування; містить теоретичний виклад важливих аспектів проблеми, що досліджується, та критичний огляд наукової літератури і періодичних джерел з визначеної тематики.

Сформовано технічне завдання, в якому передбачено переваги та недоліки існуючих аналогів, також включені ті функції, які не було знайдено у аналогів, що і робить даний програмний продукт унікальним.

# 2 АНАЛІЗ І ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ»

У даному розділі розглядаються різновиди вимог, поданих для програмних систем. Обговорюються їх характеристики та пояснюються основні процеси для роботи з вимогами: формування вимог, аналіз вимог, управління змінами у вимогах.

Розуміння і документування цього називається аналізом вимог. Основна задача визначення вимог – знайти, обговорити та зафіксувати те, що дійсно потребується від системи в формі, зрозумілій і клієнтам, і членам команди розробників

## 2.1 Моделювання предметної області системи

Модель предметної області – це візуальне подання концептуальних класів або об’єктів реального світу в термінах предметної області. Такі моделі також називають концептуальними моделями, моделями об’єктів предметної області, або об’єктними моделями аналізу. Це свого роду словник основних абстракцій, тобто найважливіших іменників у просторі задачі. Іменники, які описують поняття з предметної області, називають доменними об’єктами. На самому початку аналізу і проектування необхідно створити модель предметної області, в якій всі доменні об’єкти будуть зображені на одній великій діаграмі класів.

У термінології UML модель предметної області – це, по суті справи, діаграма класів. Зазвичай в цій моделі опускається велика частина деталей, зокрема атрибути та операції класів. Ось чому можна вважати, що модель предметної області є зведеною діаграмою класів, як візуальний словник важливих абстракцій або словник предметної області [10].

На етапі моделювання предметної області можна створити діаграму класів, в якій будуть відображені основні ідея та логіка роботи системи, без поглибленого опису атрибутів і методів класів та їх взаємодії між собою. Це дозволить відобразити головну логічну концепцію системи, що посприяє подальшому проектуванню даної системи.

Основні класи програми представлено на рис. 2.1 та рис. 2.2.

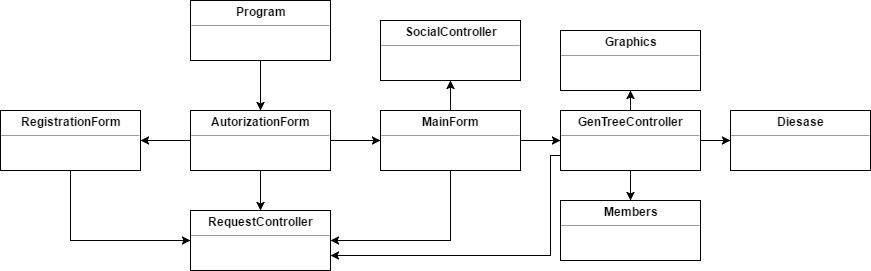


Рисунок 2.1 ­– Основні класи клієнтської частини інформаційної системи генеалогічного дерева родини

Класи «Program» та «MainForm», зображені на рис. 2.1 це головні класи програми, що відповідають за керування програмою та перегляд результатів її роботи.

Класи «AutorizationForm» та «RegistrationForm» – це класи, що відповідають за авторизацію та реєстрацію користувача відповідно. У класі «SocialController» реалізовані можливості зв’язку з іншими користувачами програми. Для генерації дерева та різноманітних графіків використовується класи «GenTreeController» та «Graphics», щоб це стало можливим їм потрібні дані, що знаходяться у класах «Disease» та «Members». Всі дані програма отримує з сервера за допомогою утиліти, що описана в класі «RequestController».

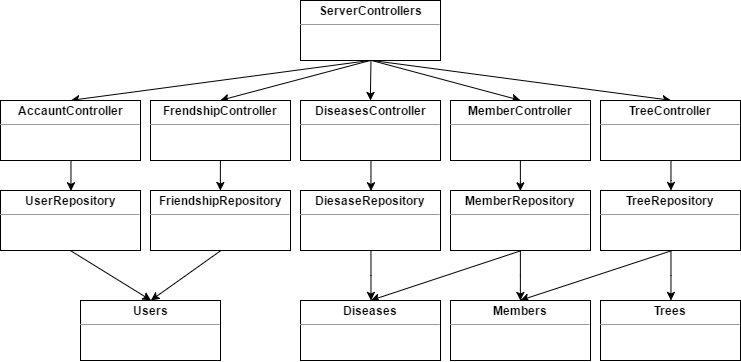


Рисунок 2.2 ­– Основні класи серверної частини інформаційної системи генеалогічного дерева родини

Класи, що зображені на рис 2.2 відповідають за збереження та обробку даних на сервері. Для реалізації API використовується клас «ServerControllers» у ньому описані всі методи зв’язку з клієнтською частиною.

Класи «AccauntController», «FriendshipController», «DiseasesController», «MemberController», «TreeController» ­– це класи для обробки запитів які надходять від клієнта. За зв’язок з даними, що знаходиться на сервері відповідають класи «UserRepository», «FriendshipRepository», «DiseaseReposito», «MemberRepository» та «TreeRepository».

## 2.2 Формулювання та аналіз вимог до інформаційної системи

Розширення сфери застосування інформаційних технологій і постійно зростаюча складність інформаційних систем привели до підвищення значущості перед проектного аналізу та робіт з бізнес-аналізу в цілому. Одним з ключових завдань у даній області є виявлення вимог до побудови систем.

Вимоги для інтелектуальних технологій є концентрованим відображенням потреб зацікавлених у створенні системи замовників, користувачів і розробників. Вони виникають у процесі збору первинної інформації, обговорення проекту системи, аналізу предметної області та мають тенденцію бути модифікованими під час виконання проекту [11].

Формування та аналіз вимог полягають у визначенні потреб та умов, які висуваються щодо нового чи зміненого продукту, враховуючи можливо конфліктні вимоги різних замовників, таких як користувачі.

Аналіз вимог є критичним для успішної розробки проекту. Вимоги мають бути задокументованими, вимірними, протестованими, пов’язаними з бізнес-потребами, і описаними з рівнем деталізації, достатнім для конструювання системи. Вимоги можуть бути архітектурними, структурними, поведінковими, функціональними, та не функціональними [10].

Основні вимоги для створення системи:

* можливість отримувати дані про генеалогічні дерева з серверу;
* наявність сервера, який дозволятиме отримувати всі дані;
* застосування розробленого алгоритму побудови генеалогічного дерева та прогнозування прояву генетичних захворювань;
* можливість графічного представлення генеалогічного дерева;
* можливість створення звітів та графіків за даними з генеалогічного дерева;
* наявність зручного інтерфейсу користувача;
* простота освоєння програми та роботи з нею;
* відкритість для модифікацій і доповнень наступними версіями і модулями;
* можливість додавання доручення до друзів та перегляд дерев друзів.

Аналіз вимог слугує мостом між підготовкою, плануванням і проектуванням програмного забезпечення.

Реалізація процедур цього етапу проектування відбувається в процесі побудови відповідних діаграм мовою UML або іншими мовами, що використовують системи автоматизованого проектування. Результати процедур із формулювання та аналізу вимог оформляються у вигляді діаграми прецедентів (варіантів використання, Use Case), діаграми діяльності, діаграми взаємодій (діаграми послідовностей і комунікації ) та їх описів.

### 2.2.1 Формування вимог за допомогою діаграми прецедентів

Діаграми прецедентів відіграють важливу роль не тільки у комунікації між збирачами вимог до проекту і потенційними користувачами. Діаграми прецедентів дописані бізнес логікою і детальними специфікаціями прецедентів, як джерельна інформація, успішно використовують учасники розробки проекту на всіх його фазах.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами [12].

Візуальне моделювання в UML можна подати як деякий процес спуску від найбільш загальної і абстрактної концептуальної моделі початкової системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення таких цілей спочатку будується модель у формі так званої діаграми прецедентів, яка описує функціональне призначення системи або, іншими словами, те, що система робитиме в процесі свого функціонування.

Таким чином, діаграма прецедентів є вихідним концептуальним уявленням або концептуальною моделлю системи в процесі її проектування і розробки. Основними елементами діаграми прецедентів є дійові особи, варіанти використання і відношення між ними. Дійова особа – це роль, яку користувач грає по відношенню до системи.

На рис. 2.3 відображено узагальнене представлення функціоналу системи з використанням таких прецедентів:

* вибір даних;
* вирахування ймовірності захворювань;
* перегляд результатів ймовірності;
* формування звітів та графіків;
* отримання даних з серверу;
* формування генеалогічного дерева.

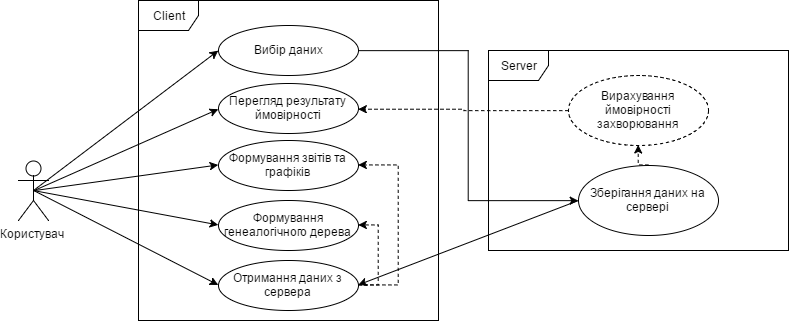


Рисунок 2.3 ­– Діаграма прецедентів інформаційної системи «Генеалогічне дерево родини»

Розроблювана програма буде універсальною та матиме можливість отримувати дані із сервера. Всі головні операції відбуватимуться на сервері, в разі необхідності їх можна змінити, або дописати нові утиліти. Для зв’язку з сервером будь-яка клієнтська програма може використовувати API, що надає змогу даному продукту бути кросплатформенним.

Отже, цей прецедент – це узагальнене представлення функціоналу як сервера так і клієнта, завдяки ньому дана система зможе отримувати потрібні дані а також представляти їх у зручній для користувача формі.

З рисунку 2.3 видно, що проектована система представляється у вигляді множини сутностей, взаємодіючих із системою за допомогою так званих варіантів використання. Варіант використання служить для опису сервісів, що система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, чинений системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

З вище згаданих варіантів використання, користувач може обрати дані про родичів , які будуть розраховувати ймовірність захворювання у їх нащадків (якщо в цьому є потреба), та надсилати серверу запит обробляти ці дані. Після цього користувач може передивитись результати проведених обрахунків. Також він може створити різні графіки, що відображатимуть статистику його генеалогічного дерева.

Деякі прецеденти залежать від інших прецедентів. Наприклад, якщо необхідно відразу сформувати звіт та графіки , програма все одно має сформувати генеалогічне дерево, а для цього їй потрібні самі дані, які в свою чергу беруться з серверу.

### 2.2.2 Формування вимог за допомогою діаграми діяльності

Діаграма діяльності – в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності.

Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку [13].

Діаграма діяльності системи автоматизації аналізу господарської діяльності підприємства зображена на рис. 2.4.

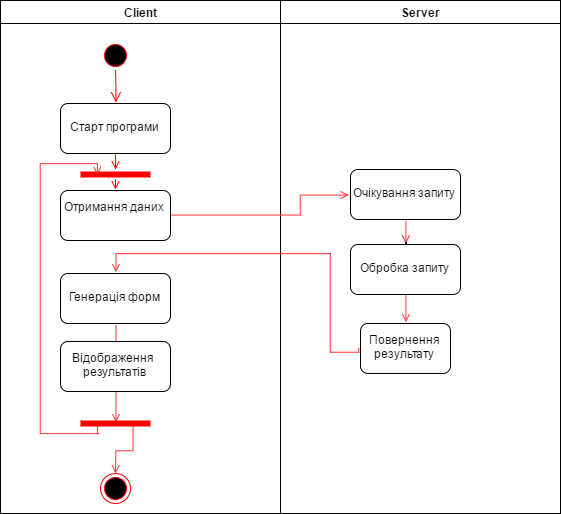


Рисунок 2.4 – Діаграма діяльності системи «Генеалогічне дерево родини»

На рис. 2.4 зображений алгоритм, який виконується протягом роботи даного програмного продукту.

Застосування діаграм діяльності для реалізації варіантів використання не дуже наближує до появи цільового артефакту ­­– програмного коду, але може привести до більш глибокого розуміння даної задачі. Діаграма діяльності дозволяє будь-кому, хто виконує даний процес, обирати порядок дій. Іншими словами, діаграма тільки встановлює правила обов’язкової послідовності дій, за якими потрібно слідувати.

### 2.2.3 Аналіз вимог за допомогою діаграми послідовностей

У мові UML взаємодія елементів розглядається в інформаційному аспекті їх комунікації, тобто взаємодіючі об’єкти обмінюються між собою деякою інформацією. При цьому інформація приймає форму закінчених повідомлень. Іншими словами, хоча повідомлення і має інформаційний зміст, воно набуває додаткової властивості робити направлений вплив на свого одержувача. Будь-які види інформаційної взаємодії між елементами системи повинні бути зведені до відправки і прийому повідомлень між ними.

Для моделювання взаємодії об’єктів в мові UML використовуються відповідні діаграми взаємодії. Кажучи про ці діаграми, мають на увазі два аспекти взаємодії. По-перше, взаємодії об’єктів можна розглядати в часі, і тоді для представлення тимчасових особливостей передачі і прийому повідомлень між об’єктами використовується діаграма послідовності. По-друге, можна розглядати структурні особливості взаємодії об’єктів. Для представлення структурних особливостей передачі і прийому повідомлень між об’єктами використовується діаграма кооперації.

На діаграмі послідовності зображуються виключно ті об’єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об’єктами. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об’єктів в часі.

Діаграма послідовностей інформаційної системи генеалогічного дерева родини зображена на рис. 2.5, з якого видно, як саме в даній системі об’єкти взаємодіють між собою у часі. Діаграма відображає роботу системи у випадку, коли користувач виконує основні дії, покладені за мету даної роботи, а саме – отримання з серверу даних та генерація генеалогічного дерева, вирахування ймовірності прояву генетичних захворювань, перегляд результату ймовірностей та генерування графіків відносно отриманих даних.

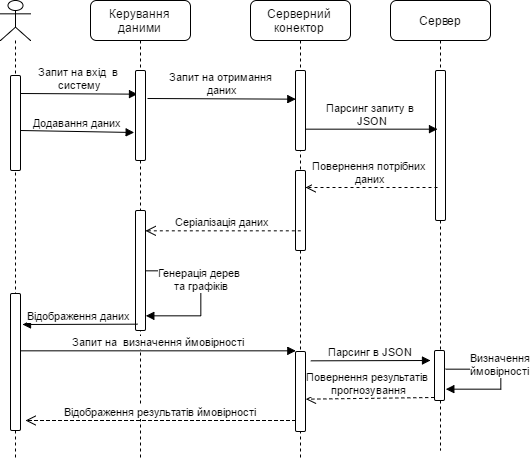


Рисунок 2.5 – Діаграма послідовностей інформаційної системи «Генеалогічне дерево родини»

Окрім цього, на рисунку 2.5 відображена також робота з сервером через серверний «конектор», що слугує засобом обміну даними між клієнтською програмою та сервером. Дана логіка роботи через «конектор» надає програмі властивостей універсальності та в майбутньому дозволить працювати з багатьма програмами, що писатимуться на будь-якій мові програмування та під будь-яку платформу.

### 2.2.4 Функціонування системи за допомогою діаграми комунікації

Діаграма комунікації моделює взаємодії між об’єктами або частинами в термінах впорядкованих повідомлень. Комунікаційні діаграми представляють комбінацію інформації, взятої з діаграм класів, послідовності і варіантів використання, описуючи одночасно і статичну структуру і динамічну поведінку системи.

Діаграма комунікації – діаграма, на якій зображуються взаємодії між частинами композитної структури або ролями кооперації. На відміну від діаграми послідовності, на діаграмі комунікації явно вказуються відносини між об’єктами, а час як окремий вимір не використовується (застосовуються порядкові номери викликів).

Комунікаційні діаграми мають вільний формат упорядкування об’єктів і зв’язків як в діаграмі об’єктів. Щоб підтримувати порядок повідомлень при такому вільному форматі, їх хронологічно нумерують. Читання діаграми комунікації починається з повідомлення 1 і триває по напрямку пересилання повідомлень від об’єкта до об’єкта. Діаграма комунікацій даної системи зображена на рис. 2.6.

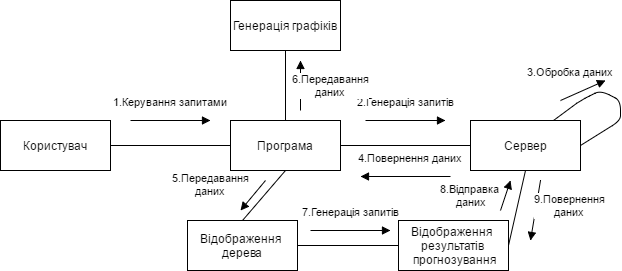


Рисунок 2.6 – Діаграма комунікацій системи «Генеалогічне дерево родини»

Як видно з рис. 2.6, діаграма комунікації відображає взаємодію ролей або об’єктів у процесі функціонування системи, описують обмін даними (повідомленнями) між різними учасниками взаємодії. Такі діаграми моделюють сценарії поведінки системи. Замість того щоб малювати кожного учасника у вигляді лінії життя і показувати послідовність повідомлень, розташовуючи їх по вертикалі, як це робиться в діаграмах послідовності, комунікаційні діаграми допускають довільне розміщення учасників, дозволяючи малювати зв'язки, що показують відносини учасників, і використовувати нумерацію для представлення послідовності повідомлень.

Об’єкти взаємодіють один з одним за допомогою зв’язків – каналів для передачі повідомлень. Зв’язок між парою об’єктів розглядається як екземпляр асоціації між їх класами. Іншими словами, зв’язок між двома об’єктами існує тільки тоді, коли є асоціація між їх класами.

## 2.4. Об’єктно-орієнтоване проектування інформаційної системи

Для створення інформаційної системи була обрана об’єктно-орієнтована технологія проектування.

Метод об’єктно-орієнтованого проектування базується на: моделі побудови системи як множини об’єктів абстрактного типу даних, модульній структурі програм, спадному проектуванні, що використовується при виділенні об’єктів.

В об’єктно-орієнтованому підході в якості так званих блоків використовуються об’єкти, що містять свої власні коди та дані. Будова програм при об’єктно-орієнтованому підході показується графом взаємодії об’єктів, а не деревом ієрархії, як це відбувається в структурному проектуванні.

За основу структурного підходу взято структуризацію і декомпозицію навколишнього світу. Задача будь-якої складності розділяються на під задачі, а ті, у свою чергу, розділяються далі, аж поки кожна під задача не стане простою, відповідною підпрограмі, що виконує тільки одну дію. Методи структурного проектування застосовують модулі в якості блоків програми, а структура програми представлена ієрархією підпорядкованості модулів.

Можна сказати використання об’єктів при розробці систем фактично зробило можливим перехід від порівняно обмеженої бази, якою є алгоритм, до більш ширших структурних та поведінкових аспектів представлення системи, що і надало змогу наповнити змістом сам процес аналізу та розробки.

Виокремлюють такі основні плюси об’єктно-орієнтованого проектування:

* наявність різнопланових модельних поглядів;
* застосування абстракції предметної області;
* стабільність в представленні змін;
* покращення можливості для повторного використання;
* можливість управління масштабністю;
* краща підтримка аспектів надійності;
* успадковується підтримка паралелізму. [11].

### 2.4.1 Архітектурне проектування

Формування архітектури – перший і основний крок у вирішенні завдання проектування, що закладає фундамент уявлення програмної системи, здатної виконувати весь спектр детальних вимог.

Створення архітектури – це проектування на найвищому рівні (логічна архітектура). Логічна архітектура описує систему в термінах її принципової організації у вигляді пакетів, програмних класів і підсистем. Вона називається логічною, оскільки не визначає способи розгортання цих елементів у різних операційних системах або на фізичних комп’ютерах в мережі (це відноситься до архітектури розгортання).

Архітектурне проектування полягає у визначенні головних структурних особливостей системи, яку будують, а саме: складу компонент, способів їхньої композиції, обмежень на їхні з’єднання.

Етап архітектурного проектування інформаційної системи шляхом опису процесу об’єктно-орієнтованої декомпозиції системи до рівня переліку підсистем та їх зв’язку, опису її логічної структури у вигляді компонентів (діаграма компонентів UML) [13].

#### 2.4.1.1 Діаграма пакетів

Діаграма пакетів – це структурна діаграма, в якій основними елементами є пакети і залежності між ними. Вона служить у першу чергу, для організації елементів у групи за певною ознакою з метою спрощення структури та організації роботи з моделлю системи. Класи складають структурний кістяк об'єктно-орієнтованої системи[10].

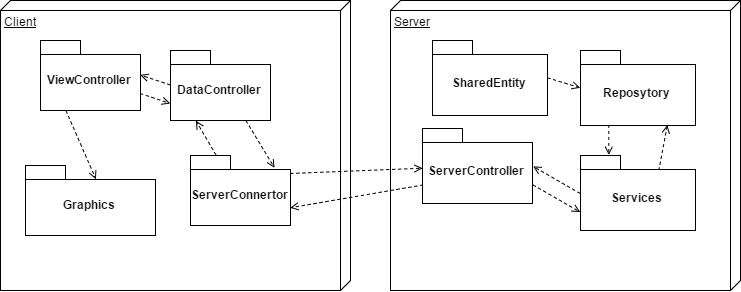
На рисунку 2.7 зображена загальна структура пакетів даної системи.

Рисунок 2.7 – Діаграма пакетів системи «Генеалогічне дерево родини»

Як можна побачити з рис 2.7, даний програмний продук поділяється на дві програми «Client», «Server». У клієнті для зв’язку з сервером використовуватиметься пакет «ServerConnector». У сервері головну роль у зв’язку з клієнтом виконуватиме пакет «ServerController». Дана архітектура дозволить як на цьому етапі розробки, так і на майбутнє, ефективно використовувати модель роботи програми та створювати додаткові компоненти для логічної будови системи.

#### 2.4.1.2 Діаграма компонентів

Діаграма компонентів показує визначення, внутрішню структуру і залежності набору компонентів. В якості фізичних компонент можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети.

Компонент – модульна і замінна частина системи, яка відповідає набору інтерфейсів і забезпечує реалізацію цього набору інтерфейсів. Реалізація компоненту завжди прихована. Компоненти системи з однаковими наборами інтерфейсів є взаємозамінними.

Той факт, що між двома компонентами завжди знаходиться інтерфейс, усуває їх пряму залежність. Компонент, що використовує забезпечений інтерфейс, функціонуватиме правильно незалежно від того, який компонент реалізує цей інтерфейс. Це дуже важливо і забезпечує гнучку заміну компонентів в інтересах розвитку системи.[10].

На рисунку 2.8 зображена діаграма компонентів даної інформаційної системи.

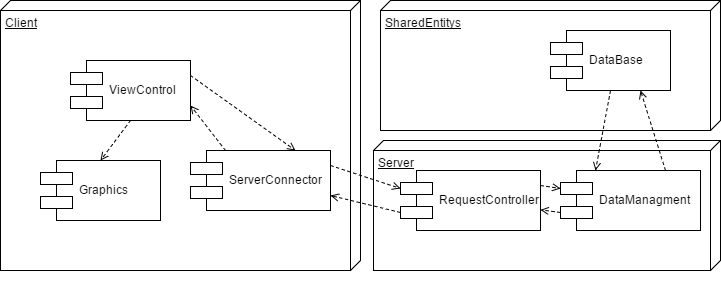


Рисунок 2.8 – Діаграма компонентів системи «генеалогічне дерево родини»

Діаграма компонентів, що зображена на рис. 2.8, показує компоненти в системі, тобто програмні підсистеми, з яких створюється архітектура, а також залежності між компонентами. Вона відображає модель роботи програми за тим самим принципом, що був описаний раніше, лише з додаванням логіки використання суспільних даних для роботи. Компонент «Data Management» виконує функцію так званої серіалізації даних, що надходять з серверу. Даний компонент дуже важливий, адже з будь-якого виду даних він може перетворити їх у такий вигляд, з яким зможе ідеально працювати наша система.

Також, компонент «SharedEntitys», що відповідає за збереження даних у базу даних, яка знаходиться на сервері.

### 2.4.2 Детальне проектування

Другим ступенем проектування, який слідує за створенням архітектури є детальне проектування. У ході цієї діяльності орієнтуються на максимальну підготовку до кодування програмної системи. Програмісти повинні отримати детальні проектні рішення, які забезпечать їх повною інформацією для створення програмного коду.

Результат етапу детального проектування повинен бути відображений діаграмами класів (що містять опис всіх операцій та атрибутів класів), які призначені для статичного моделювання об’єктів.

Діаграма класів описує типи об’єктів системи і різного роду статичні відносини, які існують між ними. На діаграмах класів відображаються також властивості класів, операції класів та обмеження, які накладаються на зв’язку між об’єктами. В UML термін функціональність застосовується в якості основного терміну, що описує і властивості, і операції класу.

Діаграми класів вважають основним засобом для представлення структури систем в термінах базових будівельних блоків і відносин між ними.[10].

#### 2.4.2.1 Діаграма класів

Діаграма класів інформаційної системи «Генеалогічне дерево родини» зображена на рис. 2.9.

Тут класи «Program» та «MainForm» відповідають за основну роботу програми – керування та перегляд інформації про генеалогічні дерева . Клас «MainForm» посилається на багато інших класів задля отримання інформації про родичів, хвороби їх аналізу та відображення у графіках. Класи «AutorizationForm» та «RegistrationForm» – це класи, що відповідають за авторизацію та реєстрацію користувача відповідно. У класі «SocialController» реалізовані можливості зв’язку з іншими користувачами програми. Для генерації дерева та різноманітних графіків використовується класи «GenTreeController» та «Graphics», щоб це стало можливим їм потрібні дані, що знаходяться у класах «Disease» та «Members». Всі дані програма отримує з сервера за допомогою утиліти, що описана в класі «RequestController».

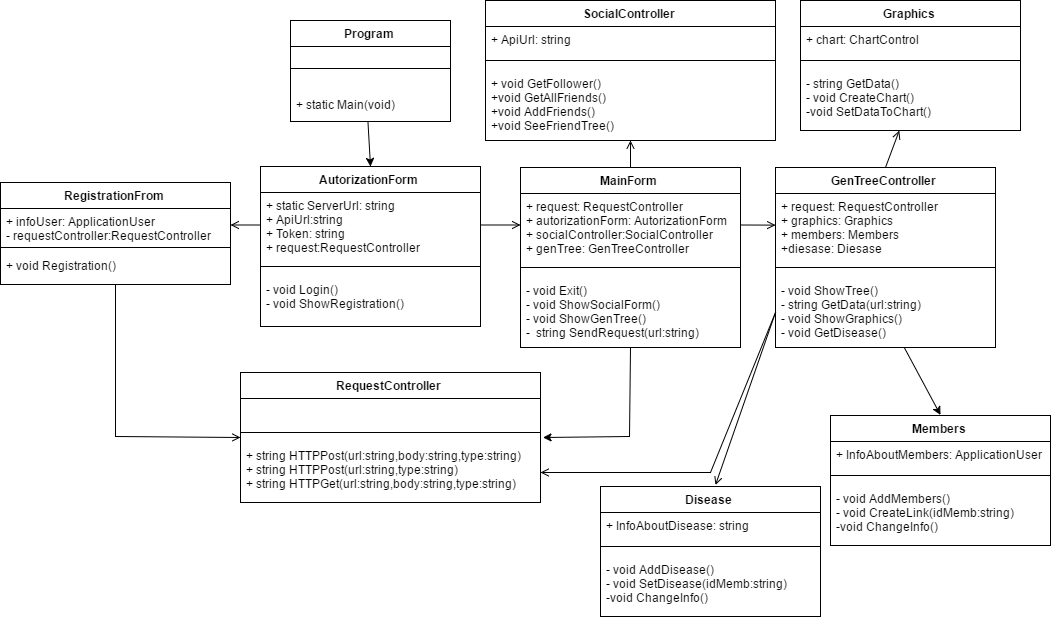


Рисунок 2.9 – Діаграма класів інформаційної системи «Генеалогічне дерево родини»

### 2.4.3 Розгортання програмної системи на апаратних засобах

Діаграми розгортання відображають відповідність конкретних програмних артефактів (наприклад, виконуваних файлів) обчислювальним вузлам (виконуючим обробку). Вони показують розміщення програмних елементів у фізичній архітектурі системи і взаємодію (зазвичай мережеву) між фізичними елементами. Діаграма розгортання дозволяє краще зрозуміти фізичну архітектуру (або архітектуру розгортання).

Графічно діаграма розгортання – це граф з вузлів (або примірників вузлів), які з’єднані асоціаціями, що показують існуючі комунікації. Вузли (або їх примірники) можуть містити артефакти (або їх примірники), що живуть і/або працюють у вузлах [10].

Діаграма розгортання інформаційної системи «Генеалогічне дерево родини» зображена на рис. 2.10.

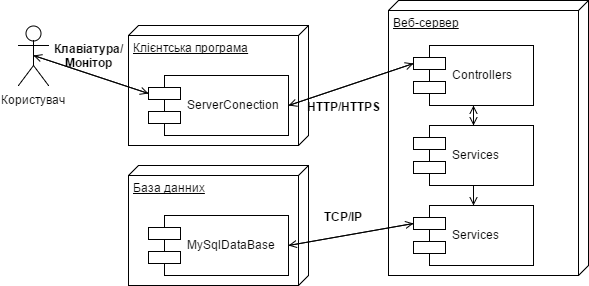


Рисунок 2.10 – Діаграма розгортання інформаційної системи «Генеалогічне дерево родини»

Як показано на рис. 2.10, артефакти з’єднуються один з одним стрілками залежностей. На цій діаграмі зображена триярусна система:

* ярус персонального комп’ютера, реалізований екземпляром «Клієнтська програма»;
* ярус бізнес-логіки сервера, представлений екземпляром «Веб - сервер»;
* ярус засобів зберігання даних, представлений екземпляром «База даних».

З рис. 2.10 видно, як розроблювана система має взаємодіяти з сервером, який в свою чергу буде розвернений окремо та матиме свої компоненти, від яких не залежить робота розроблюваної програми – сервер повертатиме потрібні дані

## 2.5 Модель поведінки інформаційної системи «Генеалогічне дерево родини»

За допомогою взаємодій зручно моделювати поведінку сукупності спільно працюючих об’єктів. Кінцевий автомат орієнтований на представлення поведінки окремого об’єкта. Як і діаграми видів діяльності, діаграми кінцевих автоматів (станів) UML відображають динамічну модель системи. За їх допомогою ілюструються події і стани об’єктів, ­ транзакцій, прецедентів, людей і т.д. Стан об’єкта визначається поточними значеннями його атрибутів (як елементарних атрибутів, так і атрибутів, що позначають інші класи).

Діаграма кінцевого автомата задає поведінку системи як цілісної, єдиної сутності, вона моделює життєвий цикл єдиного об’єкту. У силу цього автоматний підхід зручно застосовувати для формалізації динаміки окремого, важкого для розуміння блоку системи [10].

Кінцевий автомат інформаційної системи «Генеалогічне дерево родини» зображена на рис. 2.11.

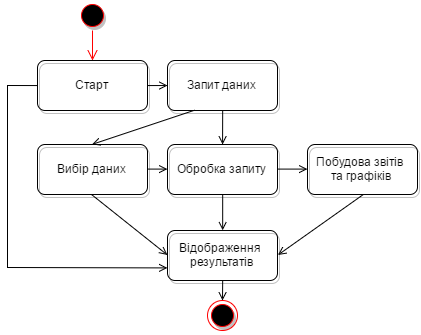


Рисунок 2.11 – Діаграма кінцевого автомату інформаційної системи «Генеалогічне дерево родини»

Ця послідовність розглядається як відповідь на події і включає реакції на ці події. Автомат задає поведінку системи як цілісної, єдиної сутності, моделює життєвий цикл єдиного об’єкта. У силу цього автоматний підхід зручно застосовувати для формалізації динаміки окремого, блоку системи.

## Висновки до другого розділу

У цьому розділі було проведено огляд різновидів вимог, поданих для програмних систем, обговорено їх характеристики та пояснено основні процеси для роботи з вимогами: формування вимог, аналіз вимог, управління змінами в вимогах. Основна задача визначення вимог – знайти, обговорити та зафіксувати те, що дійсно потребується від системи в формі, зрозумілій і клієнтам, і розробнику програмного забезпечення. Результатом аналізу вимог є документ, який зазвичай називають специфікацією вимог. Вимоги відображають те, що повинна робити система.

Також було представлено діаграми, що характеризують систему, її вимоги та основну логіку. Розроблена діаграма класів дозволяє переходити безпосередньо до наступного кроку – програмування системи. За допомоги діаграм пакетів та компонентів можливе зручне створення структури програми, а створена діаграма розгортання дозволяє бачити, які компоненти де повинні бути сформовані у програмній реалізації для подальшого використання.

# 3 РОЗРОБКА І ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ГЕНЕАЛОГІЧНЕ ДЕРЕВО РОДИНИ»

## 3.1 Огляд засобів реалізації розробки інформаційної системи «Генеалогічне дерево родини»

### 3.1.1 Архітектурний шаблон MVC

Модель-вигляд-контролер (або Модель-вид-контролер, англ. Model-view-controller, MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування (рис. 3.1). Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

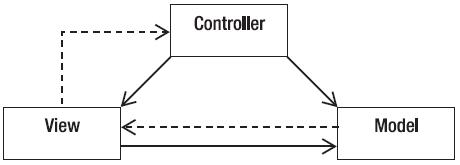


Рисунок 3.1 – Діаграма взаємодії між компонентами шаблону MVC

Архітектурний шаблон Модель-Вид-Контролер (MVC) поділяє програму на три частини. До обов’язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов’язаних областей, наприклад, різні таблиці і поля форм, в яких відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле).

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об’єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися [14].

### 3.1.2 Мова програмування C#

C# – об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов С++, Delphi, Модула і Smalltalk – С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів.

C# є дуже близьким родичем мови програмування Java. Мова Java була створена компанією Sun Microsystems, коли глобальний розвиток інтернету поставив задачу роззосереджених обчислень. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (типу вказівників без контролю виходу за межі). Для роззосереджених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп’ютера чи іншого інтелектуального пристрою.

Java набула чималої популярності, і була ліцензована також і компанією Microsoft. Але з плином часу Sun почала винуватити Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою Windows, чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду. Microsoft відмовилася піти назустріч вимогам Sun, і тому з’ясування стосунків набуло статусу судового процесу. Суд визнав позицію Sun справедливою, і зобов’язав Microsoft відмовитися від позаліцензійного використання Java.

У цій ситуації в Microsoft вирішили, користуючись своєю вагою на ринку, створити свій власний аналог Java, мови, в якій корпорація стане повновладним господарем. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, плюс врахувала досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET [15].

### 3.1.3 Microsoft Visual Studio 2017

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Версія Visual Studio 2017 включає в себе також найновіший .NET Framework 4.6 та підтримку універсальної платформи Windows 10. Суттєвою зміною даного продукту стала також підтримка багатьох цільових платформ: окрім базової Windows з’явилась можливість будувати проекти для IOS та Android [16].

### 3.1.4 Фреймворк ASP.NET

ASP.NET – технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. У цей час останньою версією цієї технології є ASP.NET 5.

ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою.

Хоча ASP.NET бере свою назву від старої технології Microsoft ASP, вона значно від неї відрізняється. Microsoft повністю перебудувала ASP.NET, ґрунтуючись на Common Language Runtime (CLR), який є основою всіх застосунків Microsoft .NET. Розробники можуть писати код для ASP.NET, використовуючи практично будь-які мови програмування, що входять у комплект .NET Framework (C#, Visual Basic.NET, і JScript.NET). ASP.NET має перевагу у швидкості в порівнянні зі скриптовими технологіями, тому що при першому зверненні код компілюється і поміщається в спеціальний кеш, і згодом тільки виконується, не вимагаючи витрат часу на парсинг та оптимізацію. [17].

### 3.1.5 Використання JSON для передачі даних

В даній роботі було використано формат даних JSON (англ. JavaScript Object Notation, укр. об’єктний запис JavaScript) – це текстовий формат обміну даними між комп’ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об’єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об’єкти [18].

## 3.2 Функціональний опис інформаційної системи «Генеалогічне дерево родини»

Функціональна схема містить інформацію про способи реалізації пристроєм заданих функцій. За такою схемою можна визначити, як здійснюються перетворення і які для цього необхідні функціональні елементи. Кожен функціональний елемент містить лише ті входи і виходи, які необхідні для його коректної роботи. Дана схема розробляється на основі [структурної схеми](http://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) для кожного блоку. В результаті з окремих функціональних елементів складається загальна функціональна схема об’єкту.

На основі функціональної схеми розробляється [основна ідея проекту, реалізація якої](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D0%BE%D0%B2%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) представлена на рис 3.2.

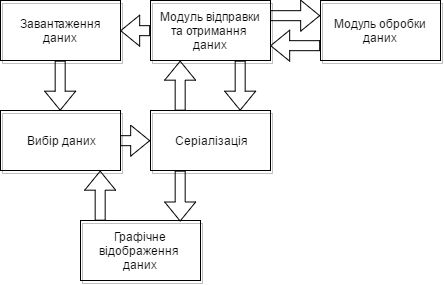


Рисунок 3.2 – Функціональна схема системи «Генеалогічне дерево родини»

Система складається із шести модулів (підсистем), що мають дев’ять потоків обміну даними. У модулях завантаження даних, серіалізації, графічного відображення та вибору даних реалізуються функції взаємодії з користувачем та координацію з робочим інтерфейсом для відображення даних про генеалогічні дерева, а також робота всієї системи загалом.

Модуль відправки та отримання даних містить в собі основні процеси роботи з сервером. Він має вхідний та вихідні потоки. Вхідний потік призначений для отримання команд запуску. За допомогою вихідного потоку відбувається формування і повернення результату в модуль серіалізації та завантаження даних. Для модуля серіалізацї один з вхідних потоків також призначений для отримання команд запуску алгоритму, який приймає дані для обробки від модулю завантаження цих даних, а вихідний – для повернення серіалізованого результату. Він є центральною ланкою взаємодії користувача з програмою.

Модуль обробки даних має один вхідний потік даних від модуля відправки та отримання даних. Цей зв’язок надає можливість передати на опрацювання зібрану інформацію і підготувати її для подальшого використання. Після того як данні готові, вони відправляться до модуля завантаження даних або серіалізації де серіалізовані дані потрапляють до модуля графічного відображення.

Модуль графічного відображення відповідає за відображення результатів обробки даних у вигляді інформації про дерева, друзів та графіків. Мін має два потоки: вхідний та вихідний. На вхідний потік передають дані для їх відображення, а через вихідний потік передаються команди на повторний вибір даних для подальшої роботи з ними.

## 3.3 Розробка бази даних інформаційної системи «Генеалогічне дерево родини»

Для роботи з базою данних було обрано систему MySQL. Вільна реляційна система управління базами даних. Розробку і підтримку MySQL здійснює корпорація Oracle, що отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Переваги MySQL:

* найкраща швидкість обробки даних;
* безкоштовні відкриті ліцензії;
* простота використання;
* підтримка більшістю хостингових компаній;
* можливість використання на різних платформах (Unix, Windows);

### 3.3.1 Концептуальна модель даних

На першому кроці проектування бази даних, була використана концептуальна модель даних, що зображена на рис. 3.3. ЇЇ використання слугує для дослідження понять проблемної області бази даних інформаційної системи «Генеалогічне дерево родини».



Рисунок 3.3 – Концептуальна модель бази даних інформаційної системи «Генеалогічне дерево родини»

Як показано на рис. 3.3, база даних складається із наступних сутностей: «User»; «Friends»; «Tree»; «Member»; «Marriage»; «Parent»; «Child»; «HaveDiseases»; «GenDisease».

Розглянемо детальніше сутності і зв’язки з ними. Сутність «User» – це вся інформація про користувачів системи. «Friends» – друзі у системі, в цю сутність записуватимуться дані про користувачів, що додали один одного до друзів, причому друзі можуть надавати доступ до перегляду своїх генеалогічних дерев.

Кожен користувач обов’язково має своє дерево, за його зберігання відповідає сутність – «Tree». Для побудови генеалогічного дерева потрібна інформація про родичів користувача, для її зберігання використовується сутність – «Member». Сутності «Marriage», «Parent» та «Child» – відповідають за збереження родинних зв’язків таких, як: шлюб, батьківство, та наявність дітей. Щоб знайти ймовірності передачі генетичного захворювання від батьків до нащадків, необхідно зберігати інформацію про всі генетичні захворювання, а також про їх зв’язок з конкретним родичем, тому доцільно було створити сутності: «GenDiseases» та «HaveDiseases».

### 3.3.2 Логічна модель даних

На наступному кроці моделювання створено логічну модель бази даних інформаційної системи «Генеалогічне дерево родини». Вона дає змогу фіксувати вимоги до системи з точки зору розробника і описувати логічну організацію системи з базою даних, що реалізує ці вимоги. Така логічна модель допомагає прослідкувати перетворення зв’язків у відношеннях.



Рисунок 3.4 – Логічна модель бази даних інформаційної системи «Генеалогічне дерево родини»

Після перетворення концептуальної моделі у логічну з’являються нові ключі у сутностях, це показано на рис. 3.4. Таким чином, зв’язки між сутностями «Friends» та» User», створюють нові ключі у сутності «Friends» такі, як «id\_User», «use\_id\_User» це зовнішні ключі із таблиці «User».

Оскільки зв’язок між таблицями «User» та «Tree» – один-до-одного, то їх первинні ключі: «idUser», «idTree» стають зовнішніми і поміщаються до сутностей один одного.

Зв’язок між «Tree» та «Member» – один-до-багатьох, з цього слідує, що первинний ключ – «idTree» із сутності «Tree», стає зовнішнім ключем у сутності «Member».

Оскільки зв’язки між сутностями «Marriage», «Parent», «Child» та сутністю «Member» – це один-до-багатьох, то первинний ключ сутності «Member» стає зовнішнім ключем у сутностях «Marriage», «Parent» та «Child».

Щоб зв’язати сутності «Member» та «GenDiseases» зв’язком – багато-до-багатьох, їх первинні ключі необхідно представити, як зовнішні ключі у сутності «HaveDiseases».

### 3.3.3 Фізична модель даних

На третьому кроці моделювання було створено фізичну модель даних інформаційної системи «Генеалогічне дерево родини», що зображена на рис. 3.5. За її допомогою проектовано внутрішню схему бази даних, зображуючи таблиці даних, атрибути (стовпці) таблиць і відносини між таблицями, якщо потрібно, то і додаткові індекси.



Рисунок 3.5 – Фізична модель бази даних інформаційної системи «Генеалогічне дерево родини»

Детально розглянемо сутності фізичної моделі та зв’язки між ними.

Таблиця 3.1 – Специфікація атрибутів сутності «Friends»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| Id\_User | Ідентифікатор хто додає друга | 1-9999999 | - |
| Use\_IdUser | Ідентифікатор кого додають | 1-9999999 | - |
| CanSeeTree | Можливість перегляду дерева | True,False | - |

Сутність «Friends» є слабкою сутністю і містить у собі інформацію про дружні зв’язки та має два зовнішніх ключі «id\_User» та «use\_idUser» з сутності «User».

Таблиця 3.2 – Специфікація атрибутів сутності «User»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| IdUser | Ідентифікатор користувача | 1-9999999 | - |
| IdTree | Ідентифікатор дерева користувача | 1-9999999 | - |
| FirstName | І’мя користувача | Символи | - |
| SecondName | По-батькові користувача | Символи | - |
| LastName | Прізвище користувача | Символи | - |
| DateOfBirth | Дата народження користувача | Символи | Дата |
| Login | Поштовий адрес користувача | Символи | - |
| Password | Пароль користувача | Символи | - |
| Photo | Аватар користувача | Символи | Піксель |

Сутність «User» є сильною сутністю, вона зберігає повну інформацію про користувача. Первинним ключем є атрибут «IdUser», має зовнішній ключ «idTree» із сутності «Tree».

Таблиця 3.3 – Специфікація атрибутів сутності «Tree»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| IdTree | Ідентифікатор дерева | 1-9999999 | - |
| IdUser | Ідентифікатор користувача дерева | 1-9999999 | - |

Сутність «Tree» є сильною сутністю, вона зберігає інформацію про дерева певного користувача. Первинним ключем є атрибут «IdTree», має зовнішній ключ «IdUser» із сутності «User».

Таблиця 3.4 – Специфікація атрибутів сутності «Member»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| IdMember | Ідентифікатор родича | 1-9999999 | - |
| IdTree | Ідентифікатор дерева якому належить | 1-9999999 | - |
| FirstName | Їм’я родича | Символи | - |

Продовження таблиці 3.4

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| SecondName | По-батькові родича | Символи | - |
| LastName | Прізвище родича | Символи | - |
| Sex | Стать родича | Символи | - |
| DateOfBirth | День народження родича | Символи | Дата |
| DateOfDeth | День смерті родича(якщо помер) | Символи | Дата |
| Addres | Адреса проживання родича | Символи | - |
| Photo | Фото родича | Символи | Піксель |
| OtherInfo | Додаткова інформація про родича | Символи | - |

Сутність «Member» є сильною сутністю, вона зберігає повну інформацію про родичів у дереві користувача. Первинним ключем є атрибут «IdMember», має зовнішній ключ «idTree» із сутності «Tree».

Таблиця 3.5 – Специфікація атрибутів сутності «Marriage»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| idMember | Ідентифікатор родича(1) | 1-9999999 | - |
| mem\_idMember | Ідентифікатор родича(2) | 1-9999999 | - |
| Marriade | Чи знаходяться у шлюбі | True,False | - |

Сутність «Marriage» є слабкою сутністю, вона зберігає повну інформацію про шлюби родичів у дереві користувача. Має два зовнішні ключа «idMember» та «mem\_idMember» із сутності «Member».

Таблиця 3.6 – Специфікація атрибутів сутності «Parent»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| idMember | Ідентифікатор родича(1) | 1-9999999 | - |
| mem\_idMember | Ідентифікатор родича(2) | 1-9999999 | - |

Сутність «Parent» є слабкою сутністю, вона зберігає повну інформацію про батьків родичів у дереві користувача. Має два зовнішні ключа «idMember» та «mem\_idMember» із сутності «Member».

Таблиця 3.7 – Специфікація атрибутів сутності «Child»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| idMember | Ідентифікатор родича(1) | 1-9999999 | - |
| mem\_idMember | Ідентифікатор родича(2) | 1-9999999 | - |

Сутність «Child» є слабкою сутністю, вона зберігає повну інформацію про батьків родичів та їх дітей у дереві користувача. Має два зовнішні ключа «idMember» та «mem\_idMember» із сутності «Member».

Таблиця 3.8 – Специфікація атрибутів сутності «GenDiseases»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| IdDisease | Ідентифікатор хвороби | 1-9999999 | - |
| Name | Назва хвороби | Символи | - |
| WomenInherited | Чи передається жінкам | True,False | - |
| MenInherited | Чи передається чоловікам | True,False | - |
| About | Інформація про хворобу | Символи | - |

Сутність «GenDiseases» є сильною сутністю, вона зберігає повну інформацію про генетичні захворювання. Первинним ключем є атрибут «IdDiseases».

Таблиця 3.9 – Специфікація атрибутів сутності «HaveDiseases»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва атрибуту | Опис атрибуту | Діапазон значень | Одиниця вимірювання |
| IdMember | Ідентифікатор родича | 1-9999999 | - |
| IdDisease | Ідентифікатор хвороби | 1-9999999 | - |
| Dominant | Домінантна чи рецесивна | True,False | - |

Сутність «HaveDiseases» є слабкою сутністю, вона зберігає повну інформацію про родичів які хворіють на певні генетичні захворювання. Має два зовнішні ключа «idMember» та «IdDisease» із сутності «Member».

## 3.4 Розробка інтерфейсу користувача

Зовнішній інтерфейс розроблюваної інформаційної системи представляє собою набір екранних форм, що дозволятимуть користувачеві взаємодіяти з елементами керування цією системою. Для того, щоб користувачу було зручно використовувати систему, було розроблено простий, інтуїтивний інтерфейс.

Схема користувацького інтерфейсу представлена на рис. 3.6.



Рисунок 3.6 – Схема користувацького інтерфейсу інформаційної системи «Генеалогічне дерево родини»

Розглянемо деякі форми та візуальні компоненти, з яких складається додаток.

Для аутентифікації користувача у системі, було розроблено форму авторизації. Загальний вигляд форми авторизації зображено на рис. 3.7.

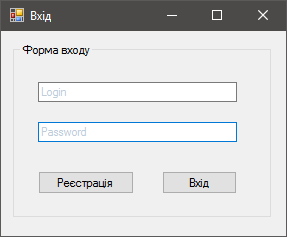


Рисунок 3.7 – Форма авторизації користувача

Форма авторизації, що зображена на рис.3.7 складається із таких компонентів:

* Поле для вводу логіну(Поштова скринька користувача);
* Поле для вводу пароля;
* Кнопка входу в програму;
* Кнопка для виклику форми реєстрації користувача.

Якщо користувач ввів коректні дані логіна та пароля та натиснув кнопку «Вхід», він перейде до головного вікна програми. Якщо ж користувач не зареєстрований, йому потрібно натиснути кнопку «Реєстрація» і перейти до форми реєстрації. На рисунку 3.8 зображено форму реєстрації нових користувачів.

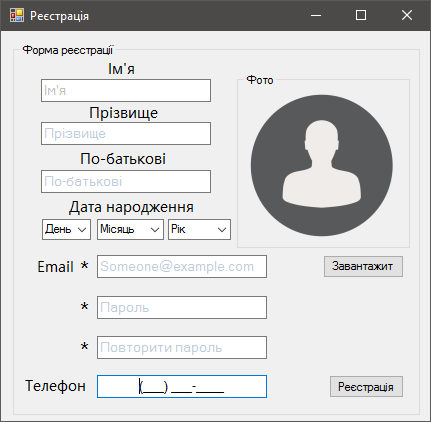


Рисунок 3.8 – Форма реєстрації користувача

Після успішної реєстрації та авторизації користувач потрапить до головного вікна програми, що зображено на рис. 3.9.

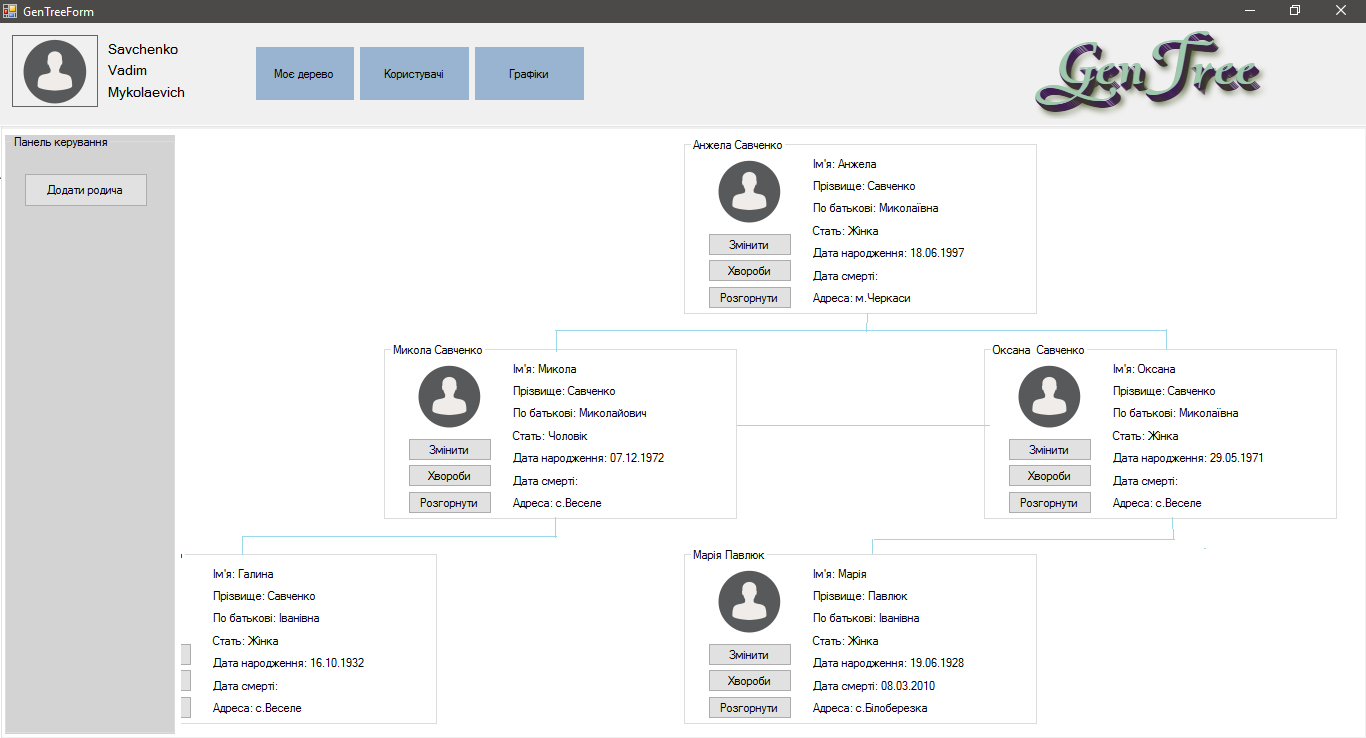


Рисунок 3.9 – Головне вікно програми

На рис.3.9 зображено генеалогічне дерево користувача. Також є кнопки для перегляду та зміни детальної інформації про кожного родича(його хвороби, сімейні зв’язки з іншими родичами). Після натискання на кнопку «Додати родича», користувачу відкриється вікно додавання родича. Якщо користувач натисне кнопку «Користувачі» або «Графіки», то він відповідно перейде до блоку зображення інформації про інших користувачів(друзів, підписників, користувачів у системі) або зображення графіків та статистичної інформації про його дерево.

## 3.5 Опис розробки програмних компонентів

Для зручного відображення елементів керування програмою, було створено велику кількість модулів та компонентів, розглянемо деякі з них.

Щоб правильно побудувати генеалогічне дерево був використаний рекурсивний обхід по родичам поточного користувача. Нижче приведені функції, які реалізують рекурсивний обхід.

public void GenerateTree(Panel panel,string urlApi)

{

List<AllMembersViewModel> members = JsonConvert.DeserializeObject<List<AllMembersViewModel>>(

RequestController.HttpGet(urlApi, MainForm.Token).Result);

List<AllMembersViewModel> childs=members.Where(x =>x.Childs.Count== 0) .ToList();

foreach (var child in childs)

{

RequrseGenerate(child,members,panel,500,3);

}

}

public void RequrseGenerate(AllMembersViewModel child, List<AllMembersViewModel> members, Panel panel ,int posx, int level)

{

MemberPanel memberPanel = new MemberPanel(child)

{

Location = new Point(posx,level)

};

panel.Controls.Add(memberPanel);

if(child.Parents.Count>0)

RequrseGenerate(members.FirstOrDefault(x=>x.Id == child.Parents[0].ParentId),members,panel ,posx-300,level+205);

if (child.Parents.Count > 1)

RequrseGenerate(members.FirstOrDefault(x => x.Id == child.Parents[1].ParentId), members,panel,posx+300,level +205);

}

Модуль відображення родичів у генеалогічному дереві містить у собі компонент «MemberPanel», який зображений на рис. 3.10.

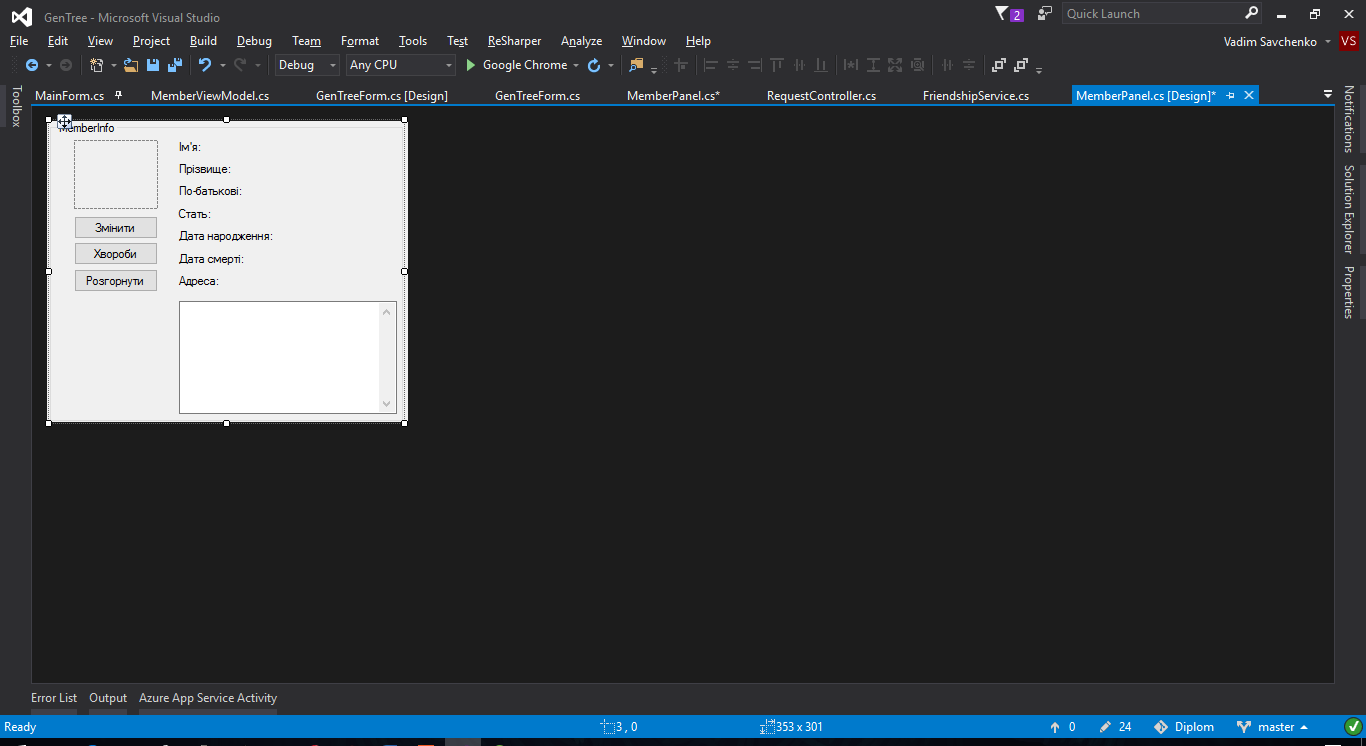


Рисунок 3.10 – Компонент «MemberPanel»

Компонент «MemberPanel», що зображений на рис.3.10, створено для зручного відображення повної інформації про родичів у поточному генеалогічному дереві. Даний компонент складається із запозичених компонентів WindowsForm таких, як: «Label», «PictureBox», «Button», та «TextBox».

Окрім розглянутих компонентів розроблюваної системи «Генеалогічне дерево родини», існують також інші (наприклад, компонент для відображення друзів або підписників, компонент для виведення інформації про генетичні захворювання), але вони мають досить тривіальну складову, тому їх не розглядатимемо.

## 3.6 Тестування інформаційної системи «Генеалогічне дерево родини»

У залежності від переслідуваних цілей види [тестування](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#Тестування) можна умовно розділити на наступні типи:

* функціональні;
* нефункціональні;
* пов’язані зі змінами.

Функціональні тести базуються на функціях та особливостях, а також на взаємодії з іншими системами, і можуть бути представлені на всіх рівнях тестування: компонентному або модульному (Component/Unit testing), інтеграційному (Integration testing), системному (System testing) і приймальному (Acceptance testing). Функціональні види тестування розглядають зовнішню поведінку системи. Одні з найпоширеніших видів функціональних тестів:

* функціональне тестування (Functional testing);
* [тестування](http://lib.mdpu.org.ua/e-book/vstup/glos.htm#Тестування) безпеки (Security and Access Control Testing) - стратегія тестування, що використовується для перевірки безпеки системи, а також для аналізу ризиків, пов’язаних із забезпеченням цілісного підходу до захисту програми, атак хакерів, вірусів, несанкціонованого доступу до конфіденційних даних. Тестування безпеки може виконуватися як автоматизовано так і в ручну, включаючи перевірку як позитивних, так і негативних тестових випадків;
* тестування взаємодії (Interoperability Testing) – це функціональне тестування, що перевіряє здатність програми взаємодіяти з одним і більше компонентами або системами і включає в себе тестування сумісності (compatibility testing) та інтеграційне тестування (integration testing).[19].

Тестування модулів інформаційної системи «Генеалогічне дерево родини» подано в таблиці 3.10.

Таблиця 3.10 – Тестування модулів інформаційної системи «Генеалогічне дерево родини»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва функції | Ознака тестування | Вид тестування | Результат тестування |
|  | 1. Модуль ViewControl | | | |
| 1.1 | Запит на завантаження даних | Перевірка завантаження даних з сервера | Чорний ящик | Дані були завантажені. Це означає, що запит до модулю, що керує серверними запитами, був вдалий. |
| 1.2 | Вибір даних для відображення | Перевірка правильності відображення елементів. | Білий ящик | Дані перевіряються успішно. При неправильних даних виводиться вікно про помилку. |
| 1.3 | Побудова генеалогічного дерева | Перевірка правильного розташування компонентів відображення родичів. | Чорний ящик | При побудові дерева всі дані коректні. Відображення дерева пройшло успішно. |
| 1.4 | Виклик методів з меню | Перевірка виклику методів завантаження даних, їх аналізу та інших із функціонального меню. | Білий ящик | Всі необхідні методи викликаються правильно. |
|  | 1. Модуль ServerConnector | | | |
| 2.1 | Перевірка парсингу | Перевірка коректності парсингу даних з JSON | Білий ящик | Перевірка проводиться коректно. Всі об’єкти серіалізуються вірно. |
|  | 1. Модуль Graphics | | | |
| 3.1 | Створення графіків | Перевірка коректності складання графіків та формату даних на сторінці. | Білий ящик | Відкритий графік містить всі необхідні дані, які відображаються коректно та мають правильний формат. |
|  | 1. Модуль Request Controller | | | |
| 4.1 | Отримання даних з серверу | Перевірка отримання даних з віддаленого серверів. | Білий ящик | Запит відправляється вдало та у відповідь з сервера приймаються необхідні дані показників у правильному форматі |
|  | 5. Модуль DataManagment | | | |
| 5.1 | Зміна даних | Перевірка коректності структури даних та можливості зміни та додавання елементів структури. | Білий ящик | Редагування структури відбувається коректно. Структура має робочий вигляд та добре пристосована для обробки програмою. |

Продовження таблиці 3.10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва функції | Ознака тестування | Вид тестування | Результат тестування |
|  | 6. Модуль DataBase | | | |
| 6.1 | Запит на додавання даних | Перевірка можливості додавання нових даних до бази | Білий ящик | Дані були додані. Відповідні класи бачать нові поля та можуть з ними коректно працювати. |

В результаті тестування інформаційної системи «Генеалогічне дерево родини» не було визначено жодних помилок. Було протестовано такі модулі: ViewControl, ServerConector, Graphics, Request Controller, Data Management, DataBase та клієнтський інтерфейс. Тестування проводилось методом білого та чорного ящика. Результатом проходження тестування можна вважати придатність системи до впровадження.

## Висновки до третього розділу

Було створено функціональний опис системи «Генеалогічне дерево родини», а також сформовано ідею реалізації розробки даної системи, на основі функціональної схеми .

Проведено огляд методів розробки системи «Генеалогічне дерево родини» – обрано засоби розробки, мову програмування, а також засоби розробки серверу для отримання даних та бази даних для зберігання інформації. Було обрано формат даних для обміну між клієнтом та сервером.

Також було розроблено базу даних на основі концептуальної, логічної та фізичної моделях даних.

Головною частиною розробки системи є написання програмного коду та розробка інтерфейсу користувача, що було описано в даному розділі.

На етапі розробки і тестування інформаційна система набуває досконалого вигляду та, саме після вдалого тестування, готова до впровадження за призначенням розробки. Попередні етапи розробки системи є основою у створенні програмного продукту, але лише розробка системи дозволяє реалізувати аналізовану інформацію, вимоги до проекту та результати його проектування.

# ВИСНОВКИ

У ході виконання дипломної роботи спроектовано та розроблено інформаційну систему «Генеалогічне дерево родини», яка дозволяє зберігати інформацію про своїх пращурів у вигляді генеалогічного дерева, а також створює графіки за результатами аналізу даних та знаходження ймовірності успадкування генетичних хвороб – нащадками.

На підставі здобутих теоретичних знань, розроблений комплексні методи побудови генеалогічного дерева та знаходження закономірностей в успадкуванні генетичних захворювань за законами Менделя, основа яких полягла в створення системи «Генеалогічне дерево родини».

У даній роботі були повністю розв’язані задачі, поставлені в завданні на виконання дипломної роботи, а саме:

* дослідження поняття генеалогічного дерева родини та його основних методів побудови;
* дослідження структури існуючих інформаційних систем побудови генеалогічного дерева родини, а також визначення їх головних переваг та недоліків;
* розробка теоретичної частини інформаційної системи «Генеалогічне дерево родини»;
* розробка практичної частини інформаційної системи «Генеалогічне дерево родини»;
* розробка простого інтерфейсу, який доступний для розуміння та використання недосвідченим користувачам.
* розробка програмного продукту, що відповідає вимогам поставленої задачі на розробку та дотримується ідеології актуальності дипломної роботи.

Розроблений алгоритм має практичну актуальність і може бути застосований у системах побудови генеалогічного дерева або знаходження ймовірності прояву генетичних захворювань у нащадків.

За допомогою UML діаграм спроектовано інформаційну систему «Генеалогічне дерево родини». Також було розроблено програмний інтерфейс, що дозволяє керувати всіма необхідними функціональними засобами системи.

Розроблена система була написана на мові програмування С# в середовищі розробки Visual Studio 2017. Сервер для отримання та зберігання даних був розроблений з використанням засобів ASP.NET, а самі дані передавались у форматі відображення даних JSON.

Тестування системи проводилося методами чорного та білого ящика. Тестування розробленої системи показало, що вона володіє хорошими показниками якості і швидкості передачі та зберігання даних на сервері. Всі тести були пройдені успішно і тому розроблена система повністю готова до впровадження.

Подальший розвиток системи може бути пов’язаний з її розширенням, що дозволить використовувати також інші генеалогічні дерева із інших інформаційних систем, направлені на отримання дещо інакшого результату, відповідного до потреб нових методів. Окрім цього є можливим покращення системи в плані управління та додаткових засобів керування системою а також впровадження у систему соціальних мереж.

Основною перевагою розробленої системи є те, що її використання є універсальним, а саме ­– вона може працювати з будь-яким середовищем. При цьому на те, щоб пристосувати дану систему до інших платформ, необхідно використати спеціально написане API. Відносно важливості задачі, це є досить швидким результатом пристосування програми до інших платформ. Отже, розроблена програма є універсальною і може бути застосована на будь якому пристрої.

Встановлення програми займає мало часу та може бути виконана навіть не досвідченим користувачем.

Програма адекватно функціонує, швидко реагує на усі запити користувача і не містить помилок, які б могли руйнувати чи порушувати роботу програми, здійснювати системні порушення, які несуть фатальні наслідки для комп’ютерної системи в цілому.

Програма відповідає всім поставленим вимогам, має доступний і простий зовнішній інтерфейс, дає можливість продовжити розробку додаткових функцій, необхідних для удосконалення її роботи, адже має модульну структуру, яка легко розширюється.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Савелов Л. М. Лекции по русской генеалогии. — М.: Археографический центр. 1994. — Репринт издания 1909 года. [ISBN 5-86169-015-4](http://www.wikiwand.com/uk/%D0%A1%D0%BF%D0%B5%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0:%D0%94%D0%B6%D0%B5%D1%80%D0%B5%D0%BB%D0%B0_%D0%BA%D0%BD%D0%B8%D0%B3/5861690154) (Т. 3); [ISBN 5-86169-015-4](http://www.wikiwand.com/uk/%D0%A1%D0%BF%D0%B5%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0:%D0%94%D0%B6%D0%B5%D1%80%D0%B5%D0%BB%D0%B0_%D0%BA%D0%BD%D0%B8%D0%B3/5861690154) (рос.)
2. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.
3. Філіпцова О. В. Спадкові хвороби.[Електронний ресурс] / О. В. Філіпцова –Режим доступу [http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi (дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
4. Генеалогія родини.[Електронний ресурс] – Режим доступу: <http://rodovody.com.ua/index.php/ua/genealogy> [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
5. Сиволоб А.В., Рушковський С.Р., Кир'яченко С.С. та ін. Генетика: підручник. — Київ : ВПЦ "Київський університет", 2008. — С. 92—93.
6. Family Tree Builder.[Електронний ресурс]. – Режим доступу: <https://www.myheritage.com/family-tree-builder?lang=RU>[(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
7. SimTree – Программа для составления генеалогического дерева [Електронний ресурс]. – Режим доступу: <http://simtree.ru/> [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
8. GenoPro – Программа для составления генеалогического дерева [Електронний ресурс]. – Режим доступу: <https://www.genopro.com/ru/>  [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
9. RootsMagic Essentials [Електронний ресурс]. – Режим доступу: <https://www.rootsmagic.com/Essentials/> [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
10. Методичні рекомендації щодо виконання дипломних робіт для студентів денної та заочної форм навчання напряму підготовки 6.050101 – «Комп’ютерні науки». / Авраменко В.С., Голуб С.В., Салапатов В.І. – Видання друге, перероблене і доповнене. – Черкаси: ЧНУ, 2016. – 108 с.
11. Литвинов В.В., Голуб С.В., Григор’єв К.М., Жигульська В.Ю. Об’єктно-орієнтоване моделювання при проектуванні вбудованих систем і систем реального часу. [Навчальний посібник з дисципліни: «Системний аналіз та проектування комп’ютерних інформаційних систем»]. – Черкаси: Вид. Від ЧНУ ім. Б. Хмельницького, 2011. – 376 с.
12. Диаграмма прецедентов. [Електронний ресурс]. – Режим доступу: [http://ru.wikipedia.org/wiki/ Диаграмма\_прецедентов](http://ru.wikipedia.org/wiki/%20Диаграмма_прецедентов) [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
13. Авраменко В.С., Голуб С.В., Салапатов В.І. Проектування інформаційних систем. [Навчальний посібник] – Черкаси: Вид. Від ЧНУ ім. Б. Хмельницького, 2015. – 501 с.
14. Модель-вид-контролер – Вікіпедія [Електронний документ]. – Режим доступу: https://uk.wikipedia.org/wiki/ Модель-вид-контролер  [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
15. C Sharp – Вікіпедія [Електронний документ]. – Режим доступу: <https://uk.wikipedia.org/wiki/C_Sharp> [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
16. Microsoft Visual Studio – Вікіпедія [Електронний документ]. – Режим доступу: <https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio> [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
17. ASP.NET – Вікіпедія [Електронний документ]. – Режим доступу: <https://uk.wikipedia.org/wiki/ASP.NET> [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
18. JSON – Вікіпедія [Електронний документ]. – Режим доступу: https://uk.wikipedia.org/wiki/ JSON [(дата](http://www.pharmencyclopedia.com.ua/article/591/spadkovi-xvorobi%20(дата) звернення:19.06.2017).
19. Канер Кем, Фолк Джек, Нгуен Енг Кек Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев : ДиаСофт, 2001. — 544 с.

**ДОДАДКИ**

# ДОДАТОК А

**Текст програми**

(Текст програми знаходиться на оптичному диску)