МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
ІМ. М. Є. ЖУКОВСЬКОГО
"ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ"

Кафедра комп'ютерних наук та інформаційних технологій

КУРСОВА РОБОТА

з дисципліни «Крос-платформне програмування»

на тему Розробка крос-платформного застосунку-помічника куратора академічної групи

Виконав: здобувач освіти
групи 316ст
спеціальності «122. Комп'ютерні науки»
(освітня програма – «Комп'ютеризація обробки інформації та управління»)

Бабич Ангеліна Олександрівна
(прізвище, ім'я та по батькові)

Керівник А.В. Попов
(підпис) (прізвище та ініціали)

Харків – 2023

ЗМІСТ

ВСТУП

Актуальність роботи: Протягом останніх років стрімко розвиваються ІТ-технології, і інформаційні системи займають у цих технологіях центральне положення.

Інформаційна система – це взаємопов'язана сукупність засобів, методів та персоналу, що використовуються для зберігання, обробки та видачі інформації на користь досягнення поставленої мети.

Інформаційні системи забезпечують збирання, зберігання, обробку, пошук, видачу інформації, необхідної в процесі прийняття рішень задач з будь-якої галузі. Вони допомагають аналізувати проблеми та створювати нові продукти.

На цей час у багатьох організаціях накопичено значні обсяги даних, на основі яких є можливість розв'язання різноманітних аналітичних і управлінських завдань. Проблеми зберігання та обробки аналітичної інформації стають все більш актуальними та привертають увагу фахівців та фірм, що працюють у галузі інформаційних технологій, що призвело до формування повноцінного ринку технологій бізнес-аналізу.

В ідеалі робота аналітиків і керівників різних рівнів повинна бути організована так, щоб вони могли мати доступ до всієї інформації, що їх цікавить, і користуватися зручними і простими засобами подання та роботи з цією інформацією. Саме на досягнення цих цілей і спрямовані інформаційні технології, що поєднуються під загальною назвою сховищ даних та бізнес-аналізу.

Одним із прикладів таких організацій є будь-який навчальний заклад, де зберігання інформації про здобувачів освіти, ведення обліку їх досягнень, та формування характеристик для пред’явлення до військкомату чи за місцем вимоги, є невід’ємним процесом.

Тема розробки такої інформаційної системи для куратора академічної групи є *актуальною і своєчасною*, оскільки очевидною є необхідність урахування даних про студентів, що постійно доповнюються, і наявність

оперативного доступу до них. У наш час існують різні види обліку студентів, які ведуться кураторами, викладачами, завідувачами відділень. Всі дані про студентів зберігаються у журналах груп, відомостях, списках тощо.

Саме для того щоб спростити та прискорити процес роботи використовуються інформаційні системи.

Метою роботи є створення інформаційної системи для куратора (керівника) групи, яка б автоматизувала процес обліку інформації про студентів, надавала оперативний доступ до даних, автоматизувала процес створення характеристик студентів, та облік їх навчальних та позааудиторних досягнень.

Для того щоб досягти мети, необхідно виконати ряд *завдань*:

— провести опис предметної області;
— розглянути та порівняти подібні системи;
— розробити базу даних інформаційної системи;
— провести характеристику програмних засобів;
— розробити автоматизовану інформаційну систему для куратора групи;
— описати програмний продукт, його структуру та функціональні можливості;

Об'єктом дослідження в роботі є процес проектування, розробки та розгортання автоматизованих інформаційних систем.

Предметом дослідження є особливості використання платформи Windows Presentation Foundation (WPF) та патерну MVVM для створення автоматизованої інформаційної системи для куратора навчальної групи.

У процесі виконання курсової роботи були використані наступні *методи дослідження*:

— Теоретичні – порівняльний аналіз підходів до створення АІС з метою детального аналізу їх структури.
— Групування – для визначення залежності одних показників від інших, утворення однорідних груп на основі розподілу сукупності на окремі

частини або об'єднання досліджуваних одиниць у часткові сукупності за суттєвими для них ознаками.

— Системний аналіз – для деталізації і розчленування на окремі важливі складові частини, що надає можливості з аналізу, прогнозування, проектування прийняття рішень в складних системах різної природи на основі системної методології.

— Аналіз і синтез – для визначення особливостей процесу візуального оформлення програмного продукту.

— Абстрагування – для формулювання узагальнених висновків на основі системного аналізу та синтезу теорії і практики.

— Порівняння – для зіставлення даних у динаміці, що використовуються з метою встановлення логічних закономірностей, які впливають на досліджувані об'єкти або явища, і пошуку переваг та вразливостей, які можуть виявлятися під впливом факторів.

Обсяг та структура роботи. Курсова робота складається з вступу, 5 розділів, висновків, списку використаної літератури та додатків. Вона викладена на 73 сторінках, ілюстрована 9 таблицями та 15 рисунками.

1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ. РОЗГЛЯД ПОДІБНИХ СИСТЕМ ТА ЇХ ПОРІВНЯННЯ

1.1. Поняття, структура та класифікація автоматизованих інформаційних систем

Автоматизовані інформаційні системи (АІС) – це системи для пошуку, збирання, зберігання, накопичення, обробки, передачі інформації за допомогою використання обчислювальної техніки, засобів і каналів зв'язку, комп'ютерних інформаційних мереж.

Основною метою застосування більшості інформаційних автоматизованих систем є використання технологій, які здатні виконувати завдання, які не може обробити людський мозок: обробку великої кількості інформації, виконання складних обчислень і контроль багатьох одночасних процесів [1].

СТРУКТУРА АІС

Структура АІС – внутрішня організація системи при поділі її на частини, виявлення зв'язків між цими частинами.

АІС складається з двох базових підсистем: функціональної та забезпечувальної.

Функціональна частина АІС включає низку підсистем, що охоплюють вирішення конкретних завдань планування, контролю, обліку, аналізу та регулювання діяльності керованих об'єктів. У результаті аналітичного обстеження може бути виділено різні підсистеми, набір яких залежить від виду підприємства, його специфіки, рівня управління та інших чинників. Для нормальної діяльності функціональної частини АІС до її складу входять підсистеми забезпечувальної частини АІС (так звані забезпечувальні підсистеми).

Загальну структуру інформаційної системи можна розглядати як сукупність підсистем незалежно від сфери застосування. У цьому випадку говорять про структурну ознаку класифікації, а підсистеми називають

забезпечувальними. Таким чином, структура будь-якої інформаційної системи може бути представлена сукупністю забезпечувальних підсистем. Серед забезпечувальних підсистем, зазвичай виділяють інформаційне, технічне, математичне, програмне, організаційне та правове забезпечення**.**

Інформаційне забезпечення – сукупність єдиної системи класифікації та кодування інформації, уніфікованих систем документації, схем інформаційних потоків, що циркулюють в організації, а також методологія побудови баз даних.

Технічне забезпечення – комплекс технічних засобів, призначених для роботи інформаційної системи, а також відповідна документація на ці засоби та технологічні процеси.

Математичне та програмне забезпечення – сукупність математичних методів, моделей, алгоритмів та програм для реалізації цілей та завдань інформаційної системи, а також нормального функціонування комплексу технічних засобів.

Організаційне забезпечення – сукупність методів і засобів, що регламентують взаємодію працівників з технічними засобами та між собою у процесі розробки та експлуатації інформаційної системи.

Правове забезпечення – це сукупність заходів щодо створення нормативно-правової бази для діяльності конкретного підприємства.

КЛАСИФІКАЦІЯ АІС

За призначенням функціонуючої інформації ІС поділяються на: державні, юридичні (законодавчі), ділові, фінансові, науково-технічні, навчальні, соціальні, розважальні та інші.

За галузями застосування виділяють: ділову, професійну, споживчу інформацію та електронну комерцію.

За рівнем управління виділяють: стратегічні, тактичні та оперативні інформаційні системи.

За рівнем застосування технічних засобів ІС ділять на автоматизовані та неавтоматизовані. При цьому автоматизовані мають на увазі автоматизацію

від окремих процесів та завдань до рівня автоматизації підприємств, установ та їх сукупності в масштабах території.

За типами інформації: документальні, фактографічні та документально-фактографічні ІС.

Виділяють чотири типи АІС:

— що охоплює один процес (операцію) в одній організації;
— що об'єднує кілька процесів в одній організації;
— забезпечує функціонування одного процесу в масштабі кількох взаємодіючих організацій;
— реалізує роботу кількох процесів чи систем у масштабі кількох організацій [2].

1.2. Процес розробки автоматизованих інформаційних систем

Процес створення АІС являє собою сукупність упорядкованих у часі, взаємозв’язаних і об’єднаних у стадії та етапи робіт, виконання яких необхідне і достатнє для створення системи, що відповідає заданим вимогам.

Стадії та етапи розробки АІС:

— Стадія формування вимог до АІС. Етапи: обстеження об’єкта і обґрунтування необхідності побудови системи; формування вимог користувача до неї; оформлення звіту і заявки на її розробку (тактико-технічне завдання).
— Стадія розробки концепції АІС. Етапи: вивчення об’єкта; виконання необхідних науково-дослідних робіт (НДР); розробка варіантів концепції АІС і вибір того з них, який задовольняє вимоги користувача; оформлення звіту про виконану роботу.
— Стадія розробки технічного завдання. Етапи: розробка технічного завдання та його затвердження.
— Стадія ескізного проектування. Етапи: розробка попередніх проектних вирішень стосовно системи та окремих її частин.

— Стадія технічного проектування. Етапи: розробка проектних вирішень стосовно системи та її частин; розробка документації АІС та її частин; розробка й оформлення документації на поставляння або розробку виробів для комплектування системи; розробка завдань на проектування в суміжних частинах проекту автоматизації.
— Стадія робочого проектування. Етапи: розробка робочої документації на систему та її частини; створення або адаптація програм.
— Стадія впровадження системи в дію. Етапи: підготовка об’єкта автоматизації до впровадження АІС; підготовка персоналу; комплектування АІС (програмними і технічними засобами, інформаційними виробами); будівельно-монтажні роботи; пусконалагоджувальні роботи; попередні випробування; дослідна експлуатація; приймальні випробування.
— Стадія впровадження. Етапи: виконання робіт згідно з гарантійними зобов’язаннями та післягарантійне обслуговування [3].

1.3. Огляд систем подібного призначення та їх порівняння

У мережі Інтернет можна легко знайти найрізноманітніші системи обліку, які вирішують завдання обрахунку вартості продукції, контролю абонементів у спортивному клубі тощо. Якщо ж розглядати системи обліку з обраної предметної області – то вибір інформаційних систем дещо зменшується.

Аналіз ринку аналогів програмних продуктів заданої предметної області необхідний, щоб визначити плюси та мінуси існуючих проектів та зуміти створити унікальний продукт.

Для порівняння будемо розглядати наступні інформаційні системи: «Універсальна система обліку», «Кадри Плюс Україна», «Управління життєвим циклом студентів».

УНІВЕРСАЛЬНА СИСТЕМА ОБЛІКУ

«Універсальна Система Обліку» – це функціональна інформаційна система, яка зберігає велику кількість персональних даних клієнтів, постачальників та інших організацій, а також відомості про компанію з детальним описом її функціонування та характеристик (Рисунок 1.1).

Універсальна Система Обліку підвищує продуктивність кадрів та ефективність бізнес-процесів, скорочує витрати праці на ведення облікової та звітної діяльності підприємства в цілому та часові витрати при роботі з клієнтами зокрема.

Універсальна Система Обліку може використовуватись в будь-якій сфері діяльності і на будь-якому підприємстві, інсталюється на будь-який електронний пристрій (планшет, ноутбук, комп'ютер) [4].

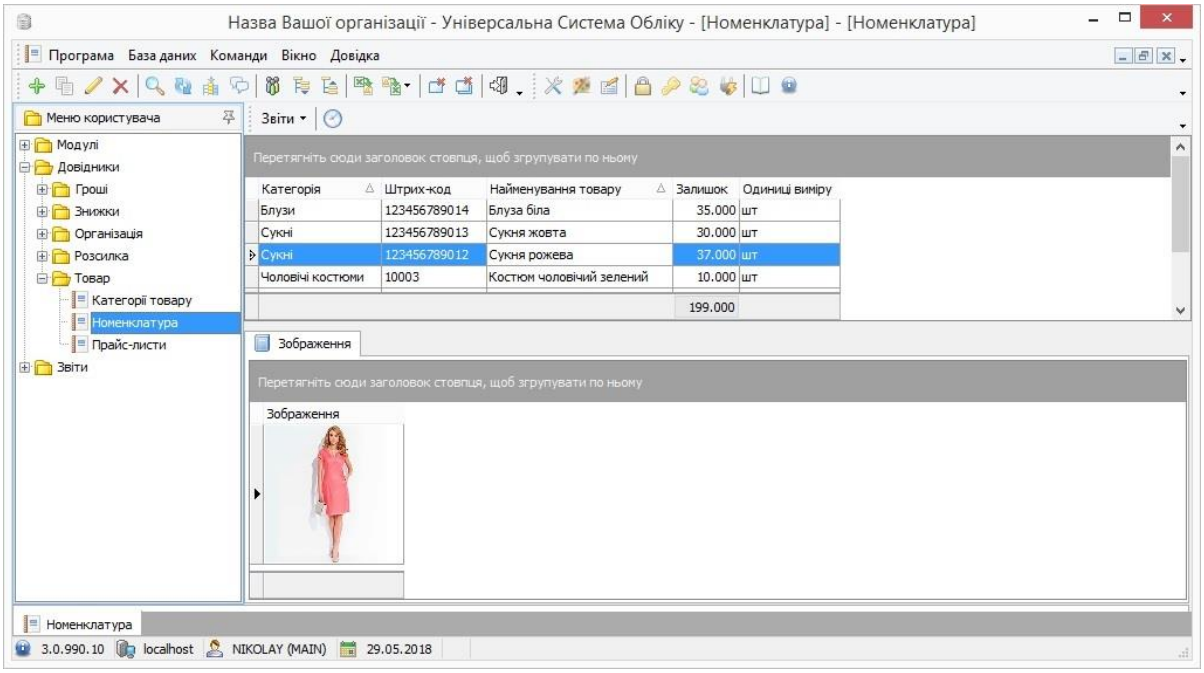


Рисунок 1.1 – Інтерфейс програми «Універсальна система обліку»

ПЕРЕВАГИ СИСТЕМИ

— Універсальність. Систему можна використовувати у будь-якій сфері життєдіяльності та застосовуватися на підприємствах різного напрямку.
— Не вимагає від користувача особливих знань інформаційних технологій та вмінь.

— Має інтуїтивно-зрозумілий інтерфейс користувача.

— Присутній багатокористувацький режим.

— Присутня можливість експорту/імпорту даних.

— Досить легка у інсталяції.

НЕДОЛІКИ

— Відсутність крос-платформності.

— Висока вартість.

— Сумнівна надійність розробника.

— Відсутність автентифікації з використанням соціальних облікових записів.

КАДРИ ПЛЮС УКРАЇНА

Кадри Плюс Україна – це автоматизована інформаційна система для обліку персоналу та робочого часу, розроблена компанією AnDeeSoft. Вона працює на персональних комп’ютерах під управлінням ОС Windows (Рисунок 1.2).

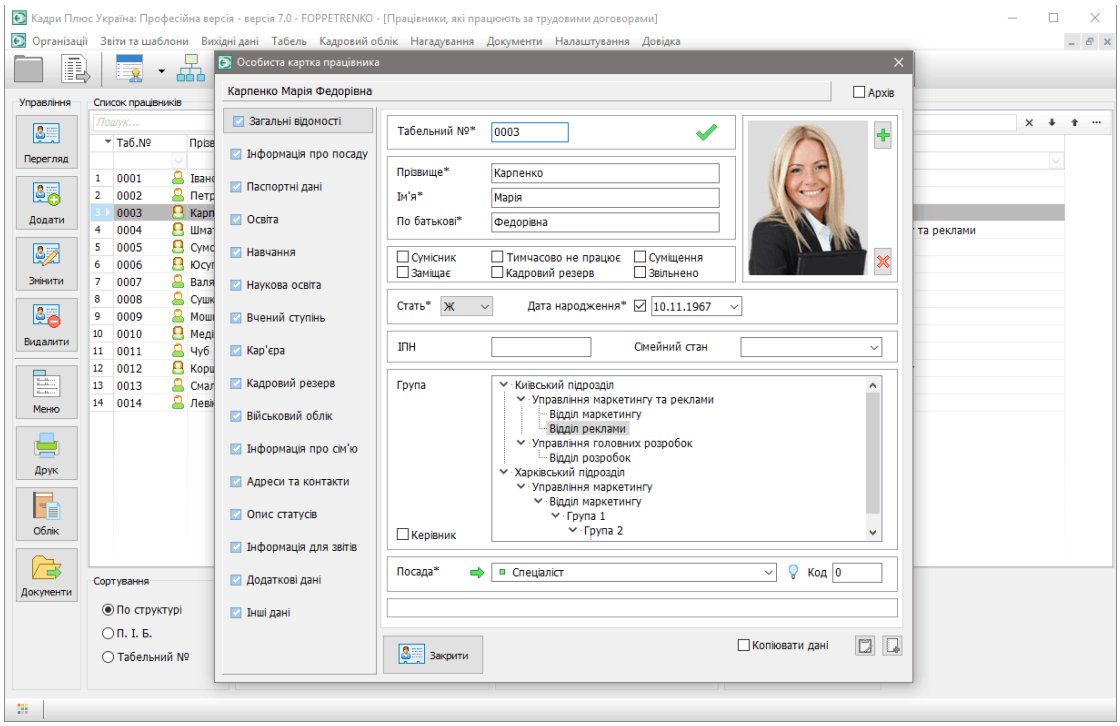


Рисунок 1.2 – Інтерфейс програми «Кадри Плюс Україна»

Програма допомагає з легкістю створювати накази, заяви, звіти, інші

типові документи, відслідковувати рух персоналу, вести журнали документів та облік робочого часу. Програма також має інші додаткові функції, які будуть корисні кадровому працівникові [5].

Приклад роботи програми наведено на рисунку 1.3.

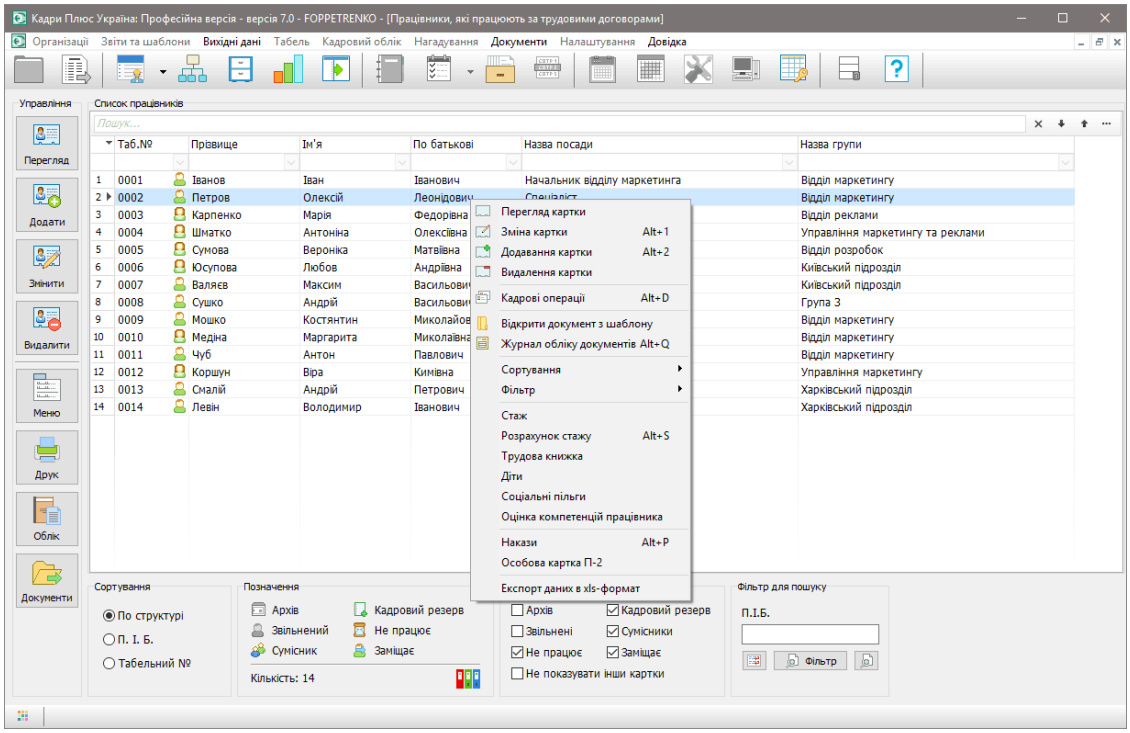


Рисунок 1.3 – Приклад роботи програми «Кадри Плюс Україна»

Функціональні можливості даної системи:

— Можливість ведення будь-якої кількості організацій в одній програмі.
— Ведення бази даних працівників, які працюють за трудовими договорами.
— Структурована система організації.
— Класифікатор посад.
— Облік робочого часу працівників.
— Загальний та особисті журнали документів.
— Облік руху працівників та ведення кадрової статистики.
— Розрахунок всіх видів стажу.
— Книга обліку руху трудових книжок і вкладишів до них, а також ведення трудових книжок.
— Ведення обліку додаткових відомостей про працівника.

— Можливість роботи програми в режимі клієнт-сервер.

ПЕРЕВАГИ СИСТЕМИ

— Має велику кількість функціональних можливостей.

— Має інтуїтивно-зрозумілий інтерфейс користувача.

— Присутній багатокористувацький режим.

— Наявність української мови.

— Присутня можливість експорту/імпорту даних.

НЕДОЛІКИ

— Відсутність крос-платформності.

— Відсутність автентифікації з використанням соціальних облікових записів.

— Висока вартість ліцензій

УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ СТУДЕНТІВ

Останньою з трьох інформаційних систем для аналізу розглянемо систему «Управління життєвим циклом студентів» [6].

Інтерфейс системи можна розглянути на рисунку 1.4.

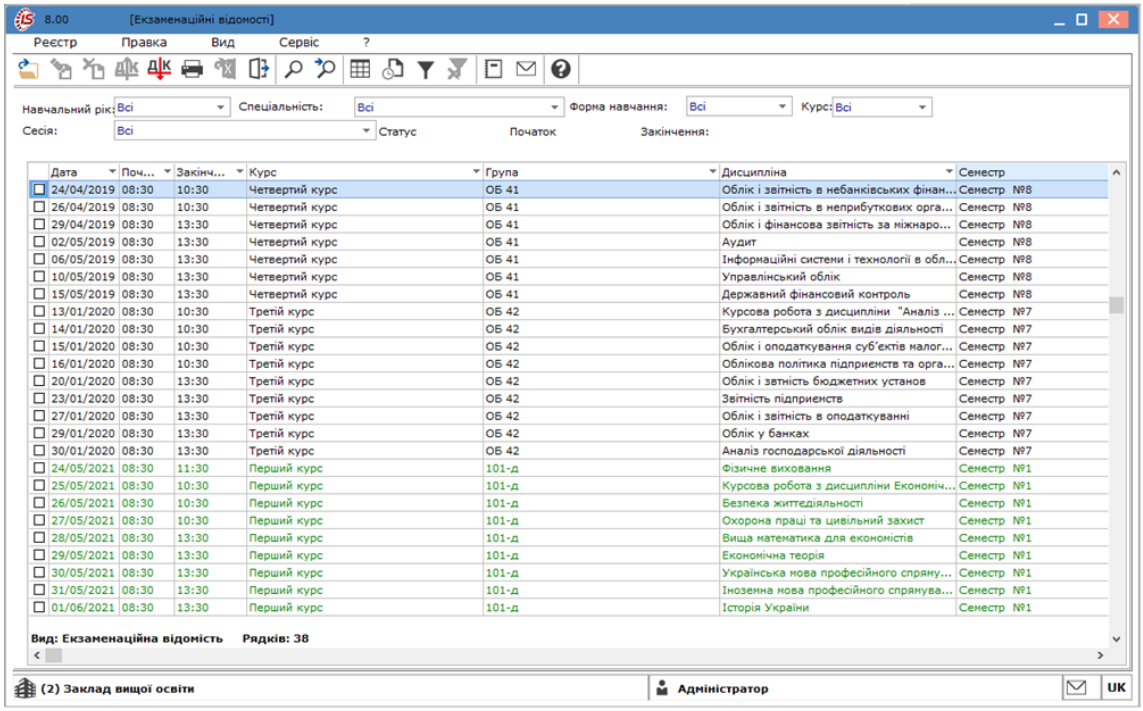


Рисунок 1.4 – Інтерфейс системи «Управління життєвим циклом студентів»

Функціональні можливості системи «Управління життєвим циклом студентів»:

— Вирішення завдання обліку і управління даними про студентів. Кожному студенту присвоюється «Картка студента» (Рисунок 1.5).
— Ведення документообігу по студенту від вступу до закінчення навчального закладу (в тому числі відрахування, відновлення, академічні відпустки).
— Ведення особової справи студентів, формування наказів по студентах (в тому числі про переведення на наступний курс / навчальний період).
— Облік договорів зі студентами (контракт, оплата гуртожитку).
— Призначення стипендії (соціальна, академічна).

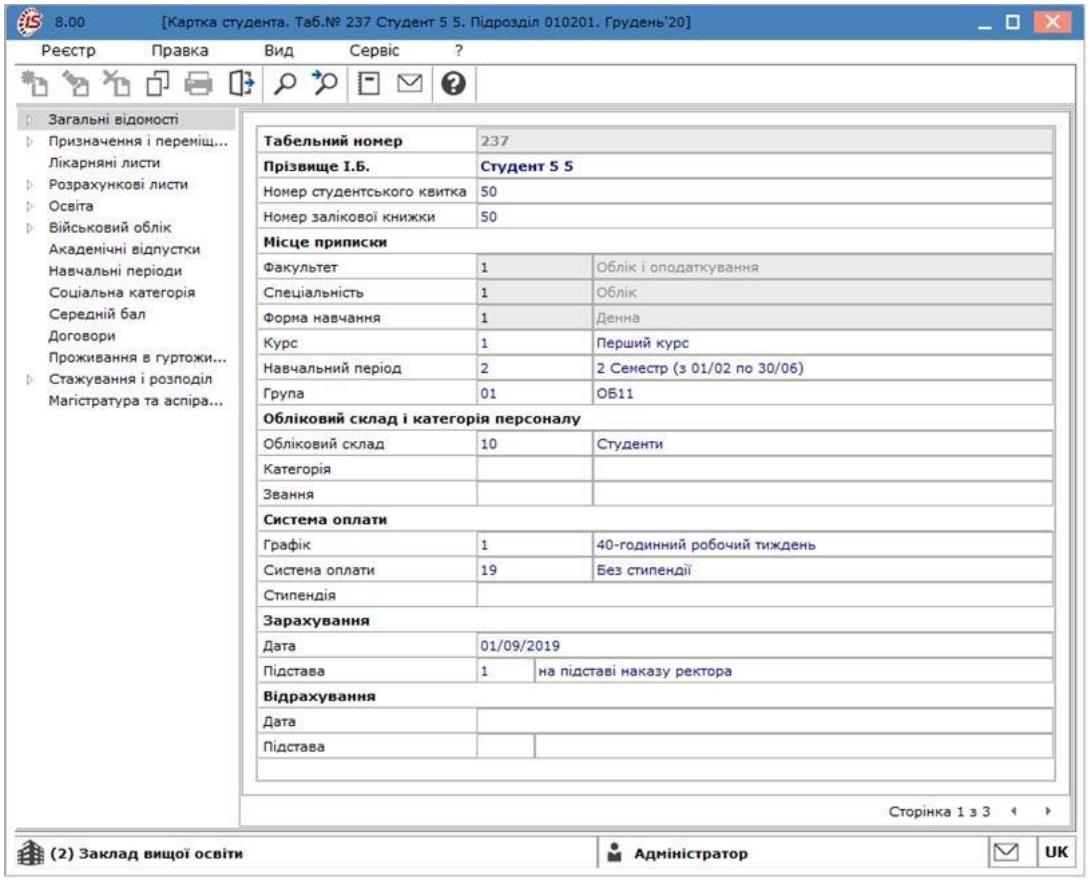


Рисунок 1.5 – Картка студента

— Розрахунок стипендії, інших нарахувань, розрахунки за гуртожиток (Рисунок 1.6).

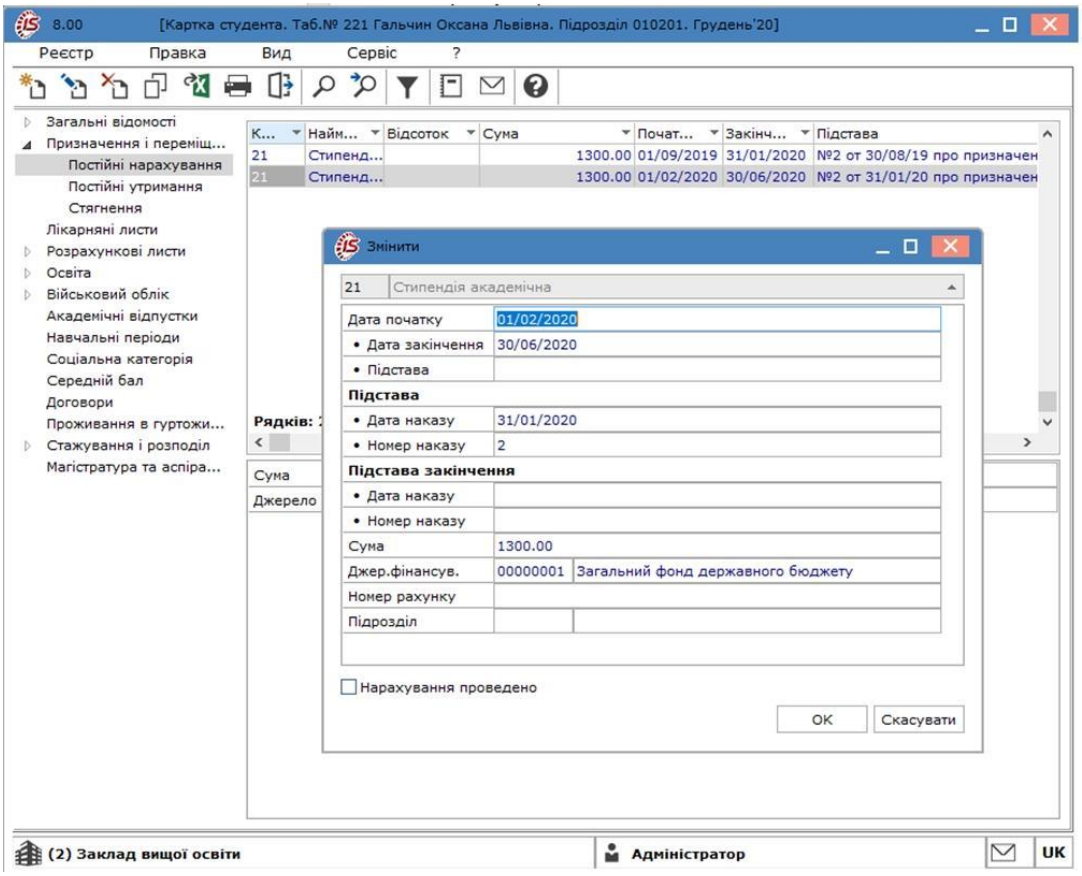


Рисунок 1.6 – Нарахування стипендії

— Формування графіків екзаменаційної сесії.

— Формування екзаменаційних відомостей, ведення обліку результатів іспитів і заліків, облік оцінок успішності студентів.

— Формування звітності згідно законодавства.

— Опрацювання відомості по успішності.

ПЕРЕВАГИ СИСТЕМИ

— Має інтуїтивно-зрозумілий інтерфейс користувача.

— Присутній багатокористувацький режим.

— Наявність української мови.

— Присутня можливість експорту/імпорту даних.

НЕДОЛІКИ

— Відсутність крос-платформності.

— Відсутність автентифікації з використанням соціальних облікових записів.

— Система має обмежений доступ.

1.4. Постановка завдання

Розглянувши автоматизовані інформаційні системи: «Універсальна система обліку», «Кадри Плюс Україна», «Управління життєвим циклом студентів», проведемо порівняльний аналіз цих програмних продуктів. Можна помітити ряд переваг та недоліків.

Переваги полягають в тому, що кожен з програмних продуктів має багатокористувацький режим, інтуїтивно-зрозумілий інтерфейс, наявність української мови та можливість експорту/імпорту даних. А недоліки в тому, що програмні продукти не є крос-пратформними та немає можливості автентифікації з використанням соціальних облікових записів.

Результати порівняння цих систем також проводилося за критеріями, які занесено до таблиці «Порівняльна характеристика програмних продуктів аналогічного призначення» в додатку А. Слід відмітити, що відповідність системи обраному критерію відмічаємо «+».

Враховуючи отримані дані виконаємо постановку завдання:

— необхідно розробити програмний продукт, що являє собою десктопний додаток, який має містити дані про здобувачів навчальної групи, їх досягнення (сертифікати за курси) та допомагати керівнику групи при формуванні характеристик для військкомату, випускових характеристик і надавати оперативний доступ до потрібних даних;
— також додаток має надавати інтуїтивно-зрозумілий інтерфейс;
— наявність української мови;
— можливість зберігати сформовані характеристики в форматі Microsoft Word (.docx).

2. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

2.1. Вимоги до програмного продукту

Після детального дослідження предметної області можна перейти до визначення основних вимог щодо системи, яка розробляється в дипломній роботі.

Автоматизована інформаційна система надає багато цікавих рішень та переваг для вирішення проблеми обліку студентів, а аналіз ринку аналогів програмних продуктів даної предметної області допоміг побачити непотрібність певних технічних рішень, проблеми з виглядом інтерфейсу та непотрібною функціональністю, що зробили б систему занадто складною.

Програмний продукт реалізує автоматизовану інформаційну систему обліку студентів групи, що полегшить збереження інформації та суттєво знизить ймовірність помилок при аналізі інформації, що міститься у документах та розробці нових документів на основі наявної інформації.

Система має бути представлена у вигляді десктопного додатку та включати в себе базу даних.

ФУНКЦІОНАЛЬНІ ВИМОГИ

Функціональні вимоги – це формулювання того, як система має поводитися. Вони визначають, що має робити система, щоб задовольнити потреби чи очікування користувача.

Функціональні вимоги можуть розглядатися як функції, які виявляє користувач.

Функціональні вимоги складаються з двох частин: функції та поведінки. Функція – це те, що виконує система. Поведінка залежить від того, як система це робить [7].

Розглянемо функціональні вимоги до розроблюваної системи.

Керівник повинен мати змогу виконувати такі функції:

— вносити, редагувати, видаляти інформацію про студентів (ПІБ, телефон, адреса і ін.);
— вносити інформацію про батьків студента (ПІБ, телефон, з можливість додавання декількох номерів);
— вносити інформацію про сертифікати, грамоти отримані студентом;
— виконувати пошук за різними критеріями (прізвище, місцевість, стать і ін.);
— формувати характеристики двох видів (у військкомат та випускну).

Відповідно до вимог, формування характеристик повинно бути частково автоматизованим та з можливість формування текстового документу.

НЕФУНКЦІОНАЛЬНІ ВИМОГИ

Нефункціональні вимоги – це обмеження, накладені на систему, які визначають її атрибути якості. Зазвичай вони позначаються такими прикметниками, як безпека, продуктивність і масштабованість.

Нефункціональні вимоги важливі, оскільки вони допомагають забезпечити відповідність системи потребам користувача [8].

Нефункціональні вимоги можна розділити на дві категорії: покращення (безпека, надійність, швидкодія, зручність у використанні); вдосконалення (маштабування, відновлюваність) властивостей системи [9].

Види нефункціональних вимог [10]:

— Вимоги до інтерфейсу (Interface Requirements). Апаратні інтерфейси (Hardware Interfaces) – апаратні інтерфейси необхідні для підтримки системи, включаючи логічну структуру, фізичні адреси і очікувану поведінку. Інтерфейси ПЗ (Software Interfaces) – інтерфейси програмного забезпечення з якими аплікація повинна взаємодіяти. Комунікаційні інтерфейси (Communications Interfaces) – інтерфейси для комунікацій (взаємодії) з іншими системами та/або пристроями.
— Апаратні та програмні вимоги (Hardware/Software Requirements) – опис апаратної та програмної платформ, необхідних для роботи (і підтримки) системи.

— Операційні вимоги (Operational Requirements): безпека та конфіденційність (Security and Privacy); надійність (Reliability); відновлювальність (Recoverability); продуктивність (Performance); потенціал (Capacity); збереження даних (Data Retention); керування помилками (Error Handling); правила перевірки (Validation Rules); узгоджені стандарти (Convention Standards).

2.2. Аналіз та візуалізація вимог на діаграмі прецедентів

Unified Modeling Language (UML) – уніфікована мова моделювання.

UML описує об'єкт в єдиному заданому синтаксисі, її правила будуть зрозумілими для всіх, хто знайомий з цією графічною мовою.

Діаграма прецедентів – в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Діаграми прецедентів відображають елементи моделі варіантів використання.

Діаграма прецедентів використовує два основних елементи:

— Actor (учасник) – множина логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Учасником може бути людина, роль людини в системі чи інша система, підсистема або клас, які представляють щось поза сутністю.

— Use case (прецедент) – опис окремого аспекту поведінки системи з точки зору користувача. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

Прецеденти дозволяють специфікувати функціональну поведінку розроблюваної системи та отримати відповідь на запитання, що має робити системи, проте не визначають реалізацію цієї поведінки системи – не торкаються питань, яким чином відповідні функції реалізуються [11].

Для проектованої системи розроблено діаграму прецедентів (Додаток Б). Визначивши прецеденти, виконаємо детальний аналіз сценаріїв використання:

ПРЕЦЕДЕНТ: РЕДАГУВАННЯ ГРУП

Передумова: необхідність змінити дані відповідної групи.

Післяумова: дані збережені до бази даних.

Сценарій (додавання нової групи):

1. Перехід до вкладки з групами.
2. Натиснути на кнопку «Додати групу».
3. Ввести назву групи.
4. Натискання на кнопку “Зберегти”.

Сценарій (зміна назви групи):

1. Перехід до вкладки з групами.
2. Натискання правої кнопки миші на відповідну групу.
3. Вибір пункту “Редагування”.
4. Зміна назви групи.
5. Натискання на кнопку “Зберегти”.

Сценарій (видалення групи):

1. Перехід до вкладки з групами
2. Натискання правої кнопки миші на відповідну групу.
3. Вибір пункту “Видалення”.
4. Підтвердження через діалогове вікно.

ПРЕЦЕДЕНТ: РЕДАГУВАННЯ СТУДЕНТІВ

Передумова: необхідність змінити оновити список студентів у групі.

Післяумова: дані про список студентів збережені до бази даних.

Сценарій (додавання нового студента):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку.
3. Обрати пункт меню “Додати студента”.
4. Ввести дані про студента.
5. Натискання на кнопку “Зберегти”.

Сценарій (зміна даних студента):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку.
3. Обрати пункт меню “Редагувати студента”.
4. Ввести оновлені дані про студента.
5. Натискання на кнопку “Зберегти”.

Сценарій (видалення студента):

1. Перехід до вкладки з групами..
2. Натисканням на іконку групи перейти до списку.
3. Обрати пункт меню “Видалити студента”.
4. Підтвердження через діалогове вікно.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СЕРТИФІКАТІВ

Передумова: необхідність перегляду усіх або окремого сертифіката.

Післяумова: виведення інформації про сертифікати.

Сценарій(перегляд окремого сертифіката):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів..
3. Обрати пункт меню “Переглянути сертифікати студента”
4. У списку сертифікатів обрати потрібний.

Сценарій(перегляд усіх сертифікатів):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати пункт меню “Переглянути сертифікати студента”.

ПРЕЦЕДЕНТ: РЕДАГУВАННЯ СЕРТИФІКАТІВ СТУДЕНТА

Передумова: необхідність змінити або внести нові дані до списку сертифікатів окремого студента.

Післяумова: дані про сертифікати студента збережено до бази даних.

Сценарій(додавання нового сертифіката):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.

3. Обрати пункт меню “Переглянути сертифікати студента”.
4. Натиснути на значок нового сертифіката.
5. Ввести дані.
6. Натиснути на кнопку “Зберегти”.

Сценарій(редагування сертифіката):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати пункт меню “Переглянути сертифікати студента”.
4. Обрати пункт “Редагування” на відповідному сертифікаті.
5. Оновити дані.
6. Натиснути на кнопку “Зберегти”.

Сценарій(видалення сертифіката):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати пункт меню “Переглянути сертифікати студента”.
4. Обрати пункт “Видалення” на відповідному сертифікаті.
5. Підтвердити видалення у діалоговому вікні.

ПРЕЦЕДЕНТ: ФОРМУВАННЯ ХАРАКТЕРИСТИК

Передумова: необхідність виконати певні дії для створення або оновлення характеристик.

Післяумова: роздрукований файл характеристики, оновлення на додавання даних.

Сценарій(вибір даних):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати відповідного студента та перейти до позначки “Характеристики”.
4. Вибір даних для нової характеристики.
5. Збереження даних.

Сценарій(збереження у форматі .docx):

1. Перехід до вкладки з групами.

2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати відповідного студента та перейти до позначки “Характеристики”.
4. Обрати характеристику та натиснути на “Зберегти”.
5. Вибір розташування файлу.

Сценарій(друк):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати відповідного студента та перейти до позначки “Характеристики”.
4. Обрати характеристику та натиснути на “Друк”.

Сценарій(додавання даних):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Обрати відповідного студента та перейти до позначки “Характеристики”.
4. Додавання даних для нової характеристики.
5. Збереження даних.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СТУДЕНТІВ

Передумова: необхідність перегляду усього списку студентів або окремого студента.

Післяумова: виведення інформації.

Сценарій(перегляд окремого студента):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Натиснути на відповідного студента.

Сценарій(перегляд усіх студентів):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.

ПРЕЦЕДЕНТ: ПОШУК СТУДЕНТА

Передумова: необхідність знайти інформацію про студента по відомим параметрам.

Післяумова: виведення результатів пошуку.

Сценарій (Пошук студента):

1. Перехід до вкладки з групами.
2. Натисканням на іконку групи перейти до списку студентів.
3. Вибір параметру пошуку.
4. Введення даних.
5. Натискання на “Пошук”.

Отже, було визначено основні прецеденти «Перегляд студентів», «Редагування даних про студентів», «Редагування груп», «Перегляд сертифікатів», «Редагування сертифікатів», «Формування характеристик», «Пошук студента».

Прецедент «Перегляд студентів», являє собою команду для перегляду студентів групи та включає в себе прецеденти «Окремий студент», «Список студентів».

Прецедент «Редагування даних про студентів», являє собою команду для редагування даних студентів групи та включає в себе прецеденти «Додавання нового», «Зміна даних», «Видалення».

Прецедент «Редагування груп», являє собою команду для редагування інформації про групу та включає в себе прецеденти «Додавання нової», «Зміна даних», «Видалення».

Прецедент «Перегляд сертифікатів», являє собою команду для перегляду сертифікатів студентів групи та включає в себе прецеденти «Окремий сертифікат», «Список сертифікатів».

Прецедент «Редагування сертифікатів», являє собою команду для редагування даних про сертифікати студента та включає в себе прецеденти «Додавання нового», «Зміна даних», «Видалення».

Прецедент «Формування характеристик», являє собою команду для формування характеристик студента та включає в себе прецеденти «Вибір даних», «Збереження у форматі .docx», «Додавання даних», «Друк».

Прецедент «Пошук студента», являє собою команду для пошуку студента зі списку групи та включає в себе прецеденти «За місцем проживання», «За гендером», «За Прізвищем».

2.3. Вимоги до інтерфейсу

Прототипування – один з ранніх етапів створення АІС. Прототип полегшує спілкування між розробниками, конкретизує техзавдання і допомагає в плануванні подальшої роботи [12].

Прототип – це інтерактивний начерк, чернетка майбутнього програмного продукту. На ньому схематично зображуються основні елементи і їх відгук на дії користувача.

Прототип використовують на ранніх стадіях розробки для презентації ідеї зацікавленим сторонам.

Розрізняють такі типи прототипування [13]:

— швидке (rapid prototyping, throw away prototyping) – у найкоротші терміни створюється прототип, який жодним чином не ввійде до складу готової системи. Швидке прототипування необов'язково виконується в рамках тієї ж платформи і тих самих технологій, що й майбутня система. Крайнім випадком є паперове прототипування, коли нариси інтерфейсу виконуються на папері від руки. Інший метод передбачає використання програм розробки графічного користувацького інтерфейсу (GUI Builder) для створення прототипу, ззовні схожого на цільову систему, але з відсутніми її функціями (click dummy). Для підготовки прототипу графічного інтерфейсу користувача можуть використовуватися просто HTML-файли. Для прототипування програмного забезпечення використовують мови високого рівня абстракції (Java, Perl, Python, Haskell або ін.), а при реалізації – іншу, машинно-орієнтовану (C, C++) для підготовки акуратного документованого коду;

— еволюційне (evolutionary prototyping, breadboard prototyping) – послідовне створення макетів, що поступово наближаються до реальної готової системи. Цей підхід застосовують за відсутності необхідних вимог до системи, вони формуються впродовж її реалізації;
— інкрементальне – кінцевий продукт створюється як сукупність відокремлених прототипів, що зрештою об’єднуються в готову систему;
— екстремальне – процес розробки, найчастіше Web-додатків, що розподіляється на три фази, кожна з яких базується на попередній. Перша з них – розроблення статичного прототипу. Під час другої фази програмуються екрани, створюється повнофункціональний інтерфейс при симуляції рівня сервісів. Сервіси реалізуються під час третьої фази.

Отже, проаналізувавши вимоги до зовнішнього вигляду програмного продукту, було визначено, що додаток буде складатися з таких компонентів:

— Головний екран, містить в собі календар та список груп.
— Екран групи, містить в собі список студентів та рядок пошуку.
— Картка студента, містить в собі всю інформацію про студента та функціональні кнопки.
— Вікно сертифікатів, містить в собі список сертифікатів студента та функціональні кнопки.
— Картка сертифікату, містить в собі всю інформацію про сертифікат та функціональні кнопки.

Для проектування був використаний сервіс Figma. Figma – це хмарний багатоплатформовий сервіс для дизайнерів інтерфейсів і web-розробників, з яким можна працювати безпосередньо в браузері.

Результатом став прототип програмного продукту, який було додано в додаток В.

3. РОЗРОБКА БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Створення ORM моделі

ORM (Object-Relational Mapping) модель – це програмний підхід, який дозволяє програмістам працювати з базою даних, використовуючи об'єктно-орієнтований підхід до програмування [14].

ORM модель дозволяє програмістам створювати класи, які відповідають таблицям бази даних, і зберігати та вибирати дані з бази даних, використовуючи ці класи, замість прямої роботи з SQL запитами. Це дозволяє розробникам працювати з даними, як з об'єктами, забезпечуючи більш зручний та інтуїтивно зрозумілий спосіб доступу до даних.

ORM модель дозволяє зменшити кількість коду, необхідного для роботи з базою даних, та дозволяє зосередитися на бізнес-логіці додатку, замість деталей роботи з базою даних. ORM модель також дозволяє підтримувати інкапсуляцію даних та сприяє підвищенню безпеки захисту даних.

ORM модель містить наступні основні елементи:

— Об'єкти – це основні одиниці даних, з якими працює ORM модель. Кожен об'єкт відображає рядок у таблиці бази даних.

— Класи – класи є шаблонами для створення об'єктів, які відповідають таблицям бази даних. Класи визначають поля та методи, які використовуються для доступу до даних.

— Поля – це атрибути, які визначають структуру даних, що зберігаються в об'єктах. Кожне поле відображає колонку у таблиці бази даних.

— Методи – методи використовуються для зчитування та запису даних у базу даних. Вони можуть бути визначені в класах та використовуватися для взаємодії з об'єктами.

— Запити – запити є основним механізмом взаємодії з базою даних в ORM моделі. Вони дозволяють вибирати, зберігати та оновлювати дані в базі даних.

— Посередники – посередники (або драйвери) використовуються для

забезпечення взаємодії між ORM моделлю та базою даних. Вони забезпечують переклад SQL запитів у команди, які розуміє база даних та забезпечують взаємодію з базою даних через API.

Виходячи з опису предметної області, можна визначити, що в базі даних зберігатимуться такі сутності: Групи, Студенти, Сертифікати, Характеристики.

Сутність «Студенти» визначає студента, який навчається в групі. Властивості:

— Id студента – ідентифікатор студента.
— Ім’я – використовується для зберігання імені студента.
— Прізвище – використовується для зберігання прізвища студента.
— По-батькові – використовується для зберігання по-батькові студента.
— Дата народження – використовується для зберігання дати народження студента.
— Стать – використовується для зберігання статі студента.
— Адреса проживання – використовується для зберігання адреси проживання студента.
— Номер телефону – використовується для зберігання номера телефона студента.
— Номер групи – використовується для збереження ідентифікатора групи, в якій навчається студент.

Сутність «Групи» визначає групу, в якій навчається студент. Властивості:

— Id групи – ідентифікатор групи.
— Назва групи – використовується для зберігання назви групи.

Сутність «Сертифікати» визначає всі сертифікати отримані студентом. Властивості:

— Id сертифікату – ідентифікатор сертифікату.
— Дата видачі – використовується для зберігання дати видачі сертифікату.
— Назва сертифікату – використовується для зберігання назви сертифікату.

— Опис – містить детальний опис сертифікату.
— Id студента – використовується для збереження ідентифікатора студента, який отримав сертифікат.
— Фото сертифікату – містить фото сертифікату.

Сутність «Характеристики» визначає характеристику, що необхідно сформувати студенту. Властивості:

— Id характеристики – ідентифікатор характеристики.
— Тип характеристики – використовується для формування типу характеристики.
— Хто заповнив характеристику – містить інформацію про особу, що сформувала характеристику.
— Дата заповнення – використовується для зберігання дати заповнення характеристики.
— Назва закладу – містить інформацію про назву закладу.
— Заголовок – використовується для зберігання назви характеристики.
— Основний опис характеристики – містить детальний опис характеристики.
— Студент, якому належить характеристика – використовується для збереження ідентифікатора студента, який отримав характеристику.

Отже, на підставі опису встановлених вимог, виділено такі зв’язки.

Сутність «Групи» зв’язана із сутністю «Студенти» по типу «один до багатьох», оскільки група може мати багато студентів

Сутність «Студенти» зв’язана з сутністю «Сертифікати» по типу «один до багатьох», оскільки один студент може мати кілька сертифікатів.

Також сутність «Студенти» пов’язана із сутністю «Характеристики» по типу «один до багатьох», оскільки з один студент може мати декілька характеристик. Дослідивши сутності та зв’язки, було побудовано ER діаграму та розміщено в додатку Г.

3.2. Підхід CodeFirst

CodeFirst – це один з підходів в Entity Framework (EF), який дозволяє розробникам визначати модель даних та взаємодіє з нею, не звертаючись до бази даних.

Умовно цей процес можна представити у вигляді трьох етапів (чи послідовності «перекладів» з однієї форми до іншої):

1. Класи
2. Entity Framework
3. База даних

У CodeFirst визначається модель даних у вигляді класів, які стають таблицями у базі даних. Для кожної властивості у класі можна встановити тип даних та обмеження. EF автоматично створить базу даних на основі цих класів.

Переваги CodeFirst полягають у тому, що:

— Швидкий розгортання проекту: розробник може швидко створювати моделі об'єктів та їх взаємозв'язки, а потім отримати автоматично згенеровану базу даних.
— Простота розробки: CodeFirst спрощує розробку, оскільки розробник може працювати з об'єктами, не зважаючи на базу даних.
— Флексібільність: CodeFirst дозволяє розробникам змінювати моделі об'єктів, не змінюючи схему бази даних.
— Тестування: CodeFirst дозволяє легко тестувати функціональність додатка, оскільки розробник може використовувати фейкові бази даних для тестування.
— Зменшення кількості коду: CodeFirst дозволяє розробникам зменшити кількість коду, оскільки вони можуть працювати з об'єктами, а не з SQL-запитами.

Один з головних переваг підходу CodeFirst – це простота використання та швидкість розробки. Можна зосередитися на моделюванні даних, не звертаючись до деталей роботи з базою даних.

Ще одна перевага підходу CodeFirst – це можливість використовувати

міграції. Міграції дозволяють вносити зміни в модель даних та автоматично застосовувати їх до бази даних.

Недоліки підходу CodeFirst – це обмеження в можливостях моделювання даних. Деякі складні сценарії можуть виявитися складними або неможливими для реалізації в CodeFirst. Крім того, якщо використовувати CodeFirst для існуючої бази даних, доведеться вручну додавати або змінювати таблиці в базі даних.

Загалом, підхід CodeFirst є дуже корисним для розробки простих додатків зі стандартною моделлю даних. Для складніших додатків можуть знадобитися інші підходи в EF [15].

4. ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ

4.1. Опис мови програмування

C# – проста, статично типізована, об’єктно-орієнтована мова програмування від компанії Microsoft [16].

C# входить до сімейства мов програмування C, синтаксис мови буде знайомим програмістам, що працювали з C, C++, Java та JavaScript.

Перша версія мови C# була створена в 1998-2001 роках, групою інженерів Microsoft під керівництвом Андреса Гейлсберга та Скотта Вільтаумота, як основна мова програмування платформи Microsoft .Net.

C# увібрав в себе найкращі властивості попередників – мов C, C++, Modula, Object Pascal, спираючись на практичний досвід їх використання. Деякі проблематичні моделі, що до цього використовувались у мовах програмування, зокрема множинне спадкування класів(яке використовується у мові C++), були свідомо виключені.

В багатьох відношеннях мова C# дуже схожа на Java, це відображено в синтаксисах та основних поняттях цих мов програмування.

C# є дуже сучасною мовою програмування з безліччю функцій та можливостей. Вона має сильну систему типізації, що дозволяє розробникам створювати більш безпечний та надійний код [17].

Крім того, C# підтримує асинхронне програмування, що дозволяє створювати додатки, які працюють швидше та ефективніше.

C# також має вбудовану підтримку для багатьох популярних технологій, таких як Windows Forms, ASP.NET, WPF, Xamarin.

Вона також має багато ресурсів та інструментів для розробників, таких як Microsoft Visual Studio, який забезпечує зручну розробку та налагодження додатків мовою C#.

Взагалі, C# є дуже потужною мовою програмування, яка може бути використана для створення широкого спектру додатків – від невеликих ігор до великих корпоративних систем.

4.2. Технологія WPF

Windows Presentation Foundation (WPF) – це платформа наступного покоління для побудови клієнтських програм Windows з візуально привабливими можливостями взаємодії з користувачем [18].

За допомогою WPF можна створювати широкий спектр як автономних, так і розміщених у браузері програм.

В основі WPF лежить векторна система візуалізації, яка не залежить від дозволу та створена з розрахунком на можливості сучасного графічного обладнання. WPF розширює базову систему повним набором функцій розробки додатків, у тому числі мовою XAML (Extensible Application Markup Language), елементами керування, прив'язкою даних, макетом, дво- та тривимірною графікою, анімацією, стилями, шаблонами, документами, мультимедіа, текстом та оформленням. WPF входить до складу Microsoft .NET Framework і дозволяє створювати програми, які включають інші елементи бібліотеки класів .NET Framework.

Для роботи з WPF підійде будь-яка .NET-сумісна мова. Цей список містить багато мов: C#, VB, C++, Ruby, Python, Delphi (Prism), Lua та багато інших [19].

Основні переваги технології WPF:

— Розділення логіки та відображення: WPF дозволяє відокремлювати логіку додатку від його візуальної частини, що дозволяє розробникам легко модифікувати та налагоджувати візуальний інтерфейс користувача.
— Інтерактивність: технологія надає можливість створювати багатофункціональні та інтерактивні елементи управління, які можуть взаємодіяти з користувачем, наприклад, перетягувати, розтягувати, переміщати тощо.
— Анімація та графіка: WPF має потужні засоби для створення анімації та графіки, включаючи можливість створення 2D- та 3D-моделей, використання різноманітних ефектів та фільтрів.
— Масштабованість: вона дозволяє створювати додатки, які можуть

працювати на різних пристроях з різними розмірами екрану, і при цьому зберігати високу якість візуального представлення.

Взагалі, WPF є технологією для розробки десктопних додатків, яка дозволяє створювати привабливі та функціональні графічні інтерфейси користувача

4.3. Паттерн MVVM

MVVM – це паттерн проектування програмного забезпечення, що використовується в розробці користувацьких інтерфейсів [20] .

Паттерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку програми від візуальної частини (подання). Цей патерн є архітектурним, тобто він задає загальну архітектуру програми.

Цей патерн був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в різних технологіях, у тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

MVVM складається з трьох компонентів:

— Модель (Model) – це компонент, який містить бізнес-логіку та дані, з якими працює програма.

— Представлення (View) – це компонент, який відповідає за відображення даних користувачеві та обробку користувацьких дій.

— В'юмодель (ViewModel) – це компонент, який служить посередником між «Моделлю» та «Представленням». Він отримує дані від «Моделі» та передає їх у «Представлення» у відповідному форматі. Крім того, В'юмодель також обробляє взаємодію користувача з «Представленням», трансформуючи введені дані у формат, придатний для обробки «Моделлю».

4.4. Фреймворк Caliburn Micro

Caliburn Micro – це легкий фреймворк для розробки додатків на платформі .NET, зокрема для WPF (Windows Presentation Foundation).

Він надає просту та зрозумілу модель програмування, яка спрощує розробку графічного інтерфейсу користувача (GUI) та забезпечує розширення MVVM (Model-View-ViewModel).

Caliburn Micro базується на ідеях, запроваджених у фреймворку Caliburn, але спрощує його використання та зменшує його розмір. Він надає реалізацію шаблону MVVM, який дозволяє відокремити логіку додатку від його представлення, забезпечуючи легкість тестування та обслуговування.

Основні принципи Caliburn Micro включають автоматичну прив'язку даних (automatic data binding), автоматичне виявлення взаємозв'язків між представленням (View) та моделлю (ViewModel), а також розпізнавання дій користувача.

Він також надає механізми для розширення та налаштування поведінки елементів у користувацькому інтерфейсі.

Загалом, Caliburn Micro є потужним фреймворком для розробки додатків на платформі .NET, зокрема для WPF, і спрощує роботу з шаблоном MVVM, допомагаючи покращити розподіл обов'язків та підтримувати чистий та організований код.

4.5. Система управління базою даних

Microsoft SQL Server – це багатофункціональне рішення для роботи з базами даних з можливостями бізнес-аналітики і зберігання даних, а також аварійного відновлення і резервного копіювання [21].

Microsoft SQL Server забезпечує вбудовані у бази даних можливості обробки даних в оперативній пам'яті при будь-яких робочих навантаженнях, більш швидше отримання результатів аналізу даних з використанням

знайомих засобів аналітики, а також використання рішень по обробці великих даних на корпоративному рівні.

Microsoft SQL Server має багато переваг, серед яких:

— Надійність та стабільність: SQL Server є дуже надійною та стабільною системою, яка дозволяє зберігати та керувати великими обсягами даних.
— Швидкість: система може працювати з великою кількістю запитів та даних, що дозволяє обробляти великі обсяги даних швидко та ефективно.
— Відмінна підтримка: Microsoft забезпечує відмінну підтримку для SQL Server, включаючи регулярні оновлення та виправлення помилок.
— Легкість використання: SQL Server має досить простий інтерфейс, який дозволяє легко створювати, змінювати та видаляти бази даних, таблиці та інші об'єкти.
— Можливості розширення: система має велику кількість можливостей для розширення, включаючи підтримку мов програмування, таких як C#, Python, R, інтеграцію з іншими системами та інше.
— Висока безпека: SQL Server має вбудовану систему безпеки, яка дозволяє забезпечити безпеку даних та запобігти несанкціонованому доступу до бази даних.
— Масштабованість: система дозволяє масштабувати базу даних, що дозволяє збільшувати кількість даних та користувачів без втрати продуктивності.

4.6. Технологія Entity Framework 6

Entity Framework 6 (EF6) є об'єктно-реляційною маппером (ORM) для .NET Framework, розробленим компанією Microsoft [22].

Це бібліотека, яка дозволяє розробникам .NET легко та зручно працювати з базами даних, використовуючи об'єктно-орієнтований підхід.

EF6 надає можливість створення моделі даних у вигляді класів, які можуть взаємодіяти з базою даних за допомогою LINQ (Language-Integrated Query) або SQL.

Основні переваги EF6 включають:

— Легкість використання: EF6 дозволяє створювати моделі даних у вигляді класів, що дуже легко інтегрується з .NET Framework.
— Безпека даних: технологія забезпечує вбудовану підтримку для безпеки даних, що дозволяє захистити базу даних від несанкціонованого доступу.
— Швидкість: EF6 дозволяє виконувати запити до бази даних швидко та ефективно, що дозволяє забезпечувати високу продуктивність додатків.
— Масштабованість: вона підтримує масштабування додатків, що дозволяє збільшувати кількість даних та користувачів без втрати продуктивності.
— Можливості розширення: EF6 має велику кількість можливостей для розширення, включаючи підтримку різних провайдерів баз даних, підтримку різних мов програмування, таких як C#, VB.NET, та інші.
— Підтримка відносин між об'єктами: технологія дозволяє використовувати відносини між об'єктами у програмі та автоматично керувати залежностями між ними.
— Легке керування: EF6 дозволяє легко керувати базою даних.

4.7. Середовище розробки

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів [23].

Visual Studio містить в собі набір інструментів, які дозволяють розробникам створювати, тестувати та налагоджувати програми різних типів та мов програмування, включаючи C#, C++, Visual Basic, Iron Python тощо.

Visual Studio також містить в собі різноманітні функції для роботи з версіями, збирання та розгортання програмного забезпечення.

Microsoft Visual Studio має багато переваг, особливо для розробників програмного забезпечення. Ось деякі з найбільш важливих переваг:

— Інтегроване середовище розробки: Visual Studio містить в собі повний набір інструментів для розробки програмного забезпечення, що дозволяє розробникам працювати в єдиному інтерфейсі.
— Підтримка багатьох мов програмування: середовище підтримує багато мов програмування, таких як C++, C#, Visual Basic, F# та інші.
— Відладчик: Visual Studio має потужний відладчик, який дозволяє розробникам знайти та виправити помилки в програмному забезпеченні.
— Підтримка версійного контролю: Visual Studio підтримує популярні системи версійного контролю, такі як Git, SVN та TFS.
— Велика спільнота користувачів: середовище має велику спільноту користувачів та активний форум, де можна знайти відповіді на будь-які питання та проблеми.
— Розширення та плагіни: Visual Studio має велику кількість розширень та плагінів, що дозволяють розширити його можливості та додати нову функціональність.

5. ОПИС СТВОРЕНОГО ПРОГРАМНОГО ПРОДУКТУ, ЙОГО СТРУКТУРИ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ

5.1. Побудова та опис діаграми класів

Діаграма класів – це основа будь-якої документації програмного проекту. Діаграми класів називають «статичними діаграмами», оскільки на них показано класи разом з методами і атрибутами, а також статичний взаємозв'язок між ними: те, яким класам «відомо» про існування яких класів, і те, які класи «є частиною» інших класів, – але не показано методи, які при цьому викликаються [24].

КЛАС

Клас визначає атрибути і методи набору об'єктів. Всі об'єкти цього класу (екземпляри цього класу) мають спільну поведінку і однаковий набір атрибутів (кожен з об'єктів має свій власний набір значень).

У UML класи позначаються прямокутниками з назвою класу, у цих прямокутниках у вигляді двох «відсіків» може бути показано атрибути і операції класу.

АТРИБУТИ

У UML атрибути показуються щонайменше назвою, також може бути показано їх тип, початкове значення і інші властивості.

Крім того, атрибути може бути показано з областю видимості атрибута:

— + відповідає публічним (public) атрибутам;

— # відповідає захищеним (protected) атрибутам;

— – відповідає приватним (private) атрибутам.\

ОПЕРАЦІЇ

Операції (методи) також показуються принаймні назвою, крім того, може бути показано їх параметри і типи значень, які буде повернуто.

Операції, як і атрибути, може бути показано з областю видимості:

— + відповідає публічним (public) операціям;

— # відповідає захищеним (protected) операціям;

— – відповідає приватним (private) операціям.

ШАБЛОНИ

Серед класів можуть бути шаблони, значення, які використовуються для невизначеного класу або типу. Тип шаблону визначається під час ініціалізації класу (тобто, під час створення об’єкта).

АСОЦІАЦІЇ КЛАСІВ

Класи можна співвіднести (пов’язати) один з одним у декілька способів: узагальнення, асоціація, агрегація, композиція.

Наслідування є однією з фундаментальних основ об’єктно-орієнтованого програмування, у якому клас «отримує» всі атрибути і операції класу, нащадком якого він є, і може перевизначати або змінювати деякі з них, а також додавати власні атрибути і операції.

У UML узагальнення буде показано у вигляді лінії, яка поєднує два класи, зі стрілкою, яку спрямовано від базового класу.

Асоціація означає взаємозв’язок між класами, вона є базовим семантичним елементом і структурою для багатьох типів «з’єднань» між об’єктами.

У UML асоціації позначаються лініями, що з’єднують класи, які беруть участь у зв’язку, крім того, може бути показано роль і численність кожного з учасників зв’язку. Численність буде показано у вигляді діапазону [мін..макс] невід’ємних чисел, зірочка (*) на боці максимального значення позначає нескінченність.

Агрегації є особливим типом асоціацій, за якого два класи, які беруть участь у зв’язку не є рівнозначними, вони мають зв’язок типу «ціле-частина».

У UML агрегації буде показано асоціаціями, у яких з боку цілої частини буде намальовано ромб.

Композиції – це асоціації, які відповідають дуже сильній агрегації. Це означає, що у композиціях ми також маємо справу з співвідношеннями ціле-частина, але тут зв’язок є настільки сильним, що частини не можуть існувати без цілого. Вони існують лише у межах цілого, після знищення цілого буде знищено і його частини.

У UML композиції буде показано як асоціації з зафарбованим ромбом з боку цілого.

В таблиці 5.1 наведено значення полів для класу «Student».

Таблиця 5.1 – Поля класу «Student»

Назва поля	Призначення/Атрибути
Name	Ім’я студента
Lastname	Прізвище студента
Patronymic	По-батькові студента
DateOfBirth	День народження студента
Gender	Стать студента
Address	Адреса студента
PhoneNumber	Номер телефону студента
GroupId	Id групи
Id	Id студента
Characteristics	Список характеристик
Certificate	Список сертифікатів

В таблиці 5.2 наведено значення полів для класу «Group».

Таблиця 5.2 – Поля класу «Group»

Назва поля	Призначення/Атрибути
Id	Id групи
GroupName	Назва групи
Students	Студенти групи

В таблиці 5.3 наведено значення полів для класу «Characteristic».

Таблиця 5.3 – Поля класу «Characteristic»

Назва поля	Призначення/Атрибути
Id	Id характеристики
Type	Тип характеристики
Creator	Хто заповнив характеристику
Date	Дата заповнення

Institution	Назва закладу
Header	Заголовок
Description	Основний опис характеристики
StudentId	Той студент, якому належить характеристика

В таблиці 5.4 наведено значення полів для класу «Certificate».

Таблиця 5.4 – Поля класу «Certificate»

Назва поля	Призначення/Атрибути
Id	Id сертифіката
Date	Дата видачі
Institution	Установа, що видала сертифікат
Header	Заголовок сертифікату
Description	Опис сертифікату
StudentId	Id студента
CertificatePhoto	Фото сертифікату

В таблиці 5.5 наведено значення полів для класу «IRepository<T>».

Таблиця 5.5 – Поля класу «IRepository<T>»

Назва поля	Призначення/Методи
Create()	Додати новий об’єкт
RemoveAsync(T en)	Видалити об’єкт асинхронно
CreateAsync()	Додати новий об’єкт асинхронно
Update()	Редагувати

В таблиці 5.6 наведено значення полів для класу «GenericRepository<T>».

Таблиця 5.6 – Поля класу «GenericRepository<T>»

Назва поля	Призначення/Методи
Context	Контекст, де зберігається таблиця та відбувається взаємодія з ORM моделлю

В таблиці 5.7 наведено значення полів для класу «ApplicationContext».

Таблиця 5.7 – Поля класу «ApplicationContext»

Назва поля	Призначення/Методи
Certificates	Таблиця сертифікатів
Characteristics	Таблиця характеристик

Groups	Таблиця груп
Student	Таблиця студентів
ApplicationContext()	Конструктор за замовчуванням

Отже, проаналізувавши та описавши всі класи, було визначено, що кожен з класів має у собі усі необхідні поля та методи.

Класи поділено на дві групи. Перша група – це класи моделі, друга – класи репозиторіїв.

Клас Student описує клас студента. Містить у собі усі необхідні поля, що описують студента навчального закладу. Також цей клас інкапсулює у собі набір усіх наявних сертифікатів та характеристик поточного здобувача освіти. Кожен студент підв’язаний до своєї навчальної групи.

Клас Group містить у собі назву групи а також набір студентів, що в ній навчаються.

Клас Characteristic інкапсулює в собі усі поля та властивості для опису характеристики студента. Основними є: тип характеристики, особа, що написала, дата створення та опис із заголовком. Кожна характеристика прив’язана до конкретного студента.

Клас Certificate має у собі опис та фотографію сертифіката.

Клас ApplicationContext бере на себе відповідальність за з’єднання із ORM моделлю бази даних. Зберігає в собі усі таблиці бази даних.

Інтерфейс Irepository<T> використовується для зменшення програмних залежностей у коді. Маю у собі усі CRUD операції над базою даних.

GenericRepository<T> реалізує інтерфейс Irepository<T>. Інкапсулює у собі таблицю відповідного класу та контекст бази даних.

StudentRepository<Student> – клас для роботи з таблицею студентів.

CertificateRepository<Certificate> – клас для роботи з таблицею сертифікатів.

CharacteristicsRepository<Characteristic> – клас для роботи з таблицею характеристик.

GroupRepository<Group> – клас для роботи з таблицею груп.

Докладна схема структури класів, із зазначенням параметрів та методів кожного класу, а також структурою міжкласової взаємодії зображена в додатку Д.

5.2. Структура та функціональність системи

Структура АІС – внутрішня організація системи при поділі її на частини, виявлення зв'язків між цими частинами [25].

Інформаційні системи, належать до класу складних, тому для кожної такої системи існує проблема декомпозиції – поділу її на простіші складові (елементи) та подання у вигляді тих чи інших (скажімо, якомога більших) її частин [26].

Будь-яка АІС поділяється на функціональну та забезпечувальну частини, які, у свою чергу, поділяються на простіші елементи – підсистеми, котрі також припускають подальший поділ.

До функціональної частини належать ті елементи системи, які визначають її функціональні можливості, а саме: призначення, виконувані управлінські функції та функції з обробки інформації.

До забезпечувальної частини системи належать об’єкти (матеріальні та інші засоби, інструментарій), з допомогою яких виконуються функції системи, тобто реалізується функціональна її частина.

У функціональній частині АІС вирізняють такі елементи: функціональні підсистеми, блоки, або комплекси завдань та окремі завдання.

Функціональна підсистема – це відносно самостійна частина системи, яку виокремлено за певною ознакою, що відповідає конкретним функціям і завданням управління. Цю підсистему можна розглядати як самостійну систему, що характеризується певним цільовим призначенням, підпорядкованістю, відокремленістю інформаційної бази, методичною спрямованістю обчислень економічних показників і спеціалізацією робіт.

Функціональна структура АІС має розглядати проблеми інформації кінцевих користувачів, які змінюються в умовах ринку, та відтворювати зміст,

а також функцій ті що керують конкретним економічним об'єктом.

В АІС має бути гнучка структура і вона має бути відкритою системою, але повинна відокремити ті змін у нашу модель та зробити нарощування функціональних частин як потребується.

Розробка функціональної частини слід проводити згідно з вимогами висунутими у даних формування задач до системи яка автоматизується до вимог розробки картки студента.

Проведемо виділення функцій та виділення функціональних модулів:

Управління студентами:

— Додавання нових студентів.
— Видалення студентів.
— Редагування даних студентів.

Управління сертифікатами:

— Додавання нових сертифікатів студента.
— Видалення сертифікатів.
— Редагування даних сертифікатів.

Управління записами:

— Створення карток студентів.
— Створення карток сертифікатів.

Сервіс обробки:

— Створення характеристики.

Удосконалення та розвиток функціональної частини АІС відбувається в напрямку розробки та впровадження нових підсистем, засобів масової інформації, завдань.

Удосконалення функціональних даних АІС забезпечує повне та ефективне виконання функцій, а також підвищує функціональну застосовність АІС.

5.3. Тестування та розгортання

РОЗГОРТАННЯ

Розгортання програмного забезпечення (ПЗ) відноситься до процесу розміщення і налаштування ПЗ та його залежностей для виконання в реальному середовищі.

Цей процес може включати розгортання на фізичних серверах, в хмарних сервісах або контейнерах.

Основна мета розгортання – забезпечити належну роботу ПЗ та готовність його до використання.

Діаграма розгортання розроблюваної системи розміщена в додатку Е.

Розгортання системи описує спосіб фізичного розміщення її компонентів та залежності між ними:

— Клієнтські пристрої: користувач отримує доступ до системи через персональний комп'ютер або ноутбук. Він підключається до системи за допомогою програми.

— Встановлена програма: програма на комп’ютері керівника групи має рівень відображення, репозиторії та рівень взаємодії із базою даних, обробляє запити від клієнтських пристроїв та виводить дані з БД.

— База даних: база даних містить інформацію про студентів, їх сертифікати та інші дані. Програма здійснює звернення до бази даних для отримання та збереження даних.

Для безперебійної функціональності системи вимагається задоволення певних технічних вимог. Серед цих вимог обов'язково передбачається встановлення Microsoft SQL Server, який є реляційною базою даних, спеціально розробленою для забезпечення зберігання та обробки великого обсягу даних у структурованому форматі.

Додатково, необхідною є наявність платформи .NET Framework 4.7, яка є розширеною середовищем розробки програмного забезпечення. Ця платформа забезпечує необхідні компоненти та інструменти для виконання програм на мові програмування .NET.

Крім того, вимагається також присутність програми Microsoft Word, що є текстовим процесором, розробленим для створення, редагування та форматування документів у текстовому форматі. Наявність цього інструменту є обов'язковою для забезпечення повноцінної роботи системи та виконання необхідних функцій, що пов'язані з обробкою текстової інформації.

В результаті виконання поставленого завдання була розроблена інформаційна система з обліку студентів. У додатку Є наведено частину програмного коду та посилання на репозиторій.

ТЕСТУВАННЯ

Тестування програмного забезпечення – це процес перевірки відповідності заявлених до продукту вимог та реально реалізованої функціональності, що відбувається шляхом спостереження за його роботою в штучно створених ситуаціях та на обмеженому наборі тестів, вибраних визначеним чином [27].

В широкому сенсі, тестування – це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування та аналіз отриманих результатів.

Тестування є важливим етапом у розробці будь-якої програмної системи. Воно дозволяє виявити та виправити помилки, перевірити відповідність системи вимогам та забезпечити її якість.

Для тестування створеного продукту були використані наступні види тестування:

— Функціональне тестування. Функціональне тестування спрямоване на перевірку функціональності системи. Будуть проведені тестові сценарії, які охоплюють основні функції системи. Під час тестування будуть перевірені такі аспекти, як додавання студентів, внесення даних, формування характеристик тощо.

— Навантажувальне тестування. Навантажувальне тестування виконується з метою визначення продуктивності системи та її здатності працювати при

великому обсязі даних та навантаженні. Буде проведено тести, що моделюють одночасну роботу багатьох користувачів та великі обсяги даних. Результати тестування допоможуть виявити можливі проблеми з продуктивністю системи та при необхідності внести в необхідні зміни для оптимізації роботи системи.

— Модульне тестування. Модульне тестування спрямоване на перевірку окремих модулів або компонентів системи з метою виявлення помилок та підтвердження їх правильної роботи, дозволяє перевірити коректність роботи окремих функцій, методів або класів системи.

В процесі проведення тестування були виявленні помилки та відразу усунуті, а результати тестування занесені в додаток Ж.

5.4. Інструкція користувача

Роботу з системою варто почати з процесу встановлення інсталятора, це можна зробити, двічі клацнувши на файлі з розширенням ".exe".

Після запуску інсталятора, буде можливість вибору папки або шлях для встановлення системи (Рисунок 5.1).

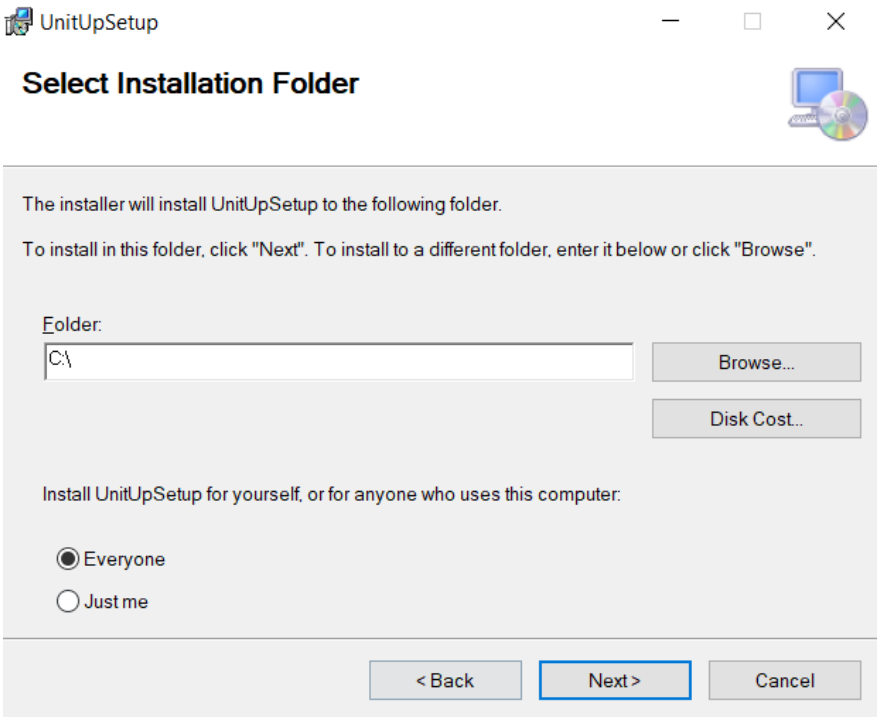


Рисунок 5.1 – Шлях для встановлення системи

Після введення всіх необхідних параметрів інсталятор розпочне процес копіювання файлів та встановлення системи на комп'ютер. Після завершення встановлення буде отримане повідомлення про успішне встановлення програмного забезпечення та додано систему на робочий стіл (Рисунок 5.2).

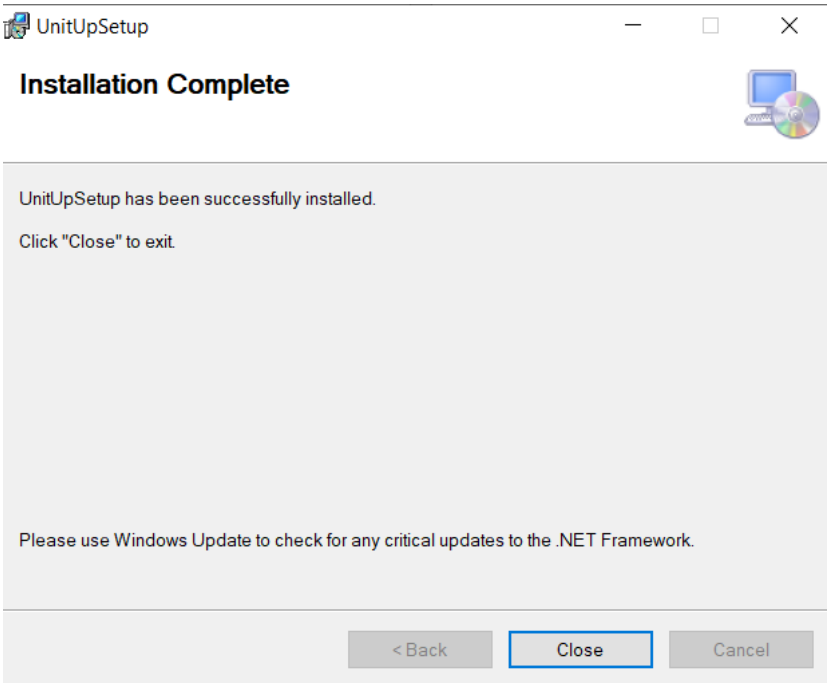


Рисунок 5.2 – Повідомлення про успішне встановлення

Після запуску програми, користувачу відображається основна сторінка, яка ілюстрована на рисунку 5.3.

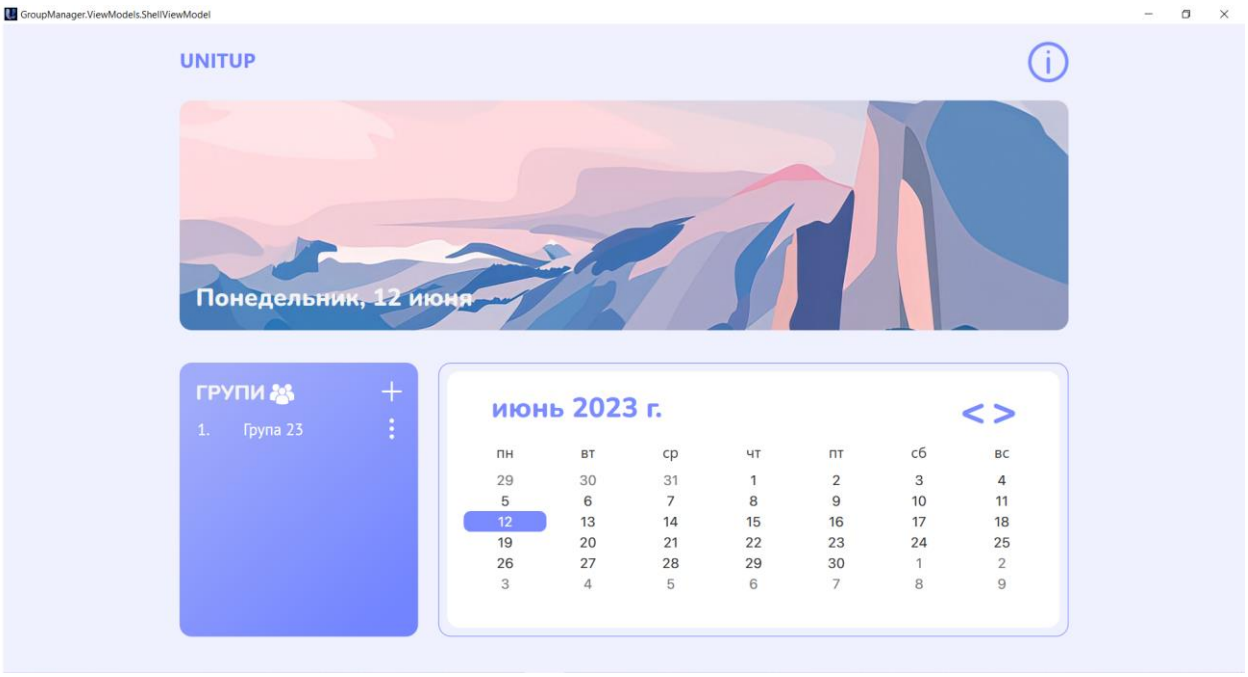


Рисунок 5.3 – Основна сторінка

Ця сторінка є початковою точкою для роботи з системою обліку студентів. Вона містить календар та список груп, що проілюстровані на рисунку.

Поряд зі списком груп розміщена кнопка у формі плюса, яка при натисканні дозволяє користувачеві додати нову групу. Для кожної групи також доступна кнопка меню з опціями "Редагувати" та "Видалити", як показано на рисунку 5.4.

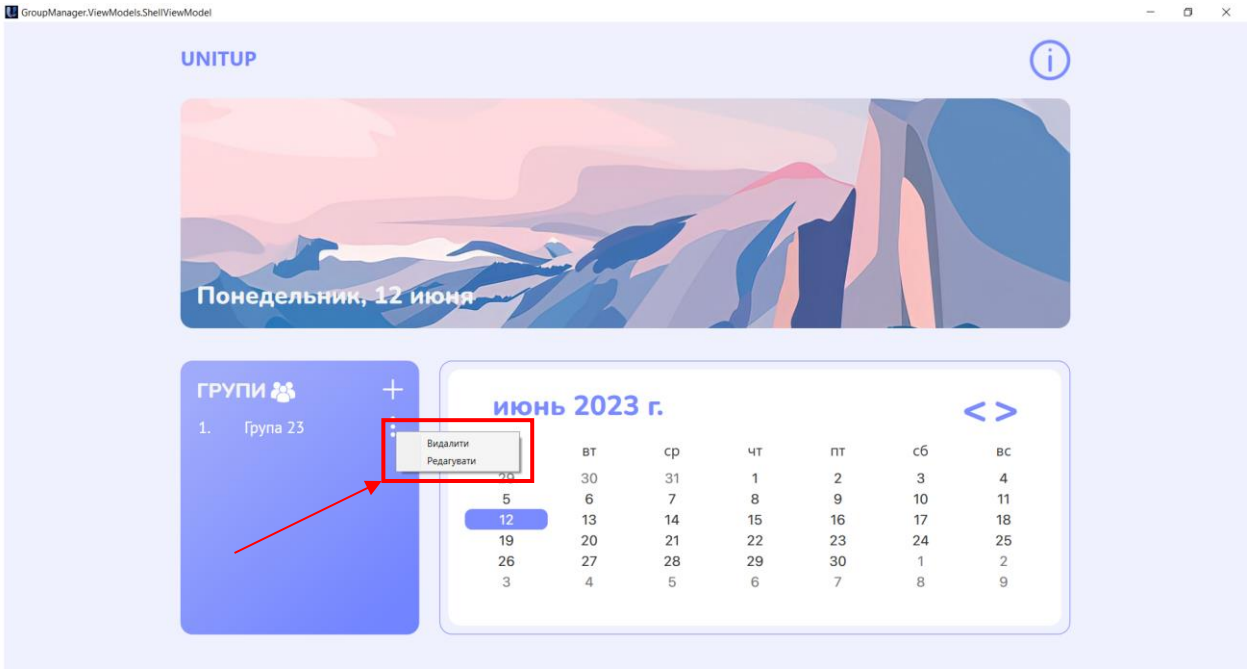


Рисунок 5.4 – Опції меню

Після натискання на будь-яку групу зі списку, користувач переходить на наступну сторінку, де відображається список студентів обраної групи. У верхній частині цієї сторінки розташоване поле для пошуку. Аналогічно до головної сторінки, на цій сторінці також присутня кнопка у формі плюса, яка виконує функцію додавання нових студентів. Кожен студент має кнопку "Видалення" поруч з його інформацією, як показано на рисунку 5.5.

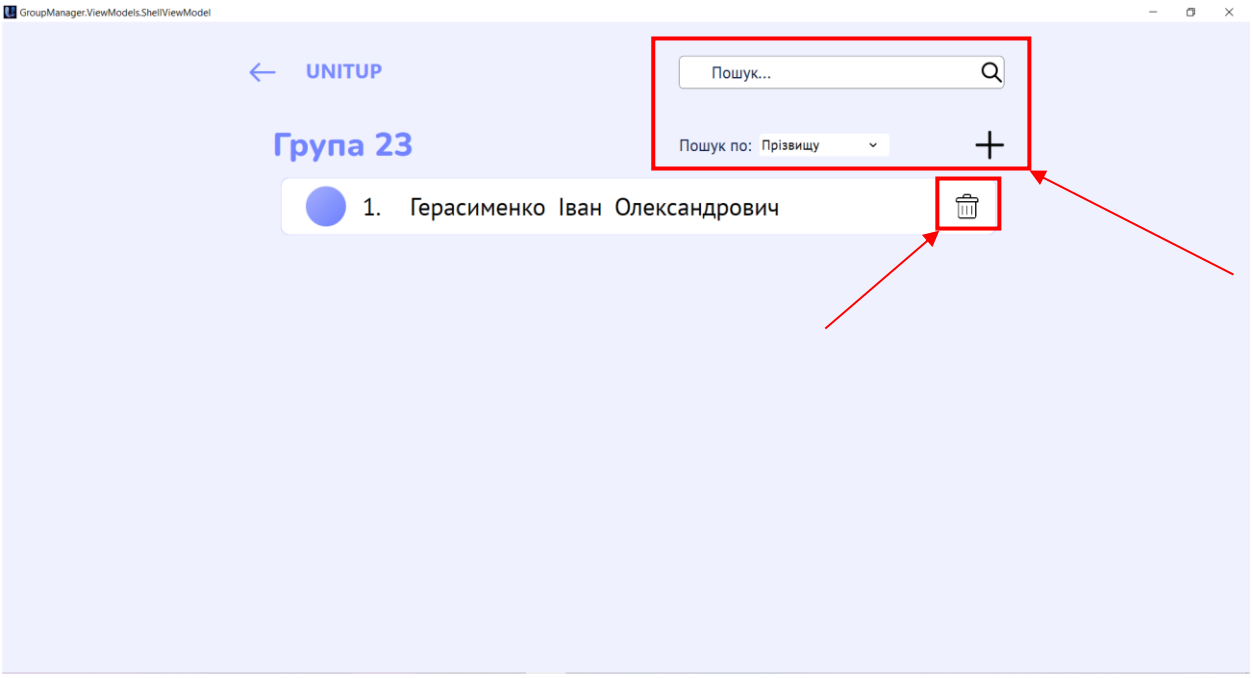


Рисунок 5.5 – Сторінка групи

Подвійне натискання на певного студента відкриває повну інформацію про обраного студента, яка представлена на рисунку 5.6. На цій сторінці також присутня сторінка "Сертифікати" та кнопки "Редагувати" та "Характеристика". У нижній частині сторінки розміщені кнопки для переміщення між студентами. Кнопка "Редагувати" дозволяє користувачеві редагувати інформацію про студента.

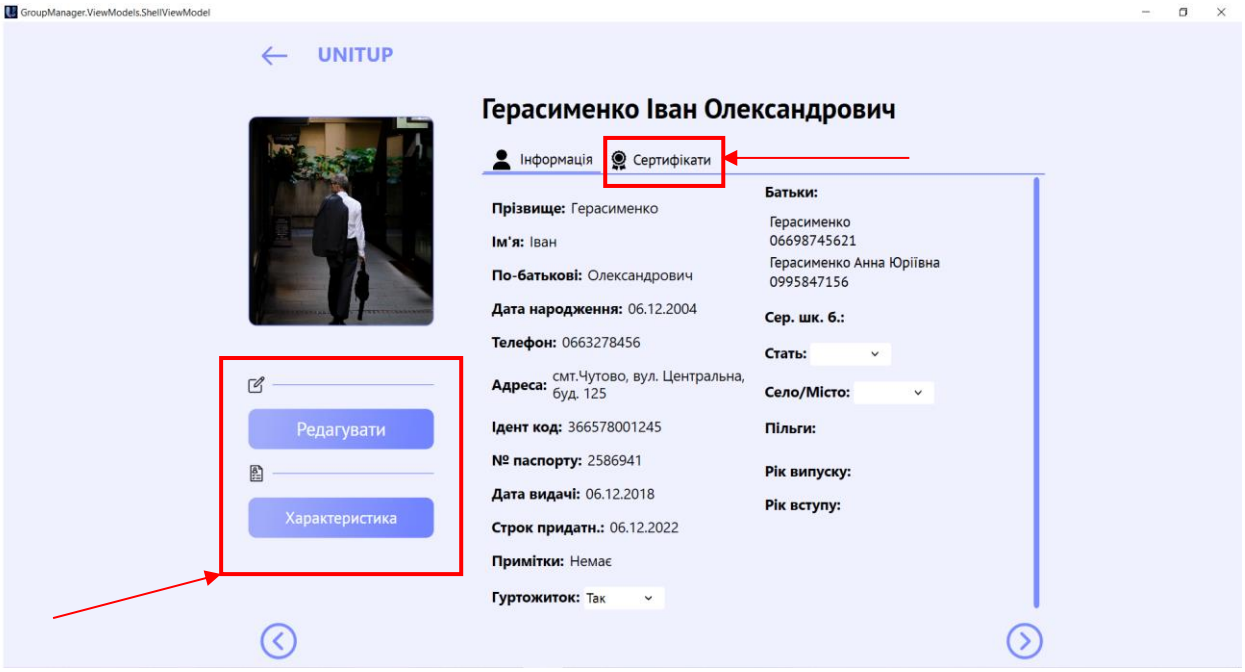


Рисунок 5.6 – Картка студента

При натисканні на сторінка «Сертифікати» відкривається список сертифікатів, які належать студенту, як показано на рисунку 5.7. Аналогічно до попередніх сторінок, на цій сторінці також доступні кнопки для додавання нових сертифікатів та їх видалення.

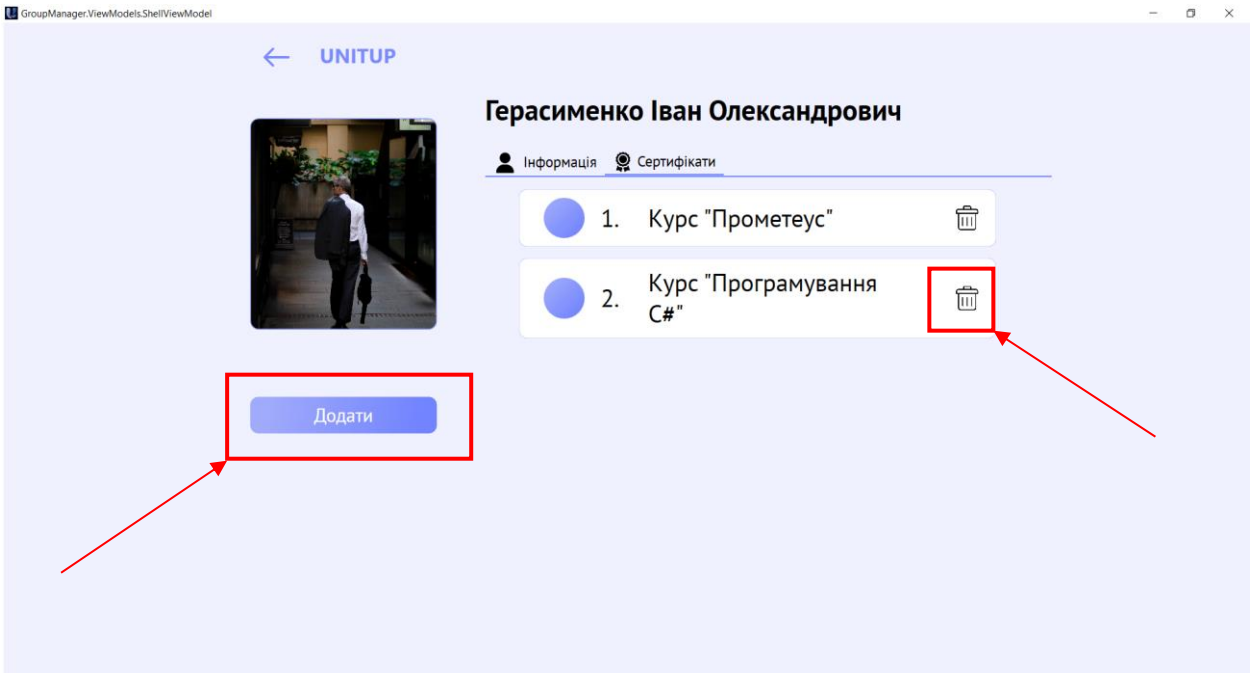


Рисунок 5.7 – Сторінка сертифікатів

При натисканні на кнопку "Характеристика" відкривається нова сторінка, яка призначена для формування характеристик, як показано на рисунках 5.8-5.9.

Рисунок 5.8 – Перший етап формування характеристики

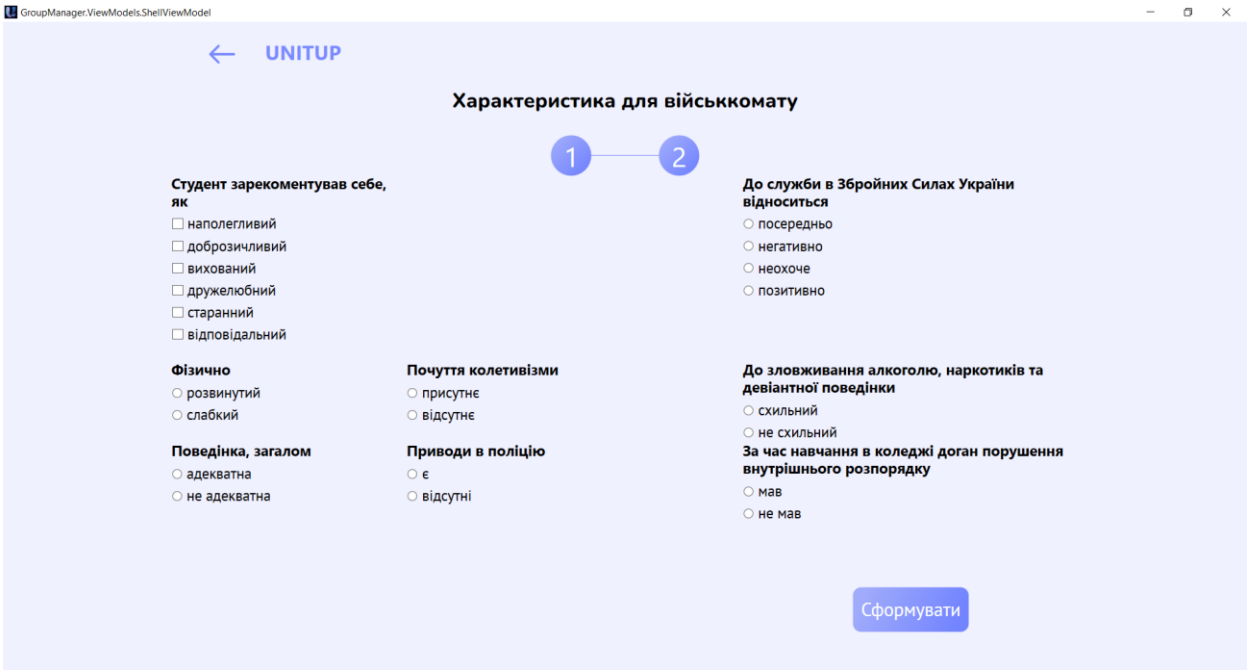


Рисунок 5.9 – Другий етап формування характеристики

Використовуючи вище зазначену послідовність сторінок та їх функціональність, користувач може зручно керувати та взаємодіяти з системою обліку студентів. Цей інтерфейс надає зручний спосіб керування групами та студентами, а також забезпечує доступ до додаткової інформації, такої як сертифікати та характеристики, для кращого ведення обліку студентів. Інші знімки системи можна переглянути в додатку З.

ВИСНОВКИ

Тема розробки інформаційної системи для куратора академічної групи є актуальною і своєчасною, оскільки очевидною є необхідність зберігання і обробки даних студентів, обліку їх досягнень та формування характеристик. В наш час існує безліч видів обліку студентів, які ведуться кураторами, викладачами, завідувачами відділень. Всі дані про студентів зберігаються у журналах груп, відомостях, списках тощо.

В результаті виконання дипломної роботи було виконано ряд завдань:

— зроблено опис предметної області;
— розглянуто та проведено порівняльний аналіз подібних систем, результати занесено в порівняльну таблицю в додатку А;
— побудовано діаграму прецедентів (Додаток Б) і проаналізовано сценарії використання створеної системи;
— створено прототип інтерфейсу системи (Додаток В);
— розроблено базу даних інформаційної системи та побудовано ER діаграму, представлену в додатку Г;
— розроблено діаграму класів та діаграму розгортання, які представлені в додатках Д та Е;
— розроблено інформаційну систему;
— описано програмний продукт, його структуру та функціональні можливості.

З метою поліпшення управління групою та забезпечення ефективності роботи, було розроблено автоматизовану інформаційну систему для керівника групи, яка відповідає основним вимогам до сучасних програмних розробок, для забезпечення автоматизації роботи, зручний інтерфейс.

Це дає право говорити про конкурентоспроможність розробленої програми і можливість практичного її використання для вирішення реальних завдань.

Ця система забезпечує збір та обробку даних про кожного члена групи, дозволяє керівнику формувати випускні характеристики, а також характеристики для військкомату. Знімки екрану та вихідні коди представлені в додатках Є та З.

Для розробки системи було використано мову програмування C#, технологію WPF, архітектуру MVVM. Система управління базою даних – Microsoft SQL Server, для взаємодії з базою даних використовувався Entity Framework 6, а середовищем розробки є Microsoft Visual Studio.

Розроблена система здатна значно підвищити ефективність управління групою, зменшити витрати часу на збір та аналіз інформації, покращити якість прийнятих рішень. Отже, можна зробити висновок, що розроблена автоматизована інформаційна система є корисним допоміжним інструментом в роботі класного керівника.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Автоматизовані інформаційні системи: опис, завдання та особливості. Інформаційний Портал «Пабліш Україна». [Електронний ресурс]. – Режим доступу: URL: https://publish.com.ua/it-ta-web/avtomatizovani-informatsijni-sistemi-opis-zavdannya-ta-osoblivosti.html#typy-informatsiinoi-avtomatychnoi-systemy (дата звернення: 29.10.2023).
2. Класифікація АІС. StudFiles. [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/6658194/page:7/ (дата звернення: 29.10.2023).
3. Стадії та етапи розробки АІС. StudFiles. [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/7195346/page:10/ (дата звернення: 29.10.2023).
4. УСО – Універсальна Система Обліку. Україна. [Електронний ресурс]. – Режим доступу: URL: http://ususoft.com.ua (дата звернення: 29.10.2023).
5. Кадри Плюс Україна. AnDeeSoft. [Електронний ресурс]. – Режим доступу: URL: https://andeesoft.com/ua/kpu/ (дата звернення: 29.10.2023).
6. Управління життєвим циклом студентів. ISPro. [Електронний ресурс]. – Режим доступу: URL: https://ispro.ua/page/upravlinnya-zhittyevim-ciklom-studentiv (дата звернення: 29.10.2023).
7. Що таке функціональні вимоги: приклади, визначення, повний посібник - Рішення Visure. Visure Solutions. [Електронний ресурс]. – Режим доступу: URL: https://visuresolutions.com/uk/блог/функціональні-вимоги/ (дата звернення: 29.10.2023).
8. Що таке нефункціональні вимоги: приклади, визначення, повний посібник – Visure Solutions. Visure Solutions. [Електронний ресурс]. – Режим доступу: URL: https://visuresolutions.com/uk/blog/non-functional-requirements/ (дата звернення: 29.10.2023).
9. Функціональні та Не Функціональні Вимоги. Секрети Шліфування Якості. [Електронний ресурс]. – Режим доступу: URL: http://lvivqaclub.blogspot.com/2008/10/blog-post_17.html (дата звернення: 29.10.2023).

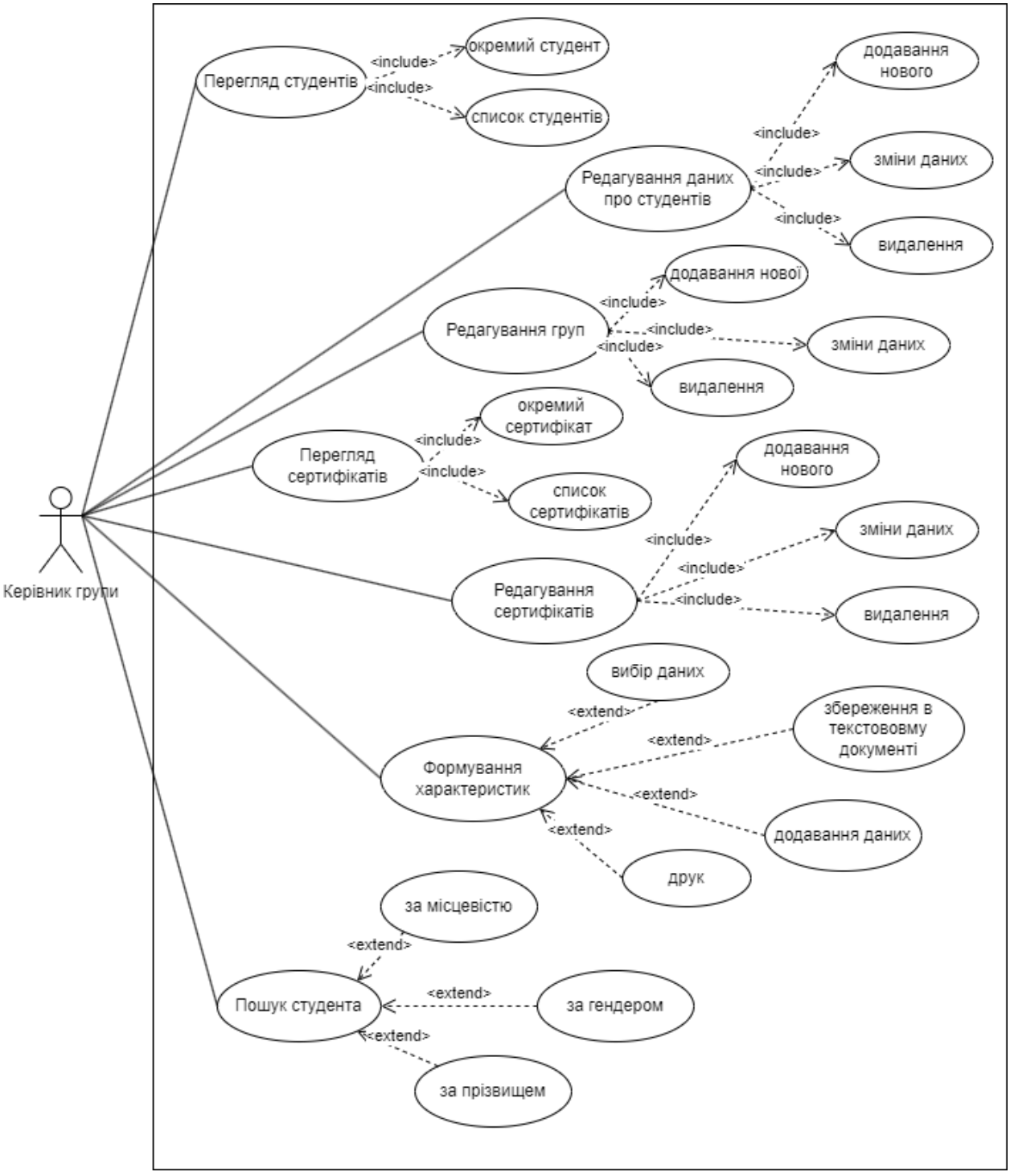
10. Нефункціональні вимоги. nina.az. [Електронний ресурс]. – Режим доступу: URL: https://www.wiki-data.uk-ua.nina.az/Нефункціональні_ вимоги.html (дата звернення: 30.10.2023).
11. Презентація "Діаграми UML. Діаграми прецедентів". Освітній проект «На Урок» для вчителів. [Електронний ресурс]. – Режим доступу: URL: https://naurok.com.ua/prezentaciya-diagrami-uml-diagrami-precedentiv-238715.html (дата звернення: 30.10.2023).
12. АВТОМАТИЗАЦІЯ ПРОТОТИПУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ. Бібліотека Posibniki. Електронна бібліотека підручників онлайн. [Електронний ресурс]. – Режим доступу: URL: https://posibniki.com.ua/post-avtomatizaciya-prototipuvannya--informaciynih-sistem (дата звернення: 30.10.2023).
13. Прототип що це значить. Сучасні взаємовідносини у суспільстві і родинному колі. [Електронний ресурс]. – Режим доступу: URL: https://pyrogiv.kiev.ua/prototip-shho-ce-znachit/ (дата звернення: 23.02.2023).
14. Об'єктно-реляційне відображення. nina.az. [Електронний ресурс]. – Режим доступу: URL: https://www.wiki-data.uk-ua.nina.az/Object-relational_mapping.html (дата звернення: 30.10.2023).
15. Що таке Code First?. Solutions IT. [Електронний ресурс]. – Режим доступу: URL: https://solutions-it.co.il/code-first/ (дата звернення: 30.10.2023).
16. Д. В. Макогон, І. Ю. Лисенко. C# і платформа .NET Core. Підручник з програмування. - К.: Видавничий дім "Інтерсервіс", 2019. - 608 с.
17. Коноваленко І. Програмування мовою C# 6.0. Тернопіль : ТНТУ, 2016. 227 с.
18. Технологія wpf. StudFiles. [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/9508772/page:2/ (дата звернення: 30.10.2023).
19. Джулін Дан. WPF: програмування і візуалізація даних. - К.: Видавництво "Діалог-МІФ", 2014. - 320 с.
20. MVVM. Brander. [Електронний ресурс]. – Режим доступу: URL: https://brander.ua/technologies/mvvm (date of access: 30.10.2023).

21. Microsoft SQL Server. ВЕРСІЯ. [Електронний ресурс]. – Режим доступу: URL: https://www.versiya.com/ua/soft/microsoft/sql-server.html (date of access: 31.10.2023).
22. Фрімен А. Entity Framework 6.0. Практичний посібник для розробників. Apress, 2013. 687 с.
23. Середовище створення програм Microsoft Visual Studio Загальні прийоми роботи. StudFiles. [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/5994722/page:7/ (дата звернення: 31.10.2023).
24. Елементи UML. KDE Documentation. [Електронний ресурс]. – Режим доступу: URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html (дата звернення: 31.10.2023).
25. Структура автоматизованих інформаційних систем. StudFiles. [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/5199402/page:6 (дата звернення: 31.10.2023).
26. Загальна структура ІС, функціональна та забезпечувальна частини. Компоненти системи. Бібліотека економіста. [Електронний ресурс]. – Режим доступу: URL: https://library.if.ua/book/94/6502.html (дата звернення: 31.10.2023).
27. Що таке тестування програмного забезпечення? QALight. [Електронний ресурс]. – Режим доступу: URL: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/ (дата звернення: 31.10.2023).

ДОДАТОК А. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ПРОГРАМНИХ ПРОДУКТІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

	«Універсальна система обліку»	«Кадри Плюс Україна»	«Управління життєвим циклом студентів»	Створений програмний продукт
Адаптивність	+	+	+	+
Багатокористувацький режим	+	+	+	–
Інтуїтивно-зрозумілий інтерфейс користувача	+	+	+	+
Наявність української мови	+	+	+	+
Можливість імпорту/експорту даних	+	+	+	+
Крос-платформність (наявність версій для Linux/MacOS)	–	–	–	+
Автентифікація з використанням соціальних облікових записів (Google/Facebook/Twitter тощо)	–	–	–	–
Наявність довідкових матеріалів та документації	+	+	+	+
Вартість ліцензії	Від 2250	5550-17100	Безкоштовна	Безкоштовна

ДОДАТОК Б. UML ДІАГРАМА ПРЕЦЕДЕНТІВ



ДОДАТОК В. ПРОТОТИП ІНТЕРФЕЙСУ

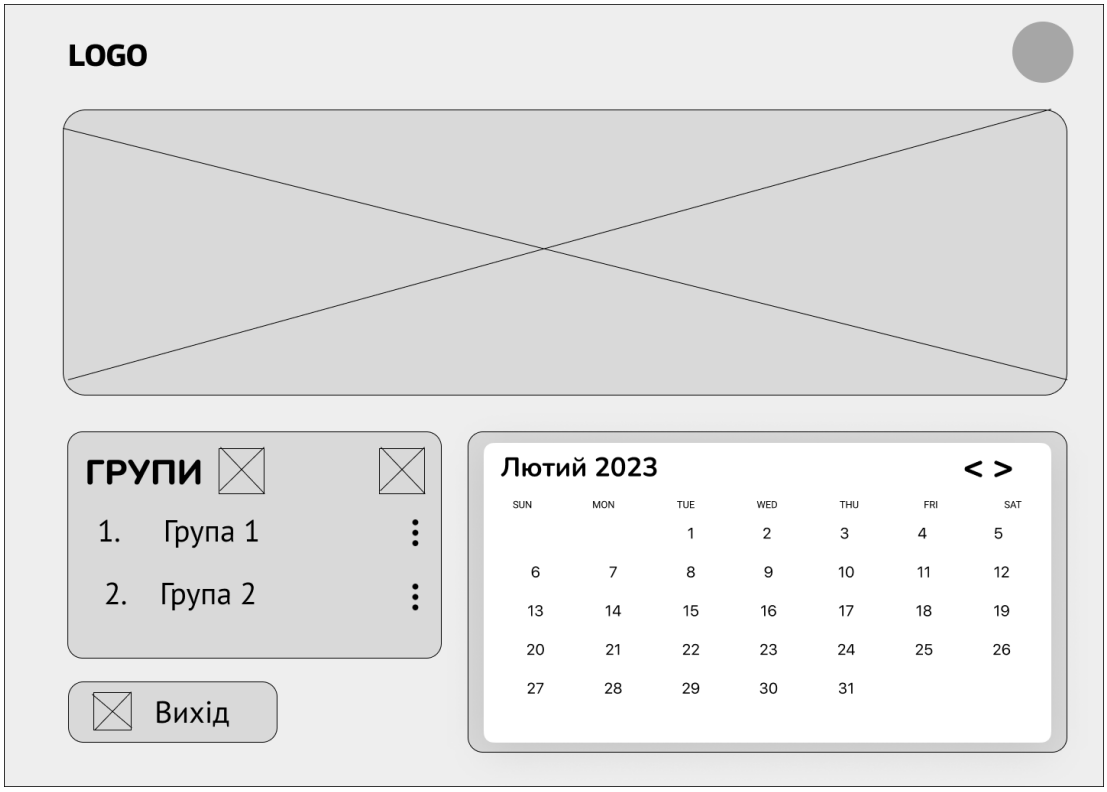


Рисунок В.1 – Головна сторінка

Рисунок В.2 – Список студентів групи

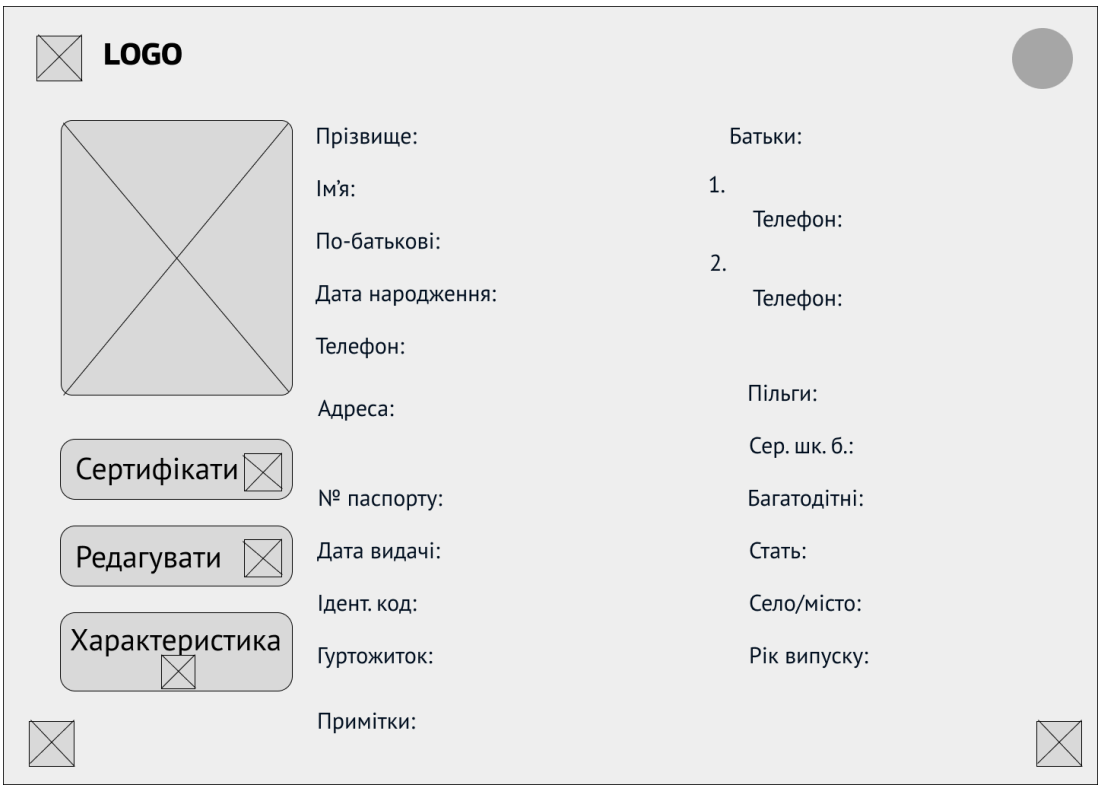


Рисунок В.3 – Карта студента групи

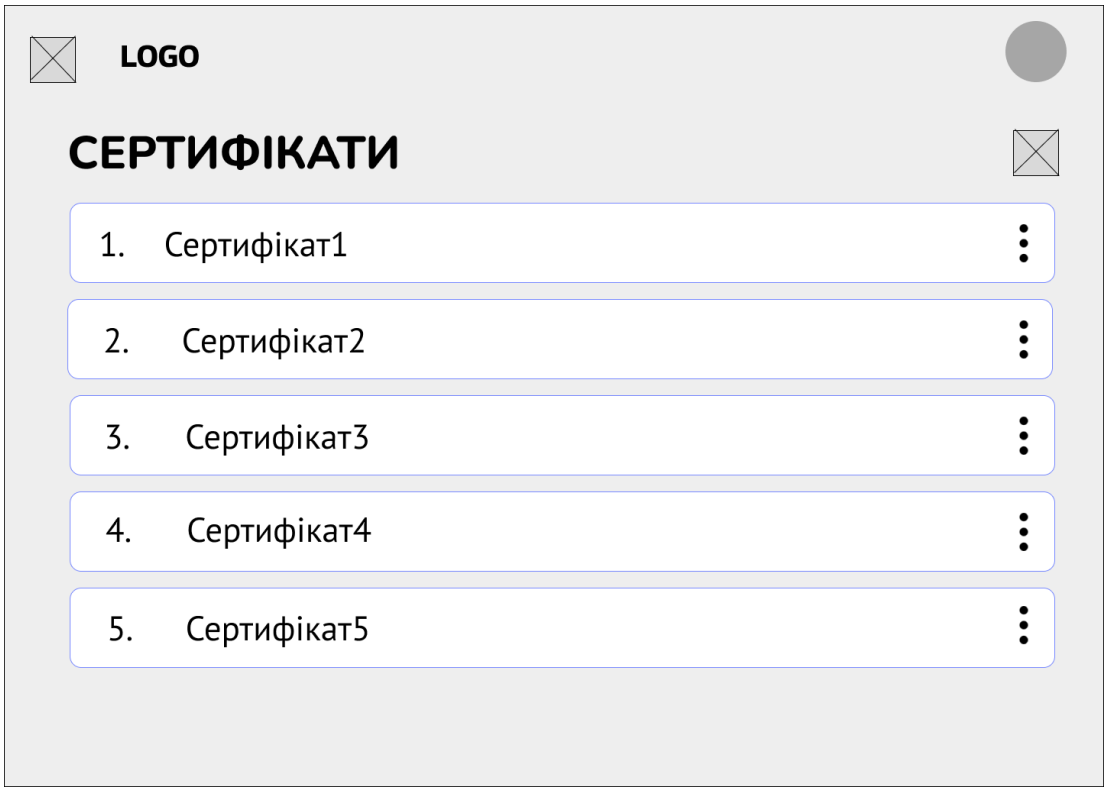


Рисунок В.4 – Список сертифікатів студента

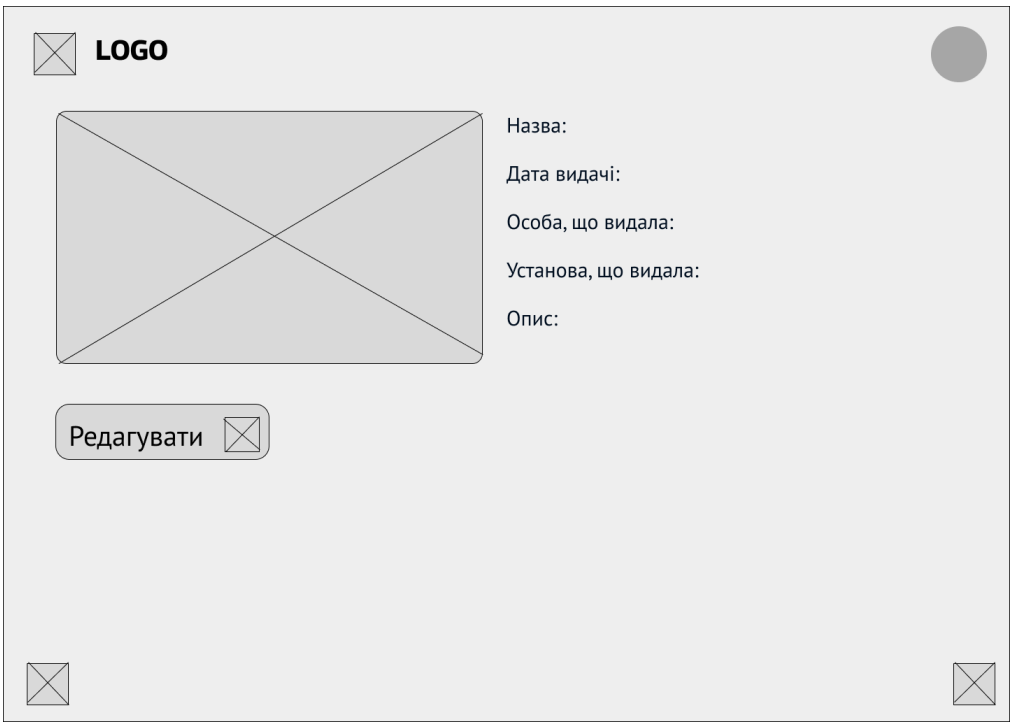
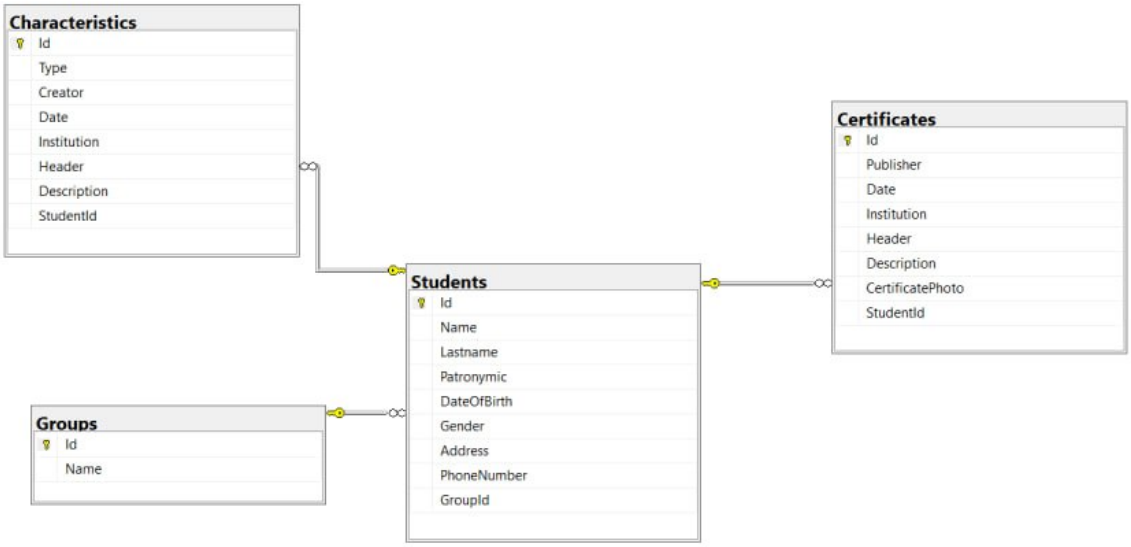
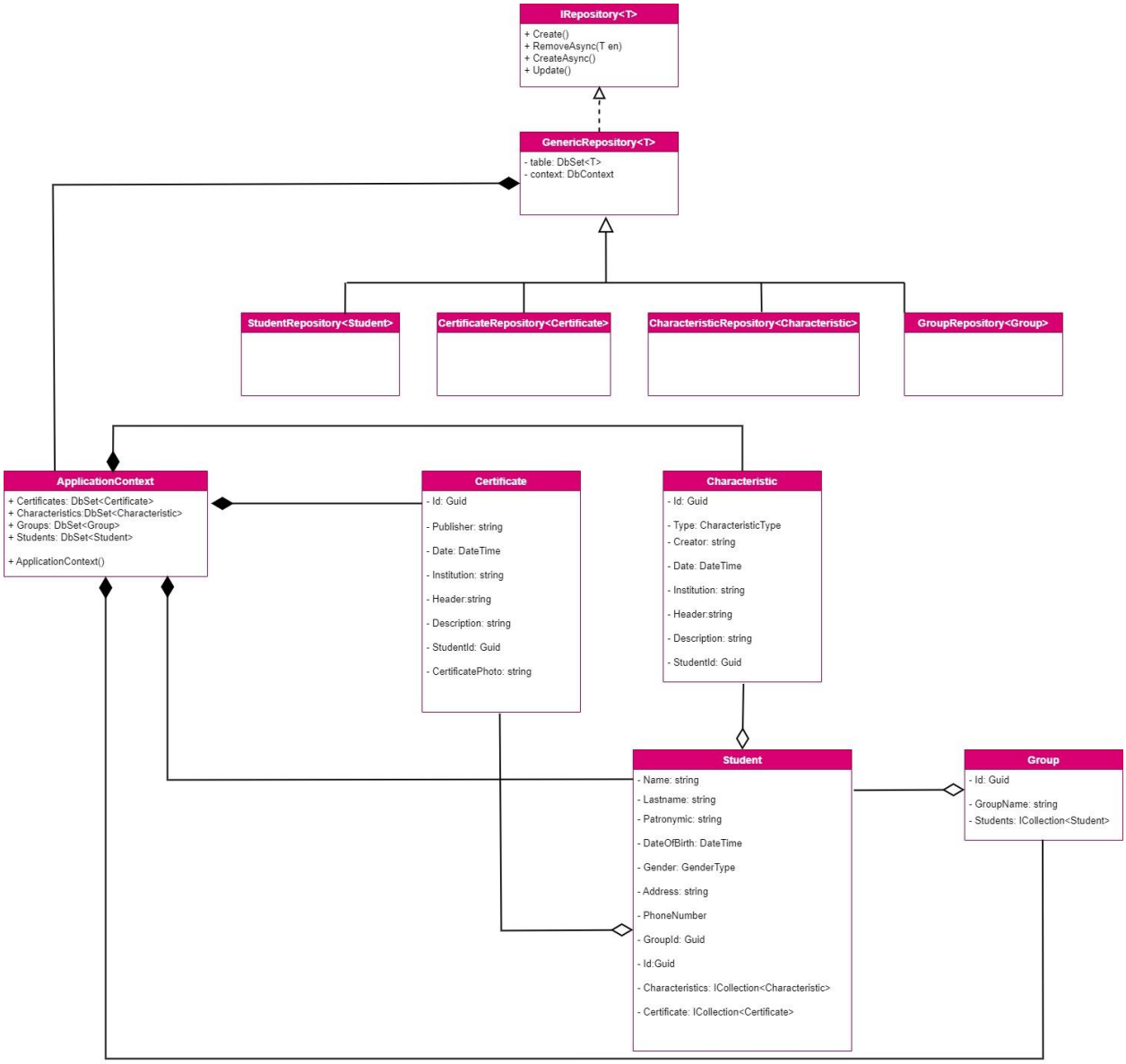


Рисунок В.5 – Карта сертифікату студента

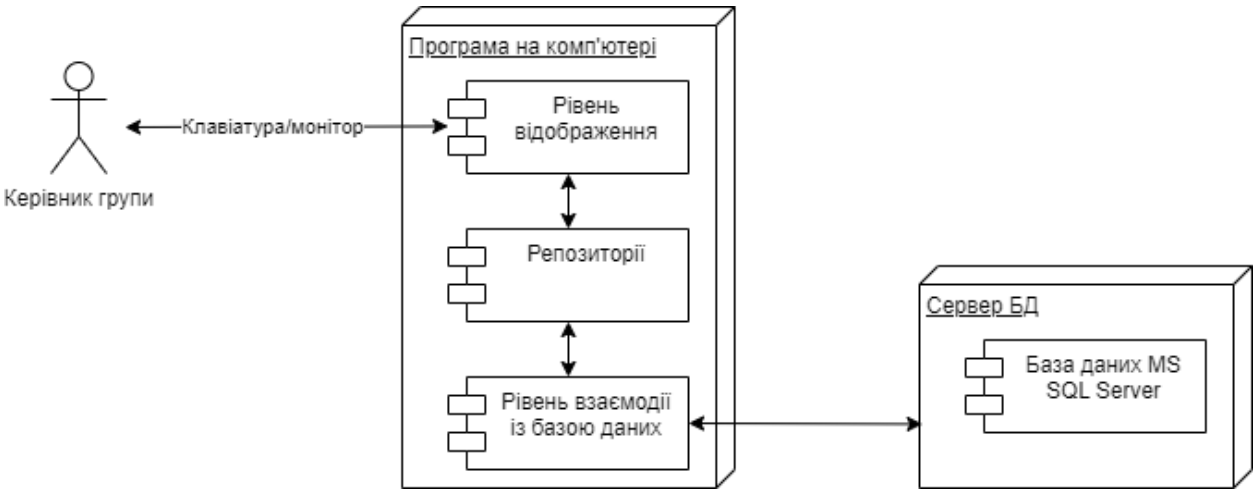
ДОДАТОК Г. ER ДІАГРАМА



ДОДАТОК Д. UML ДІАГРАМА КЛАСІВ (ОБ'ЄКТІВ)



ДОДАТОК Е. UML ДІАГРАМА РОЗГОРТАННЯ



ДОДАТОК Є. ВИХІДНІ КОДИ

Оскільки код програми досить об’ємний, нижче наведено приклад найважливіших частин програми, решта ж розміщена в репозиторії за посиланням: https://github.com/angelina-babych/Kursova-2-kHAI

```
using BusinessLogic.Repositories;
using Caliburn.Micro;
using GroupManager.Core.Model;
using GroupManager.Models;
using GroupManager.Views;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace GroupManager.ViewModels
{
    public class MainViewModel:Screen
    {
        IRepository<Group> _groupRepository;
        public UserControl CalendarItem { get; set; }
        BindableCollection<Group> _groups { get; set; }
        public BindableCollection<Group> Groups
        {
            get => _groups;
            set
            {
                _groups= value;
                NotifyOfPropertyChange(() => Groups);
            }
        }
        string groupName;
        public string GroupName
        {
            set
            {
                groupName= value;
                NotifyOfPropertyChange(() => GroupName);
            }
            get
            {
                return groupName;
            }
        }
        public string Date { get; set; }
        Group selectedGroup;
        public Group SelectedGroup {
            get=>selectedGroup;
            set {
                selectedGroup = value;
                NotifyOfPropertyChange(() => SelectedGroup);
```

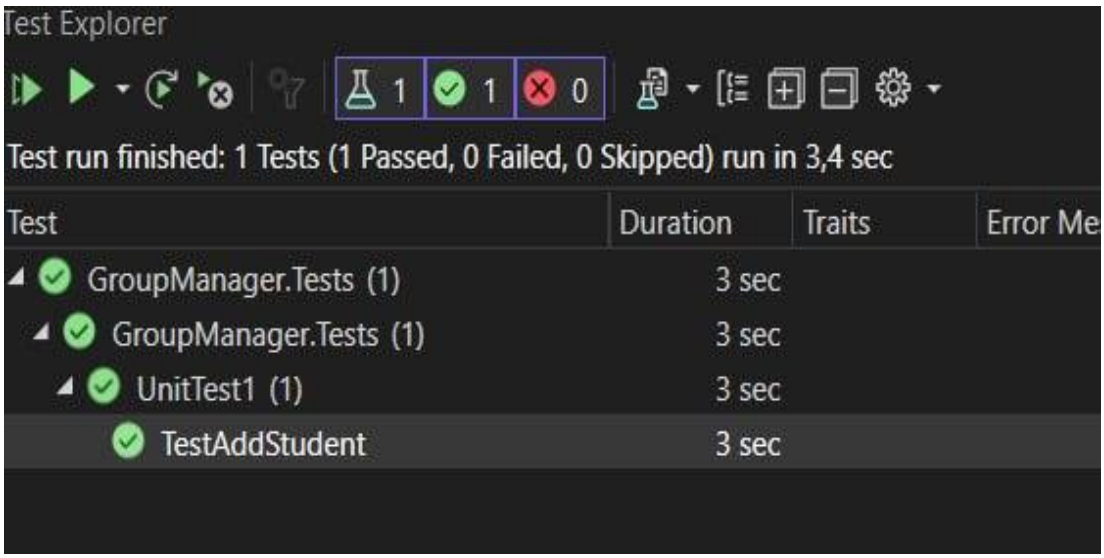
```
    }
}
public MainViewModel(
    IRepository<Group> repository)
{
    string strDate = DateTime.UtcNow.ToString("dddd, dd MMMM");
    Date = char.ToUpper(strDate[0]) + strDate.Substring(1);
    _groupRepository = repository;
    CalendarItem = new CustomControls.Calendar();
    UploadGroups();
}
private async void UploadGroups()
{
    var reverseList = (await _groupRepository.GetAllAsync())
        .ToArray()
        .Reverse();
    Groups=new BindableCollection<Group>(reverseList);
}

public async void AddGroup()
{
    if (SelectedGroup == null)
    {
        Group newGroup = new Group
        {
            Name = GroupName,
            Id = Guid.NewGuid(),
        };
        _groupRepository.Add(newGroup);
        GroupName = "";
        Groups.Clear();
        var groupsList = (await _groupRepository.GetAllAsync())
            .ToArray()
            .Reverse();
        Groups.AddRange(groupsList);
    }
    else
    {
        SelectedGroup.Name=GroupName;
        _groupRepository.Update(SelectedGroup);
        Groups.Clear();
        var reverseList = (await _groupRepository.GetAllAsync())
            .ToArray()
            .Reverse();
        Groups.AddRange(reverseList);
    }
}
public async void RemoveGroup()
{
    if (SelectedGroup != null)
    {
        await _groupRepository.RemoveAsync(SelectedGroup);
        Groups.Clear();
        var groupsList = (await _groupRepository.GetAllAsync())
            .ToArray()
            .Reverse();
        Groups.AddRange(groupsList);
    }
}
protected override void OnViewReady(object view)
{
```

```
            base.OnViewReady(view);
            //MainView shellView = (MainView)view;
            //shellView.CommandBindings.Add(new
CommandBinding(ApplicationCommands.Delete,RemoveGroup));
        }
        public void GroupClicked()
        {
            if (SelectedGroup != null)
            {
                var studentsList = IoC.Get<StudentsListViewModel>();
                studentsList.CurrentGroup = SelectedGroup;
                Switcher.SwitchAsync(studentsList, new
System.Threading.CancellationToken());
            }
        }
        protected override void OnViewLoaded(object view)
        {
            base.OnViewLoaded(view);

        }
        public void ExitProgram()
        {
            System.Windows.Application.Current.Shutdown();
        }
        public void AboutProgram()
        {
            var viewModel = IoC.Get<AboutViewModel>();
            Switcher.SwitchAsync(viewModel, new
System.Threading.CancellationToken());
        }
    }
}
```

ДОДАТОК Ж. РЕЗУЛЬТАТИ МОДУЛЬНОГО ТЕСТУВАННЯ



ДОДАТОК З. ЗНІМКИ ЕКРАНУ

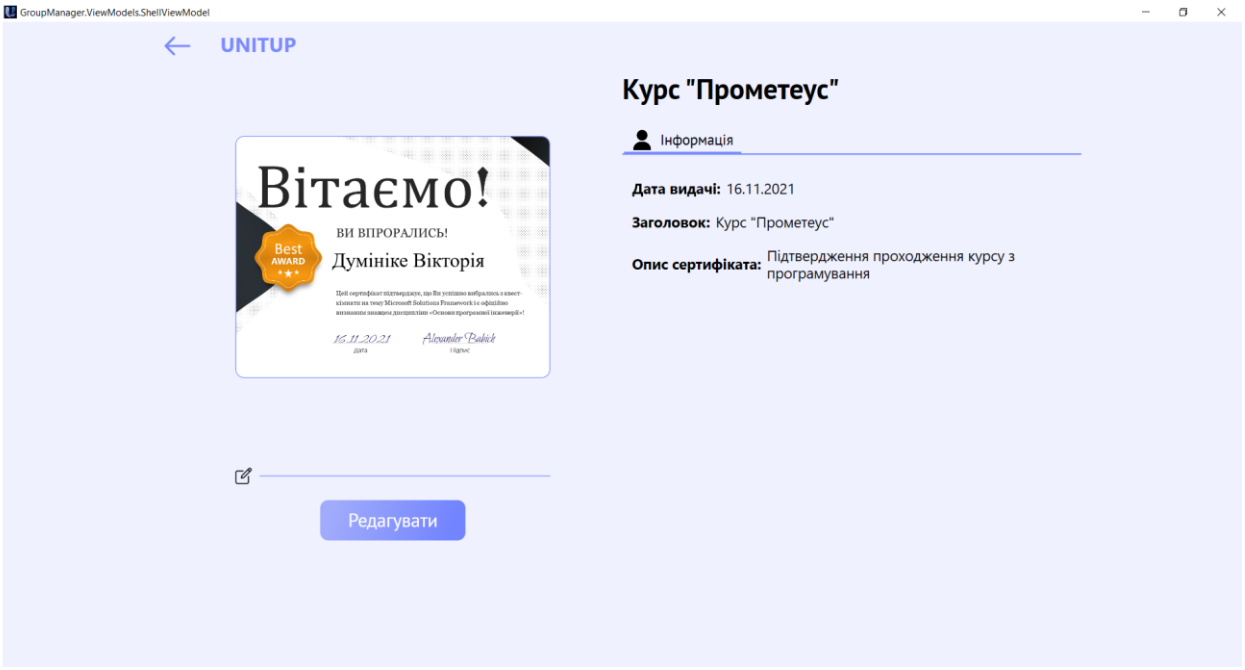


Рисунок 3.1 – Інформація про сертифікат

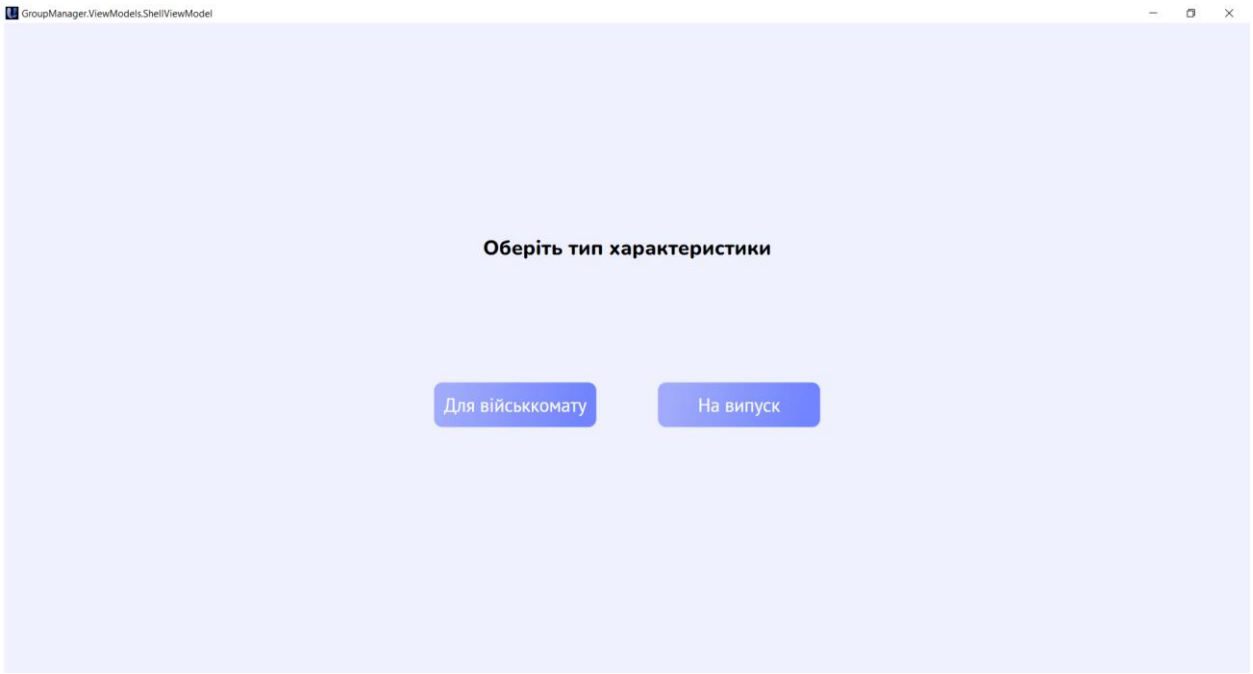


Рисунок 3.2 – Вікно вибору характеристики

Рисунок 3.4 – Характеристика на випуск

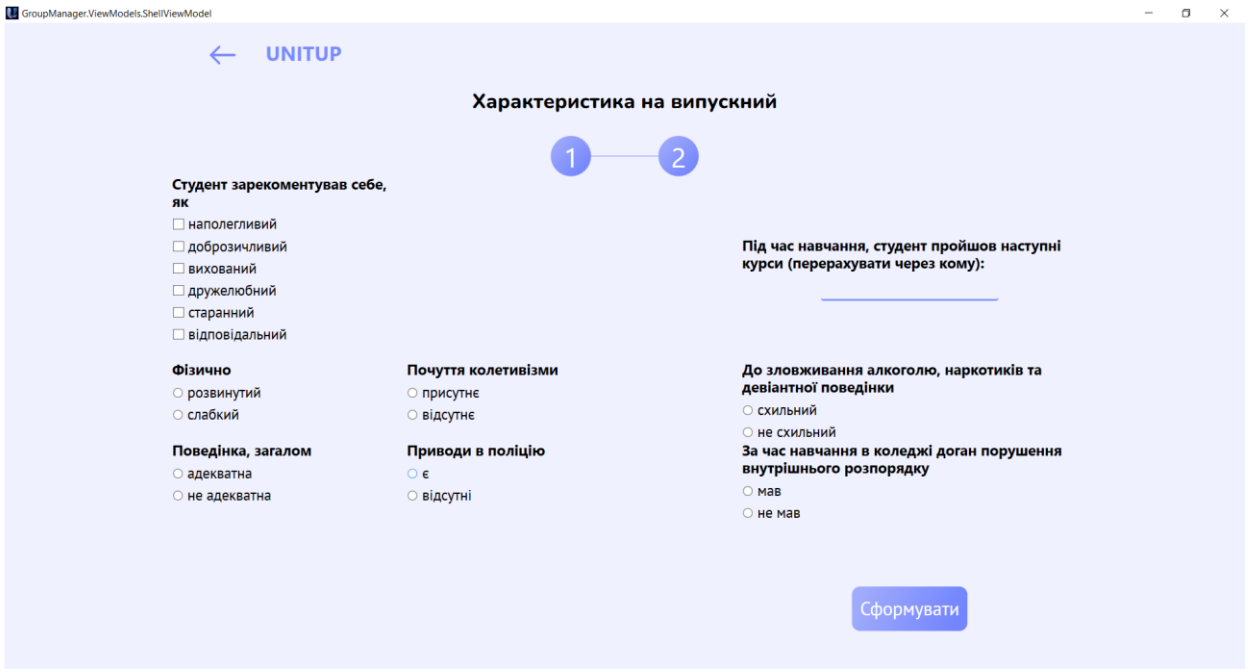


Рисунок 3.4 – Характеристика на випуск