МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Інформаційна система обліку науково-дослідницької роботи студентів

Виконав студент групи К-41
спеціальності 122 «Комп'ютерні науки»
Аліфлі Ельдагіз Мадад оглу

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант

Рецензент к.техн.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2021

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АСУ – автоматизована система управління
БД – база даних
ВЗО – вищий навчальний заклад освіти
ІС – інформаційна система
НДР – науково-дослідна робота
НДРС – науково-дослідна робота студента
ОДЕКУ – Одеський державний екологічний університет
ОС – операційна система
ПЗ – програмне забезпечення
СКБД – система керування базами даних
DFD – Data Flow Diagrams – диаграми потоків даних
JVM – Java Virtual Machine – віртуальна машина Java
UI – User Interface – інтерфейс користувача
UML – Unified Modeling Language – уніфікована мова моделювання

ВСТУП

В останні роки у вищій освіті впроваджується система рейтингування студентів і викладачів. Обробка персональних даних і показників успішності з разних напрямках діяльності потребує використання спеціального автоматизованого програмного забезпечення. Подібна автоматизована система повина мати зручний інтерфейс і дозволяти користувачу розрахувати загальний рейтинг студента на основі його успішності та показників його участі у науково-дослідній роботі.

Обсяг науково-дослідних робіт, число конференцій збільшується постійно. Тому просто необхідно впроваджувати автоматизовану систему обліку науково-дослідних робіт студентів. Принцип системи автоматизованого обліку полягає в наступному. Відповідальна за облік науково-дослідницьких робіт особа вводить в систему нові записи про досягнення студентів: участь у наукових кружках та семінарах, конференціяїх різного рівня, конкурсі студентських наукових робіт, олімпіадах, грантах тощо. Таким же чином вводяться дані викладачів, які виконували керівництво науково-дослідною роботою студентів (НДРС); студентів, яким можуть призначатися науково-дослідні завдання, нові групи учнів, нові майбутні конференції. За результатами виконаних науково-дослідних робіт розраховуються кредити студентам за вказані періоди часу. За існуючими даними при заданому критерії може бути здійснено пошук, результати пошуку можуть бути збережені в звіт і згодом при необхідності роздруковані. Для внесення змін система потребує наявності зручного інтерфейсу, який дозволяє переглядати таблиці бази даних і редагувати їх.

Метою кваліфікаційної роботи бакалавра є створення інформаційної системи обліку науково-дослідницької роботи студентів. Ця система повинна забезпечувати можливість зручного додавання науково-дослідних робіт, пошуку робіт за заданими критеріями, складання звіту за результатами пошуку у табличному вигляді, а також перегляд і редагування таблиць бази даних.

Для досягнення поставленої мети були сформульовані наступні завдання:

- провести опис та аналіз предметної області;
- провести порівняльний аналіз існуючих програм аналогів;
- обґрунтувати вибір програмних засобів розробки та технологій для вирішення поставленої проблеми;
- провести проектування програмної системи з використанням мови UML;
- розробити структуру бази даних;
- виконати проектування інтерфейсу користувача;
- реалізувати підсистему обліку науково-дослідної роботи студентів ОДЕКУ;
- виконати тестування та налагодження системи.

Структура кваліфікаційної роботи бакалавра складається з вступу, трьох розділів, висновків, переліку посилань на 24 найменування, додатків. Повний обсяг роботи становить 64 сторінок, містить 20 рисунків і 6 таблиць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОГРАМНИХ СИСТЕМ АНАЛОГІВ

1.1 Збір інформації та підготовка вихідних даних

Вихідними даними в системі, що розробляється є дані, які відповідають критеріям пошуку, інформація про тип наукової роботи, групу, персональні дані студента, дані викладача і файл в який записується звіт у вигляді таблиці. До структури підсистеми з обліку науково-дослідницької роботи входить база даних. Нижче наводяться типи наукових робіт студентів, які підлягають врахуванню і кількість кредитів по кожному типу робіт [1]:

- виступи з доповідями на наукових гуртках та семінарах;
- участь в олімпіадах та конкурсах наукових робіт;
- участь з доповідями в університетських, всеукраїнських та міжнародних конференцях;
- підготовка наукових публікацій;
- участь у виконанні науково-дослідних робіт (НДР) кафедр, проблемних лабораторій науково-дослідної частини;
- участь в роботі студентських наукових шкіл, стажуванні за програмами міжнародного співробітництва.

Наукова та науково-технічна діяльність студента оцінюється щосеместрово через нарахуванн наукових кредитів в залежності від успіхів студента та виду діяльності, що була здійснена студентом, згідно з табл. А1, наведеною у додатку А.

1.2 Мета та задачі інформаційної системи

Мета ІС: автоматизувати роботу, пов’язану з обліком та нарахуванням кредитів студентам ОДЕКУ за виконані види наукової та науково-технічної роботи. Інтерфейс підсистеми має бути зручним для кожного користувача та зберігати дані про досягнення студів в БД.

Цільова аудиторія: відповідальні за науково-дослідницьку роботу студентів на кафедрах та відповідних структурних підрозділах університету.

Підсистема повинна виконувати наступні функції:

– дані, що вносяться користувачем повинні автоматично додаватися до БД;

– повинна бути можливість додавати до БД нові записи, видаляти застарілі або непотрібні записи, редагувати існуючі;

– виконувати необхідні запити на отримання потрібних даних;

– програма повинна мати зручний і досить простий інтерфейс, який буде зрозумілий і не кваліфікованому користувачеві;

– БД повинна зберігати інформацію про студентів на протязі їх навчального часу, потім відправляти у архів;

– якщо студент відраховується, він потрапляє у архів;

– підсистема повинна виконувати разрахунок кредитів студентів за вказаний семестр та виводити звіти у зручному форматі на друк.

Інформаційні потоки системи зображено на рисунку 2.1.

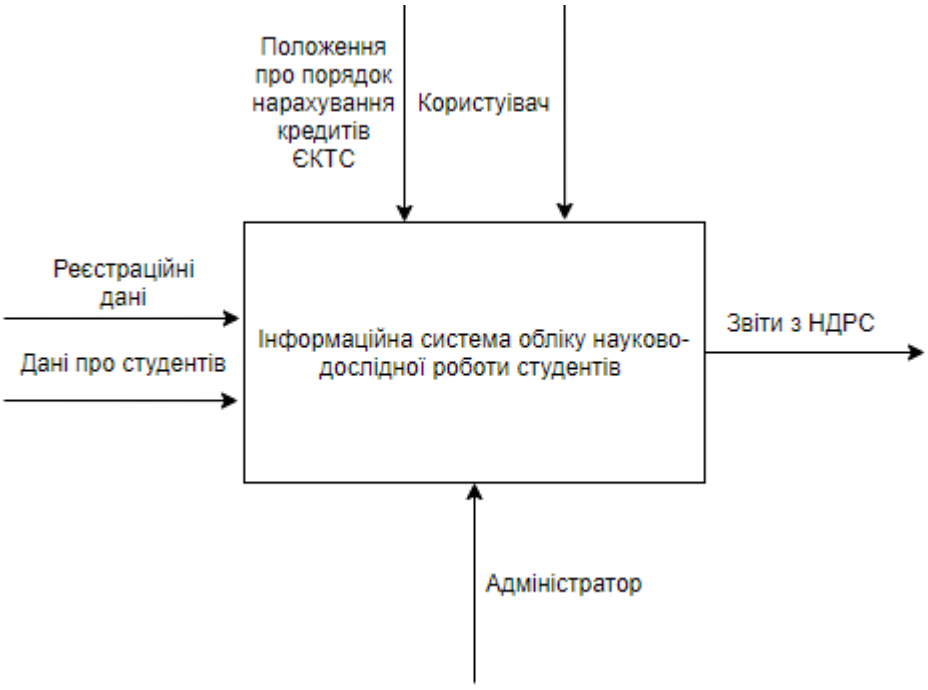

Рисунок 1.1 – Діаграма інформаційних потоків

1.3 Аналіз програмних систем аналогів

В даний час існує не так багато відомих автоматизованих систем управління ВЗО. Більшість з них не має специфічних модулей обліку саме науково-дослідної роботи студентів. В деяких системах наявні модулі наукової роботи, але вони стосуються саме роботи викладачів.

АСУ УЗ [2] – модульна система управління управління ВЗО від НПП «МКР» м. Харьків (рис.1.1).

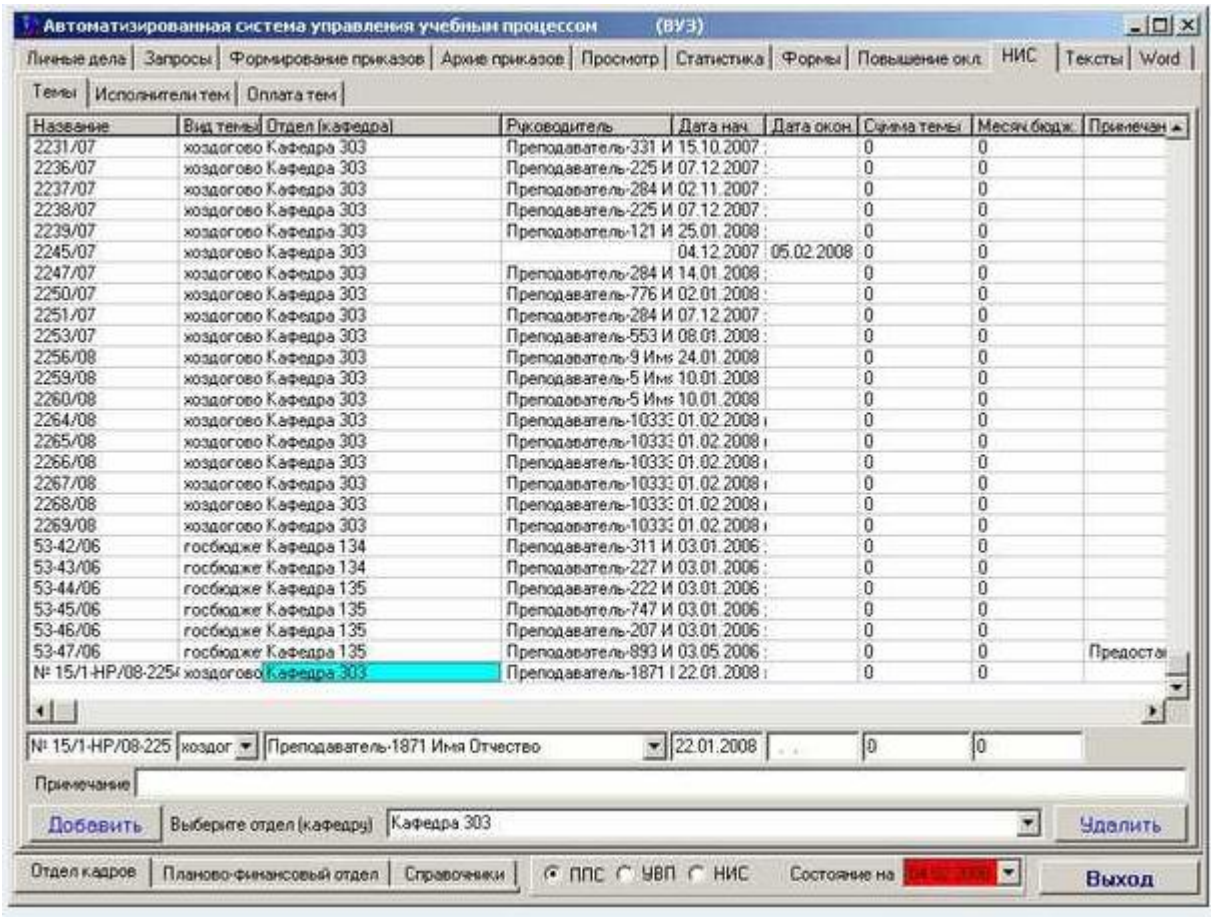

Рисунок 1.1 – Інтерфейс системи АСУ УЗ

Система є популярною і використовується в різних ЗВО України. Вона має окремий модуль «Деканат», однією з функцією якого є розрахунок рейтингу студентів по групах або по потоках. Нажаль, рейтинг розраховується виключно за нарахованими балами, отриманими під час навчального проце-

су. Також у системи є можливість вести облік науково-дослідної роботи, але лише співробітників університету. Основні її функції [2]:

- ведення довідника всіх державних і господарських тем;
- ведення інформації про відповідальних і всіх виконавців різних видів тем (наукових робіт);
- розрахунок і облік оплат виконання робіт по темам.

Як можно побачити, система АСУ УЗ не має готового рішення для обліку НДРС, яке би враховувало діючи в університеті нормативні документи та положення.

Серед відомих систем управління навчальним процесом ВЗО на ринку України можна відзначити ще автоматизовану систему управління навчальним процесом для вищих навчальних закладів усіх рівнів акредитації АСК “ВНЗ”, розроблена у Науководослідницькому інституті прикладних інформаційних технологій, яка є частиною інформаційно-виробничої системи “Освіта” [3,4].

Поряд з цим у багатьох великих ВНЗ функціонують і власні розробки подібних систем. До них можна віднести:

– електронну систему управління ВНЗ "Сократ" Вінницького національного аграрного університету [5];

– засоби автоматизації управління навчальним закладом, що діють в НУ “Львівська політехніка” та Львівського національного університету імені Івана Франка;

– автоматизовану інформаційну систему “Електронний університет” [6], створену у Хмельницькому національному університеті.

Усі наведені в цьому розділі АСУ є комерційними і не пропонують потрібної для ОДЕКУ функціональності. Функції систем є чи занадто загальними і потребують доопрацювання, чи є специфічними пристосованими до потреб конкретних ВЗО.

Тому цей факт підтверджує необхідність створення власної системи управління навчальним процесом в ОДЕКУ і безпосередньо модулю обліку

науково-дослідної роботи студентів, який би враховував всі чинні в університеті документи.

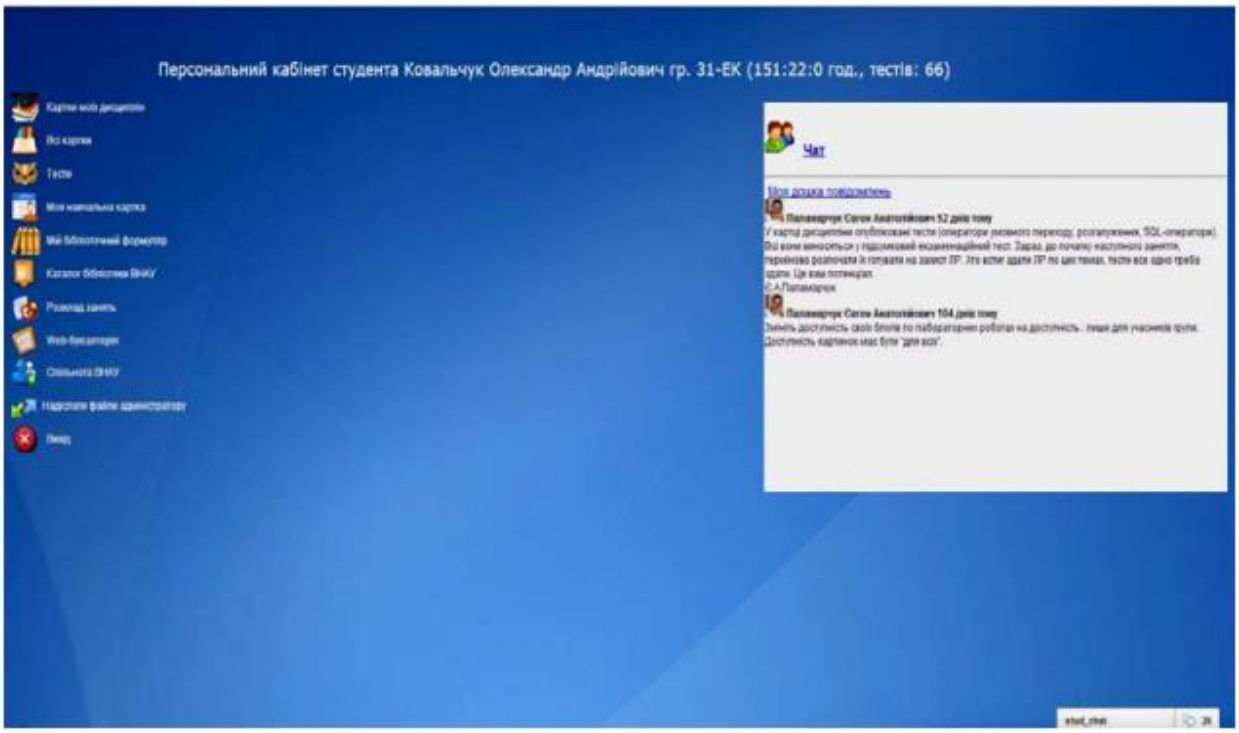

Рисунок 1.2 – Вигляд персонального кабінету студента в АСУ «Сократ»

Проведений аналіз існуючих програм аналогів виявив, що кількість їх обмежена, доступ до цих систем є платним, а функціональні можливості забезпечують розрахунок різних видів наукових робіт відповідно до внутрішніх нормативних документів, що діють у певних ВЗО, де функціонує та чи інша система.

Тому актуальним є створення власної програмної розробки, яка буде безкоштовною для університету, а також буде враховувати всі діючі положення про нарахування кредитів НДРС в ОДЕКУ, з можливістю змін налаштувань системи за потребою.

Висновки до розділу: під час виконання першого розділу кваліфікаційної роботи проведено аналіз предметної області, в рамках якої розглянуті аналоги і їх особливості. Переваги та недоліки розглянутих систем управлін-

ня навчальним процесом були взяті до уваги при розробці програмного продукту. Показана необхідність створення власної системи управління навчальним процесом в ОДЕКУ і безпосередньо модулю обліку науково-дослідної роботи студентів, який би враховував всі чинні в університеті документи.

2 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Вибір мови програмування

Наведемо порівняльний аналіз сучасних мов програмування та обгрунтуємо вибір мови для написання програми.

Згідно рейтингу IEEE Spectrum за 2020 рік [7] серед найбільш популярних мов програмування можна виділити мови python, java, C, C++, C#. Тому розглянемо їх переваги та недоліки більш докладно.

Рисунок 2.1 – Рейтинг мов програмування за версією IEEE Spectrum

Приблизно цей же перелік мов програмування і у рейтингу TIOBE [8], що розраховується виходячи з кількості результів запитів до пошукових систем, що містять назву мови [9].

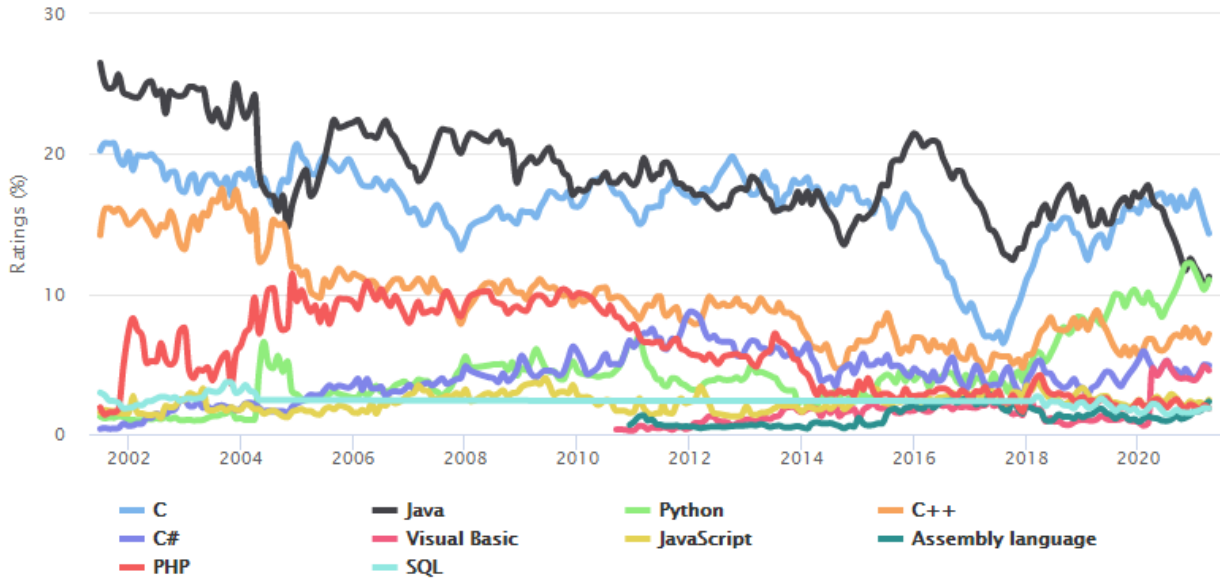

Рисунок 2.2 – Рейтинг TIOBE Programming Index

Основні переваги мови Java [10]:

- крос-платформність;
- строго типізована мова;
- вихідні тексти середовища виконання відкриті, і підтримуються великою кількістю розробників по всьому світу;
- широкий вибір фреймворків і бібліотек для побудови корпоративних застосунків;
- найбільш популярні рішення для створення веб-застосунків мають відкритий вихідний код і поширюються під вільною ліцензією. До таких відносяться наприклад: Spring Framework, Apache Karaf, GWT.

Недоліки:

- низька, в порівнянні з іншими мовами, швидкодія, підвищені вимоги до обсягу оперативної пам'яті (ОП);
- великий обсяг стандартних бібліотек і технологій створює складності у вивченні мови;
- постійний розвиток мови викликає наявність як застарілих, так і нових засобів, що мають одне і те ж функціональне призначення.

Основні переваги C#:

- підтримка Microsoft, на відміну від Java, якої не пішов на користь перехід у власність Oracle, C # добре розвивається завдяки зусиллям Microsoft;
- останнім часом багато вдосконалюється, так як C# був створений пізніше, ніж Java та іншими мовами, то потрібно дуже багато доопрацювати;
- багато синтаксичного цукру. Синтаксичний цукор – конструкції мови програмування, які створені для полегшення написання і розуміння коду і не грають ролі при компіляції;
- середній поріг входження: синтаксис схожий на C, C ++ або Java полегшує перехід для інших програмістів;
- наявність додаткового функціонального мови програмування (F#).

Недоліки:

- орієнтованість, в основному, тільки на .NET (на Windows платформу);
- безкоштовна ліцензія тільки для невеликих компанії, учнів і програмістів, які працюють над власними проектами в поодинці, для великих проектів покупка ліцензій зажадає значних матеріальних вкладень.

Основні переваги Python [11]:

- великі бібліотеки підтримки;
- наявність вбудованих інструментів для інтеграції Enterprise Application Integration, яка полегшує розробку веб-сервісів, викликаючи компоненти COM або COBRA;
- покращена продуктивність програміста, завдяки потужним функціям інтеграції процесів, модульному тестування і розширеним можливостям управління, що сприяє збільшенню швидкості таі продуктивності застосунків.

Недоліки:

- Python виконується за допомогою інтерпретатора замість компілятора, що призводить до його уповільнення, тому що компіляція та виконання допомагають йому працювати нормально;
- у порівнянні з популярними технологіями, такими як JDBC і ODBC, рівень доступу до бази даних в Python виявився трохи недорозвиненим і примітивним.

Таким чином, провівши аналіз мов програмування, можна вважати, що вибір Java є обгрунтованим, в зв'язку з тим, що дана мова має такі важливі переваги:

- значна підтримка з боку розробників, які не прив'язані до якоїсь конкретної корпорації, як наприклад C# прив'язаний до Microsoft;
- тексти програм середовища виконання Java відкриті, що дозволяє не тільки детально розібратися з архітектурою мови, а й вносити доопрацювання для інтеграції з новими можливостями мови;
- тісна інтеграція з системами зберігання, не тільки файловими, але і СКБД; на відміну від мови Python, Java має бібліотеки JDBC (Java Data Base Connectivity) для всіх популярних реляційних баз даних;
- здатність працювати на всіх популярних платформах під керуванням таких операційних систем як Windows, Linux, MacOS, Solaris та інші. Це важливий аспект, так як продуктивне середовище може бути розгорнуте на ОС Linux, в той час як розробники для себе можуть вибрати будь-яку зручну ОС з графічною оболонкою, це і Windows 10, і Linux Ubuntu, Linux Kubunu, Linux Mint, MacOS і багато інших.

2.2 Вибір та обґрунтування середовища розробки

Серед найбільш популярних середовищ розробки на мові Java можна виділити три:

- IntelliJ IDEA;
- Eclipse;

- NetBeans.

Переваги IDE Intellij IDEA [12]:

- крім платній версії (Ultimate edition) з розширеними можливостями, має безкоштовну версію (Community edition);
- підтримує не тільки останні версії Java, а й Kotlin, Groovy і Scala;
- працює з такими системами контролю версій як Git, SVN, Mercurial і CVS;
- має підтримку популярних фреймворків Java EE, Spring, GWT, Vaadin, Play, Grails;
- має вбудований інструментарій для підключення до СКБД використовуючи драйвери JDBC; також є можливість виконувати SQL запити, переглядати / додавати / змінювати / видаляти дані в таблицях;
- крім підсвічування синтаксису і автопідстановки має багатий інструментарій по рефакторингу коду, аналізу коду;
- тісно інтегрується з такими системами збірки як Ant, Maven і Gradle, дозволяє запускати програму прямо з вихідного коду, не звертаючись до терміналу;
- наявність розпізнавання контексту в вихідному коді, наприклад, при написанні в файлі з розширенням java, впроваджений фрагмент SQL запиту, IDE розпізнає і дозволяє не тільки в ньому підсвітити ключові слова та виконати автопідстановку при редагуванні, але і запустити його прямо звідти, де він знаходиться, використовуючи контекстне меню [13].

Основним недоліком даного середовища розробки можна вважати ціну, яку доведеться заплатити за розширений функціонал Ultimate edition.

Переваги Eclipse IDE [14]:

- IDE побудована на базі специфікації динамічної модульної системи, іншими словами, середовище розробки вдає із себе модульну платформу, до якої приєднуються плагіни для розробки на різних мовах, в тому числі Java, а також різні інструменти інтеграції;

- Eclipse IDE повністю безкоштовна;
- можливість встановити найрізноманітніші плагіни: для інтеграції з системами контролю версій, системами збирання проекту, побудова звітів, інструменти для моделювання бізнес-процесів тощо.;
- середовище розробки написано на Java, тому є кросс-платформним;
- можливість налаштовувати відображення панелей в залежності від контексту роботи.

Недоліки:

- велика кількість плагінів значно уповільнюють роботу середовища;
- плагіни часто пишуться сторонніми розробниками, тому встановлюючи додаткові модулі, нерідко можна отримати їх конфлікт;
- модулі часто нестабільні або мають невелику кількістю функціональності.

Переваги NetBeans [15]:

- включає в себе інструментарій для роботи з Java, JavaScript і HTML, а також підтримку інтеграції з такими веб-серверами як GlassFish і Tomcat;
- редактор дозволяє відстежувати помилки «на льоту» при введенні тексту;
- швидкість роботи не залежить від кількості встановлених плагінів, вони не конфліктують один з одним;
- в інструментарій NetBeans входять графічні редактори для побудови призначеного для користувача інтерфейсу.

З недоліків можна відзначити бідний набір інструментів щодо інтеграції з СКБД, системами контролю версій, системами збірки, в порівнянні з Inellij IDEA і в Eclipse IDE.

Як підсумок, можна сказати, що найбільш доцільно використовувати IntelliJ IDEA Community Edition в зв'язку з наявністю у неї досить різноманітного інструментарію для роботи з системами управління версіями і системами збірки, можливості працювати з базою даних безпосередньо, не

встановлюючи додаткового ПЗ. Одним з аргументів також слід зазначити і продуктивність. На відміну від Eclipse IDE, система IntelliJ IDEA вже включає необхідний інструментарій, який не конфліктує між собою і не уповільнює роботу програми.

Інтерфейс IDE містить такі основні елементи як: вікно текстового редактора, проекти, переходи і завдання (рис.2.3).

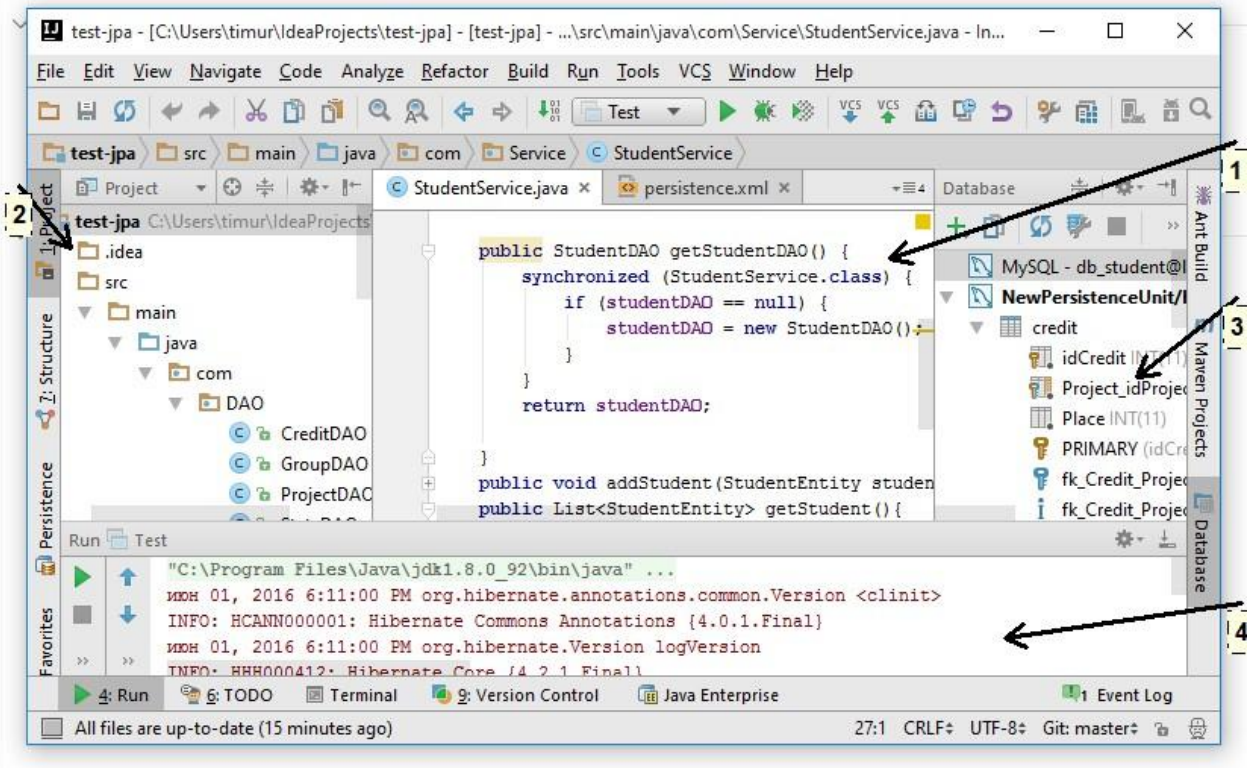

Рисунок 2.3 – Інтерфейс IDE

1) Вікно текстового редактора. Написання та редагування програмного коду. Є можливість переключатись між файлами і перехід на стартову сторінку. Також тут можна подивитися історію написання та редагування коду, треба відзначити дуже зручно реалізовано, показуються ділянки коду до і після певної зміни.

2) Вікно проекту. Область де можна дивитися структуру проекту. Перегляд вихідних файлів програми, бібліотеки, додаткові файли конфігурацій і так далі.

3) Вікно для роботи з базою даних.

4) Вторинні вікна, які забезпечують доступ до різних конкретних завдань (управління проектами, пошук, запуску та налагодження, інтеграції з системами контролю версій і т.д.

Написання коду в редакторі відбувається дуже зручно. Прекрасно реалізовано автодоповнення, де є пояснення елементів, які доповнюються, в поясненнях можна переходити на опис інших елементів, також там є посилання на повний опис в інтернеті. При виникнення помилки пропонуються варіанти виправлення, доповнення, зміни та їх автоматична реалізація (наприклад імпорт відсутнього класу).

Середа розробки має повну документацією, є об'ємний опис всіх основних елементів. Є велика кількість навчального матеріалу.

2.3 Обґрунтування та вибір системи керування базами даних

Незважаючи на те, що всі системи керування базами даних (СКБД) виконують одну і ту ж основну задачу (тобто дають можливість користувачам створювати, редагувати і отримувати доступ до інформації, що зберігається в базах даних), сам процес виконання цього завдання варіюється в широких межах. Крім того, функції і можливості кожної СКБД можуть істотно відрізнятися. Різні СКБД документовані по-різному: більш-менш ретельно. По-різному надається і технічна підтримка.

При порівнянні різних популярних систем, слід враховувати, чи зручна для користувача і масштабуєма дана конкретна СКБД, а також переконатися, що вона буде добре інтегруватися з іншими продуктами, які вже використовуються. Крім того, під час вибору слід взяти до уваги вартість системи і підтримки, наданої розробником.

Існує кілька популярних СКБД, як платних, так і безкоштовних, які можна рекомендувати для застосування в організації.

Для порівняння візьмемо найбільш популярні СКБД:

- MySQL;
- Microsoft SQL Server;
- PostgreSQL.

Переваги MySQL [16]:

- розповсюджується безкоштовно;
- прекрасно документована;
- пропонує багато функцій, навіть у безкоштовній версії;
- MySQL включена в стандартні репозиторії найбільш поширених дистрибутивів операційної системи Linux, що дозволяє встановлювати її елементарно;
- підтримує набір призначених для користувача інтерфейсів;
- може працювати з іншими базами даних, включаючи DB2 і Oracle.

Недоліки:

- доведеться витратити багато часу і зусиль, щоб змусити MySQL виконувати нескладні завдання, хоча інші системи роблять це автоматично, наприклад: створювати інкрементні резервні копії;
- відсутня вбудована підтримка XML або OLAP;
- для безкоштовної версії доступна тільки платна підтримка.

Ідеально підходить для: організацій, яким потрібен надійний інструмент управління базами даних, але безкоштовний.

Переваги Microsoft SQL сервер [17]:

- продукт дуже простий у використанні;
- поточна версія працює швидко і стабільно;
- движок надає можливість регулювати і відслідковувати рівні продуктивності, які допомагають знизити використання ресурсів.
- можна отримати доступ до візуалізації на мобільних пристроях;
- добре взаємодіє з іншими продуктами Microsoft.

Недоліки:

- ціна для юридичних осіб виявляється неприйнятною для більшої
- частини організацій;
- навіть при ретельному налаштування продуктивності SQL Server здатний зайняти всі доступні ресурси;
- повідомляється про проблеми з використанням служби інтеграції для
- імпорту файлів.

Ідеально підходить для: великих організацій, які вже використовують ряд продуктів Microsoft.

Переваги PostgreSQL [18]:

- масштабується і здатний обробляти терабайти даних;
- підтримує формат json;
- існує безліч визначених функцій;
- доступна низка інтерфейсів.

Недоліки:

- документація туманна, тому, можливо, відповіді на деякі питання доведеться шукати в інтернеті;
- конфігурація може збентежити непідготовленого користувача;
- швидкість роботи може падати під час проведення пакетних операцій або виконання запитів читання.

Ідеально підходить для організацій з обмеженим бюджетом, але кваліфікованими фахівцями, коли потрібно можливість вибрати свій інтерфейс і використовувати json [19].

Підбивши підсумок, можна зробити висновок, що оптимальною СКБД для зберігання даних в системі обліку НДРС є MySQL.

MySQL – вільна система керування базами даних. Поширюється під GNU General Public License і під власною комерційною ліцензією, на вибір. Крім цього розробники створюють функціональність за замовленням ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації. Ліцензія на MySQL Community edition

надається безкоштовно.

2.4 Вибір інструментів для проектування

Перед створенням програмного коду мобільного застосунка необхідно виконати проектування майбутнього програмного забезпечення. Це завдання в свою чергу потребує обґрунтування вибору інструментів для проектування.

Розглянемо найбільш популярні засоби для проектування програмного забезпечення. До них відносяться:

- BPMN Модель Бізнес-Процесів;
- побудова блок-схем;
- створення ER-діаграм;
- UML-діаграми.

Модель BPMN є стандартом для моделювання бізнес-процесів, що надає графічну нотацію для визначення складної семантики бізнес-процесу у вигляді діаграми BPD. Така діаграма ґрунтується на представлені бізнес-процесу у вигляді блок-схеми, що семантично схожа на діаграму діяльності [20]. Розрізняють чотири категорії елементів діаграми BPD:

- об’єкти потоку керування: дії, події та логічні оператори;
- поєднуючі елементи: потік керування, потік повідомлень та асоціації;
- ролі: пули та доріжки;
- артефакти: дані, групи та текстові анотації.

Приклад моделі бізнес-процесу та позначення процесу із нормальним потоком наведено на рис.2.4.

Блок-схема (англ. flowchart) дозволяє представити алгоритм розв'язування або аналізу задачі за допомогою геометричних елементів (блоків), які позначають операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму. Схема машини, приладу, апарата, пристрою, в якій основні вузли

(блоки), що утворюють її, зображено прямокутниками та іншими фігурами, а зв'язок між ними показано лініями зі стрілками.

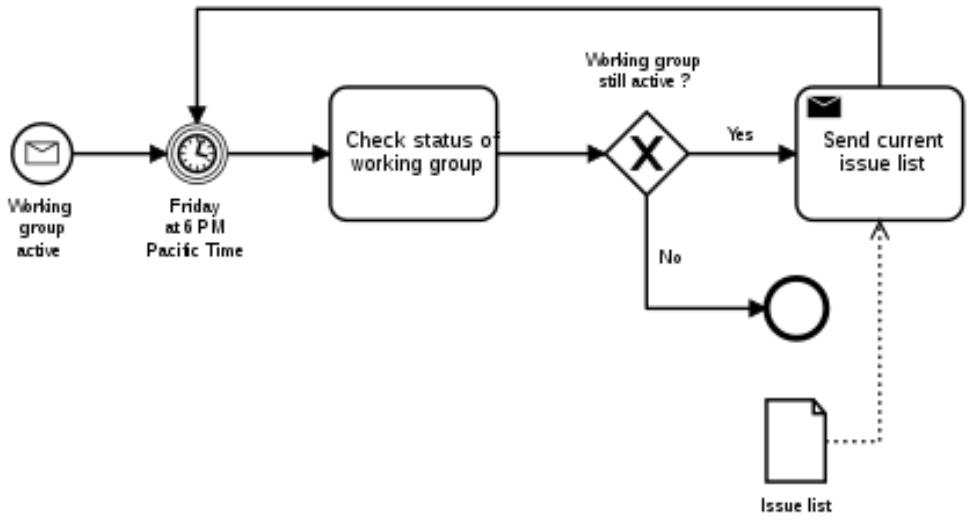

Рисунок 2.4 – Приклад діаграми моделі бізнес-процесу

Схема «сутність-зв'язок» (також ERD або ER-діаграма) – це різновид блок-схеми, де показано, як різні «сутності» пов'язані між собою всередині системи [21]. ER-діаграми найчастіше застосовуються для проектування і налагодження реляційних баз даних. ER-діаграми (або ER-моделі) використовують стандартний набір символів, включаючи прямокутники, ромби, овали і сполучні лінії, для відображення сутностей, їх атрибутів і зв'язків. ER-діаграми часто використовуються в поєднанні з діаграмами DFD, які схематично показують рух потоків інформації в рамках процесу або системи (рис.2.5).

У ER-моделях і моделях даних зазвичай виділяють до трьох рівнів деталізації:

- концептуальна модель даних – схема найвищого рівня з мінімальною кількістю подробиць, що відображаэ загальну структуру моделі і всю архітектуру системи;

- логічна модель даних містить більш детальну інформацію (операційні та транзакційні суті) та не залежить від технології, в якій вона буде застосовуватися;
- фізична модель даних: на основі кожної логічної моделі даних можна скласти одну або дві фізичних моделі, що містять досить технічних подробиць для складання і впровадження самої бази даних.

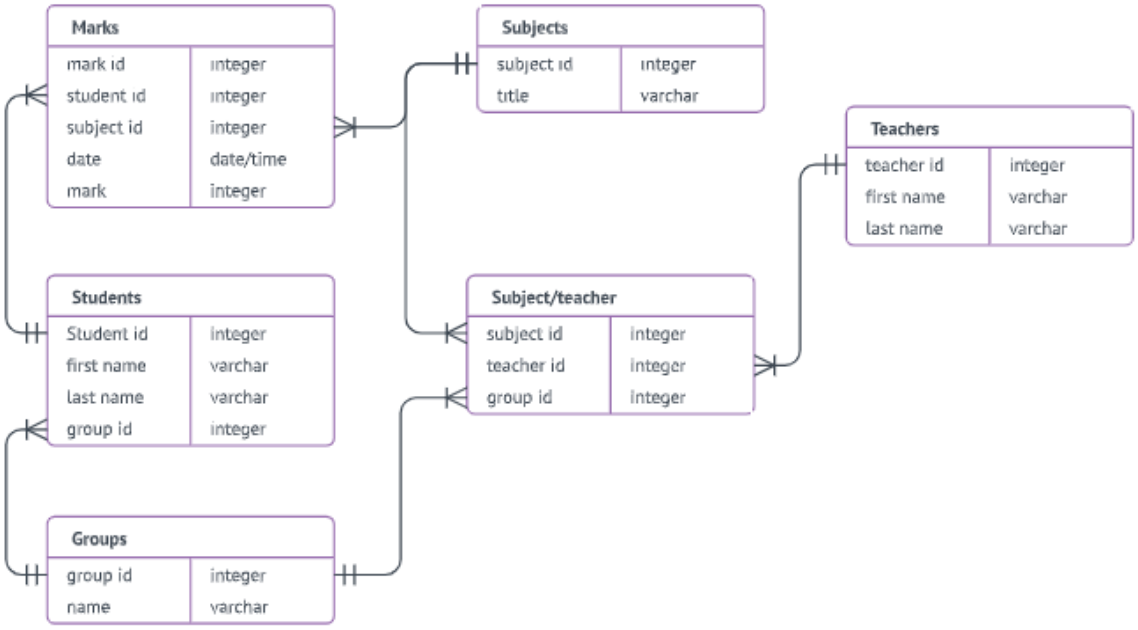

Рисунок 2.5 – Приклад ER-діаграми

ER-діаграма демонструє зв'язки і відносини між елементами, тому вона відображають тільки реляционную структуру.

UML (англ. Unified Modeling Language) – уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. Діаграми дають можливість представити систему у такому вигляді, щоб її можна було легко перевести в програмний код. Це дозволяє збільшити ефективність реалізації програмних систему кілька разів і помітно поліпшити якість кінцевого продукту. Практично усі CASE-засоби (програми автоматизації процесу аналізу і

проектування) мають підтримку UML. Серед найбільш відомих UML діаграм слід вказати наступні [22]:

- структурні діаграми: класів, компонент, розсортування, об’єктів, пакетів;
- діаграми поведінки: діяльності, станів, прецедентів;
- діаграми взаємодії: послідовності, синхронізації.

Щоб стало зрозуміліше призначення додаткових інструментів, необхідних для якісної підготовки до безпосереднього створення комп'ютерного коду майбутнього програмного продукту, перерахуємо найактуальніші програми, що дозволяють полегшити життя колективів розробників.

Найбільш часто при використанні моделі BPMN застосовуються такі пакети: Sybase Power Designer; Eclipse; Visio + BPMN; AcuaLogic BPMN.

І для створення блок-схем, що наочно формують структуру майбутнього програми або програми, а також побудови ER-діаграм використовується пакет Microsoft Visio, що можна назвати найбільш універсальним інструментом проектувальника програмного забезпечення. Для розробки ER-діаграм також застосовуються ERWin і згаданий вище пакет Sybase Power Designer.

Для побудови більш складних і дієвих UML-діаграм, які є, можливо, найефективнішим засобом для проектування програмного забезпечення, може бути використано програмний пакет StarUML. Це проект з відкритим кодом для розробки швидких, гнучких, функціональних платформ UML/MDA для 32-розрядних систем Windows. Безкоштовний платформа для моделювання StartUML є аналогом для таких комерційних проектів, як Rational Rose, Together та інших.

Таким чином, після виконаного аналізу при проектуванні програмного забезпечення будуть використані наступні засоби проектування: нотація UML для моделювання структури, поведінки та взаємодії програмної системи та ER-діаграма для опису структури БД.

Висновки до розділу: на основі проведеного аналізу в роботі було обґрунтовано вибір програмних засобів та технологій розробки застосунку об-

ліку науково-дослідної роботи студентів. Програма буде створена з використанням мови програмування Java та середовища розробки IntelliJ IDEA Community Edition. В якості СКБД буде використана MySQL. Проектування ПЗ буде здійснено з використанням засобів UML-нотацій.

ЗПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ ОБЛІКУ НДРС

3.1 Функціональні та нефункціональні вимоги

Розглянемо більш докладно функціональні вимоги.

1. Авторизація в системі

F1. Функція прямої авторизації користувача

F1.1. Користувач при авторизації повинен ввести логін і пароль. Логін не повинен повторяться, пароль повинен бути надійний. Авторизація виконується для зручності використання системи, дає можливості створювати особисті профілі користувача і зберігати результати роботи з продуктом кожного користувача.

F1.2. У разі некоректного введення логіна і пароля система виведе повідомлення про помилку і підказку. «Невірний логін / пароль, перевірте коректність введення».

F1.3. У разі успішної авторизації користувач потрапляє в свій «кабінет» (робоче вікно).

2. Робота в режимі авторизованого користувача.

F2. Додати нові групи.

F2.1. Вибір опції додавання видів робіт.

F2.2 Завантаження даних.

F2.3 Збереження результатів у особистому кабінеті.

F2.3.1 Можливість вивантажити інформацію в спеціальному файлі.

F2.3.2 При невдалому збереження користувачеві буде повідомлено про помилку.

3. Додаткові можливості в режимі авторизованого користувача.

F3. Створення груп документів.

F3.1. Виконання групування розпізнаних документів по типу, документи з певним типом знаходяться в окремій групі.

F3.2 Редагування документів.

F3.3. Виконання видалення документів або груп документів.

4. Робота в режимі адміністратора

F4. Видалення неактивних.

До нефункціональних вимог можна віднести наступні:

NF1. Застосунок має працювати з активним підключенням до інтернету.

NF 2. Застосунок має працювати на операційній системі Windows 7/8/10.

NF 3. Система повинна зберігати дані про адміністратора та користувачів в БД.

NF 4. Додаток не повинен закриватися при наявності помилок, а вміти реагувати на них і сповіщати при цьому користувача.

3.2 Діаграма прецедентів

Наведені вимоги дозволяють побудувати діаграму прецедентів, яка подана на рис. 3.1.

Діаграма прецедентів – це тип поведінкової діаграми UML, яка дозволяє візуалізувати різні типи ролей в програмній системі і те, яким чином ці ролі взаємодіють з системою [23].

Діаграма прецедентів чи варіантів використання (Use Case Diagram) визначає функціональне призначення модельованої системи або предметної області. Дана діаграма відображає безліч акторів, що взаємодіють з проектованої системою (програмним засобом) за допомогою варіантів використання. Таким чином, основними елементами діаграми варіантів використання є актор і варіант використання.

Актор – це зовнішня по відношенню до модельованої системі сутність, що взаємодіє з системою для вирішення деяких завдань. Як актор може використовуватися людина, інша система, пристрій або програмний засіб [24].

Варіант використання визначає деякий набір дій (операцій), які повинні бути виконані моделюється системою або програмним засобом при взаємодії з актором.

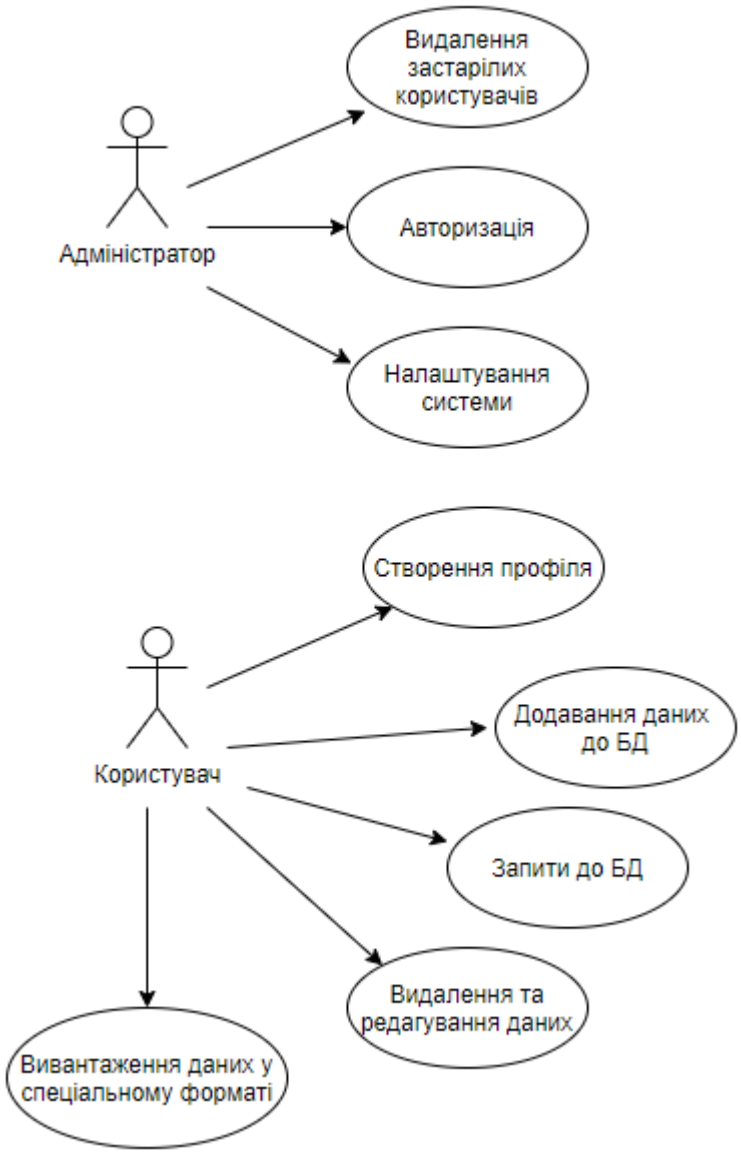

Рисунок 3.1 – Діаграма прецедентів

3.3 Логічне уявлення про систему та процеси

Для створення абстрактної уяви про майбутню систему, була розроблена діаграма логіки роботи з обліку НДРС. На рис. 3.2 подана логіка роботи системи та напрямки взаємодії користувача з застосунком та БД.

В базі даних зберігається інформація про науково-дослідну роботу студентів. Адміністратор має усі можливості користувача, та ще має змогу через за стосунок взаємодіяти з базою даних користувачів. Видаляти, редагувати та ін. Логіка адміністратора передбачає усю логіку користувача з розширеними можливостями.

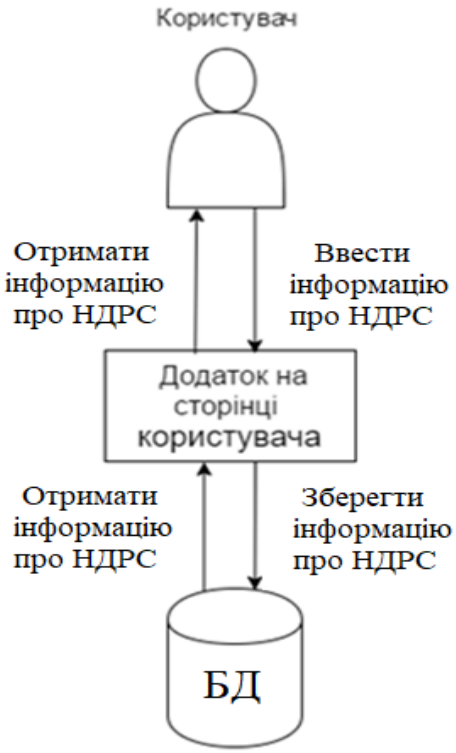

Рисунок 3.2 – Схема логіки роботи системи та зв’язку компонентів для користувача

В програмній системі застосовується багато різних алгоритмів роботи, але майже всі вони виконуються системою чи окремими бібліотеками. Та все ж в програмі є декілька алгоритмів, які мають найбільшу цінність і виділяють систему з ряду інших: авторизація, додавання даних про НДРС д БД, розрахунок кредитів НДРС за заданий період. Розглянемо процеси, якиі є нетривіальними у порівнянні з іншими.

На рис. 3.3 подано діаграму послідовності для виведення інформації по кредитам НДРС за вказаний період.

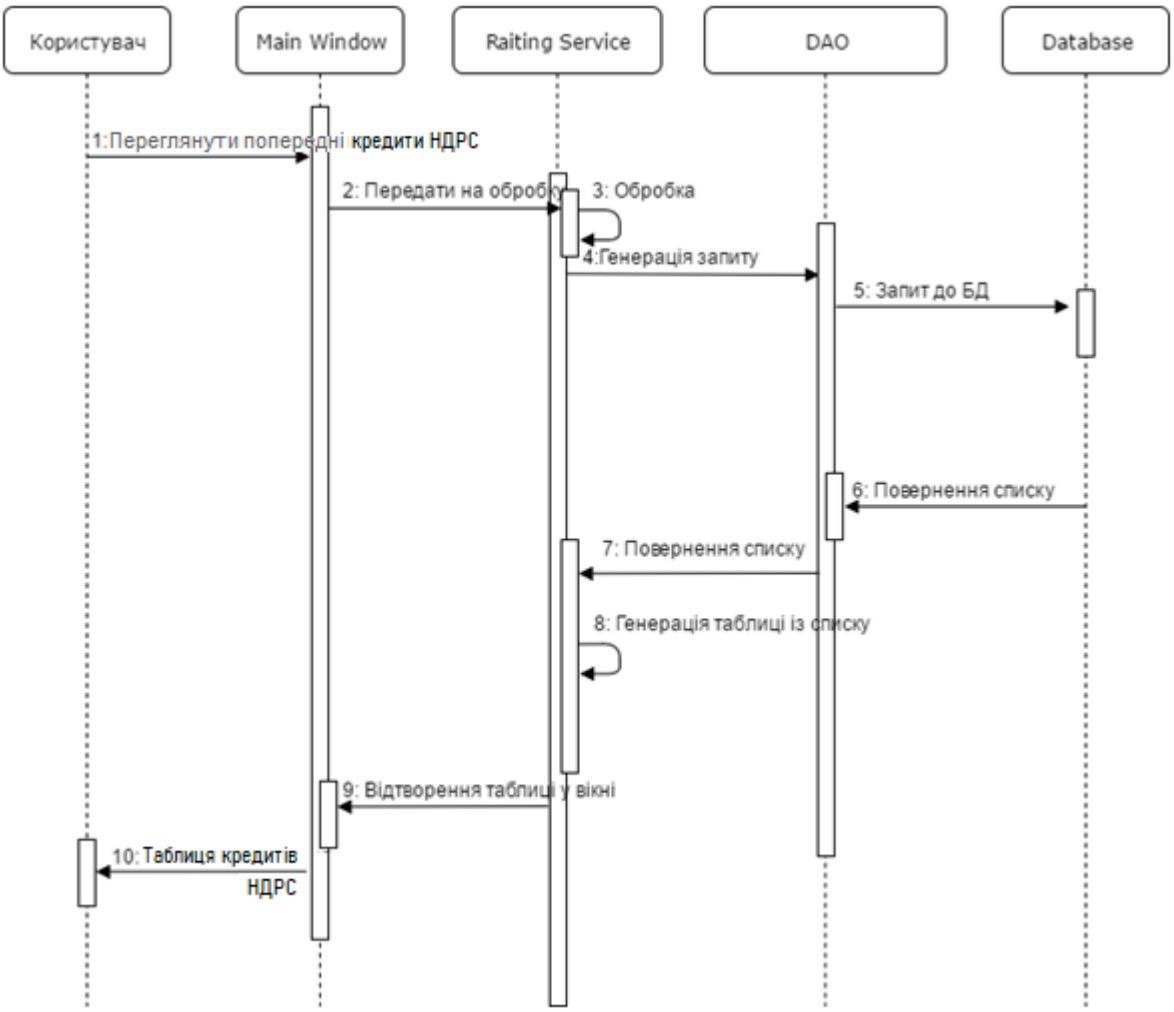

Рисунок 3.3 – Діаграма послідовності для виводу кредитів НДРС за вказаний період

3.4 Побудова моделі потоків даних

Аналіз предметної області проведемо за допомогою CASE-засобу BPwin з використанням двох методів IDEF0 і DFD. Вибір даних методів обумовлений такими факторами:

− IDEF0 – необхідністю визначення відповідних областей в досліджуваній системі, на яких необхідно сфокусувати увагу в першу чергу (обліку НДРС з метою побудови деякої інформаційної системи);

− DFD – дані діаграми використовуються для опису документообігу та

обробки інформації. Вони є доповненням до моделі IDEF0 для більш наочного відображення поточних операцій з документами в системах обробки інформації. Діаграма DFD системи обліку НДРС наведена на рис. 3.3.

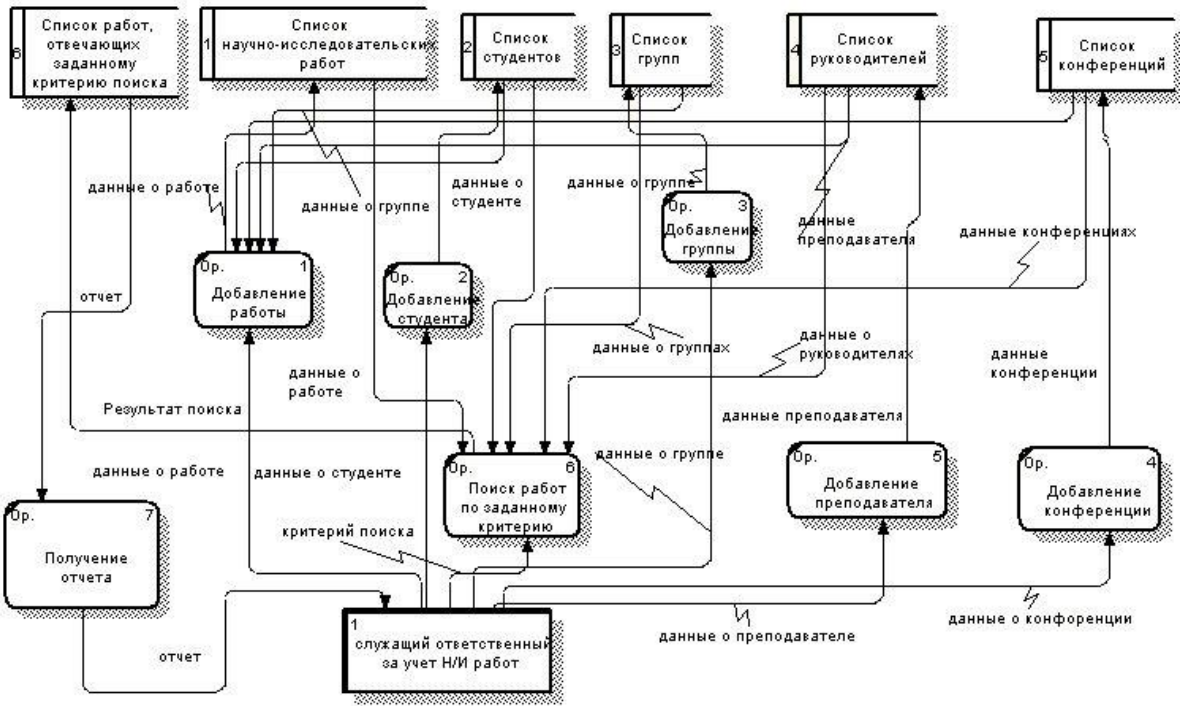

Рисунок 3.3 – Діаграма DFD системи обліку НДРС

Фізична модель даних залежить від конкретної СКБД, фактично будучи відображенням системного каталогу. У фізичної моделі міститься вся інформація про всі об'єкти БД. Представлені в фізичної моделі атрибути несуть конкретну інформацію про конкретних фізичних об'єктах. Фізична база даних буде складатися з таблиць, стовпців і зв'язків. У моделі виділено шість сутностей: Student, Group, State, Teacher, Project, Credit.

Кожна з сутностей має свій набір атрибутів і первинних ключів, які відображені на ER-діаграмі. Таким чином, визначається інформація, яка зберігається в конкретної сутності і в конкретному атрибуті, що забезпечує повну інформаційну підтримку для виконання всіх функцій, закладених в інформаційну систему. Нижче опишемо інформацію, що міститься в кожній з сутностей, наведеною на ERR-діаграмі (рис.3.4).

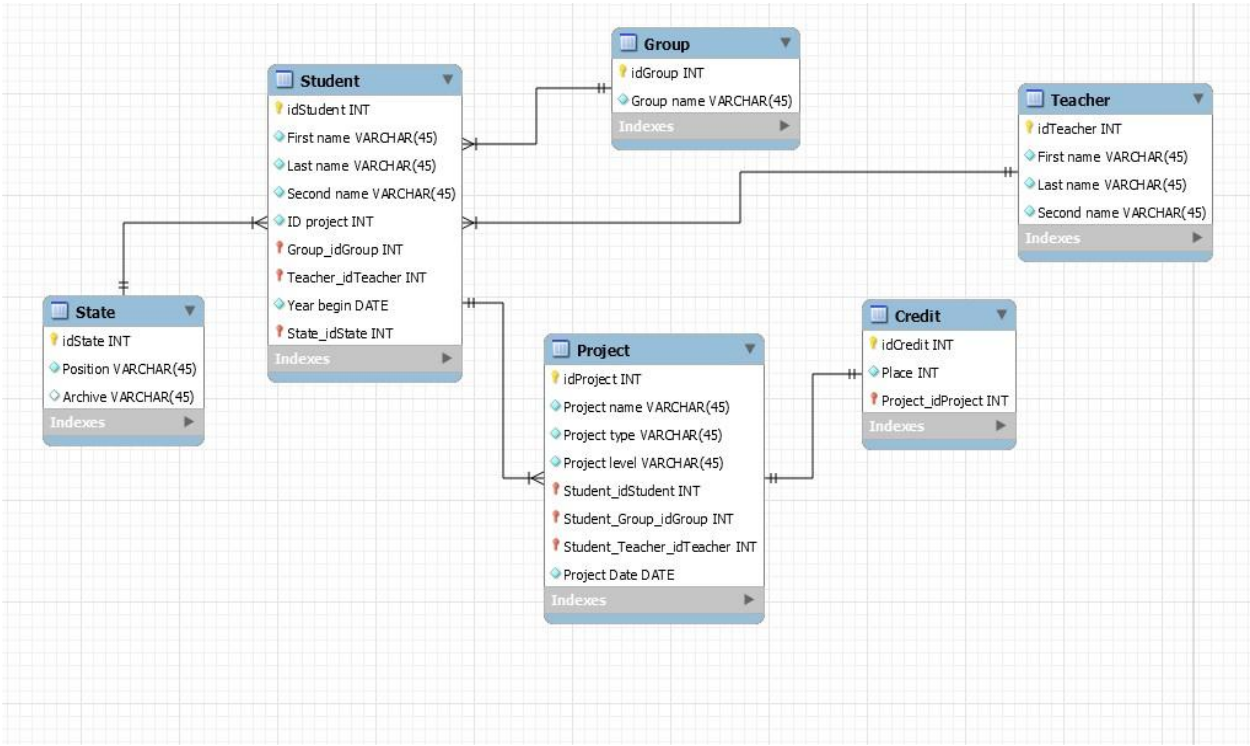

Рисунок 3.4 –ERR діаграма системи

– сутність Student містить інформацію про студента;

– сутність Group зберігає інформацію про групу, в якій студент навчається;

– сутність State містить данні, очно, заочно чи студент закінчив навчання, або його відраховано, потрапив студент до архіву чи ні;

– сутність Teacher зберігає інформацію про вчителя, який керує процесом НДРС;

– сутність Project містить інформацію про НДРС;

– сутність Credit зберігає дані про кредити.

Нижче наводиться детальна структура цих таблиць.

Таблиця 3.1 – Структура таблиці Student

Поле	Тип даних	Примітка
idStudent	int	первинний ключ
First_name	varchar	Ім’я студента
Last_name	varchar	Прізвище

Second_name	varchar	По батькові
Group_idGroup	int	Вторинний ключ(ід групи)
Year_begin	date	Рік початку навчання
Teacher_idTeacher	int	Вторинний ключ(ід вчителя)
State_idState	int	Вторинний ключ(ід позиції)

Таблиця 3.2 – Структура таблиці Project

Поле	Тип даних	Примітка
idProject	int	первинний ключ
Project_name	varchar	Назва роботи
Project_type	varchar	Тип роботи
Project_level	varchar	Рівень роботи
Student_idStudent	int	Вторинний ключ(ід студента що виконує)
Student_Group_idGroup	int	Вторинний ключ(ід групи студента)
Student_Teacher_idTeacher	int	Вторинний ключ(ід вчителя що керує)
Project_Date	date	Дата початку роботи

Таблиця 3.3 – Структура таблиці Credit

Поле	Тип даних	Примітка
idCredit	int	первинний ключ
Place	int	Зайняте місце
Project_idProject	int	Вторинний ключ(ід проекту)

Таблиця 3.4 – Структура таблиці Group

Поле	Тип даних	Примітка
idGroup	int	первинний ключ
Group_name	varchar	Назва групи

Таблиця 3.5 – Структура таблиці Teacher

Поле	Тип даних	Примітка
idTeacher	int	первинний ключ
First_name	varchar	Ім’я вчителя
Last_name	varchar	Прізвище
Second_name	varchar	По батькові

Таблиця 3.6 – Структура таблиці State

Поле	Тип даних	Примітка
idState	int	первинний ключ
Position	varchar	Позиція студента(очна/заочна ф.н.)
Archive	varchar	Скінчив навчання студент чи ні

3.5 Тестування системи обліку НДРС

Система обліку НДРС, що розроблюється повинна забезпечити можливість зручного внесення даних в базу, пошуку студентів в базі даних, і редагування таблиць бази даних.

Для запуску даного програмного продукту необхідна наявність персонального комп'ютера (ПК) з операційною системою, яка підтримує JRE 1.5 і вище, ОП 64 Мбайт і вище.

Після запуску програми користувачеві надається можливість обрати у меню чотири варіанти. Якщо треба проводити роботу з базою даних студентів, або додати чи змінити проекти, роботи, проставити відмітки, треба натиснути першу кнопку (рис.3.5). Якщо треба переглянути архів, де знаходяться всі студенти, котрі закінчили своє навчання у вищому навчальному закладі, натиснувши відповідну кнопку можна попасти до таблиці з усіма студентами. У цій таблиці можна переглянути минулі роботи, і відмітки що заробили студенти. Отже внесені дані не можуть загубитись.

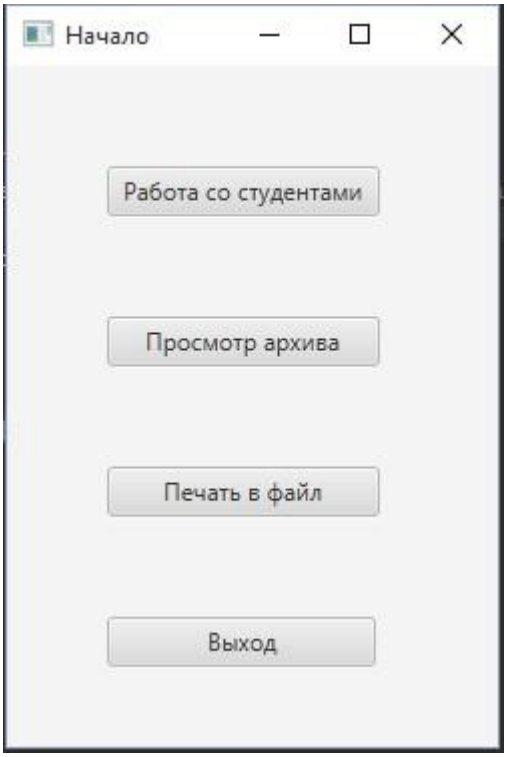

Рисунок 3.5 – Початкове вікно

Наступна кнопка дозволяє вивести дані з таблиць бази даних до файлу більш легшому для редагування форматі Excel. Після обробки й додавання потрібних даних можна роздрукувати.

Остання кнопка дозволяє вийти з програми, якщо вхід був здійснено помилково.

Після того як буде на тиснена перша кнопка, користувач попадає но наступного вікна, в якому представлено інтерфейс що зображено на рис.3.6. Зверху можна побачити три основні кнопки, за допомогою яких:

- додаються до бази даних студенти чи роботи зв’язані безпосередньо зі студентами;
- можливість змінювати інформацію про студентів, якщо це потрібно;
- у цьому ж меню можна проставити оцінку за роботу, виставити місце яке зайняв той чи інший студент;
- перемістити студента до архіву, якщо останній закінчив навчання,

або його відраховано, а анкету потрібно зберігати деякий час.

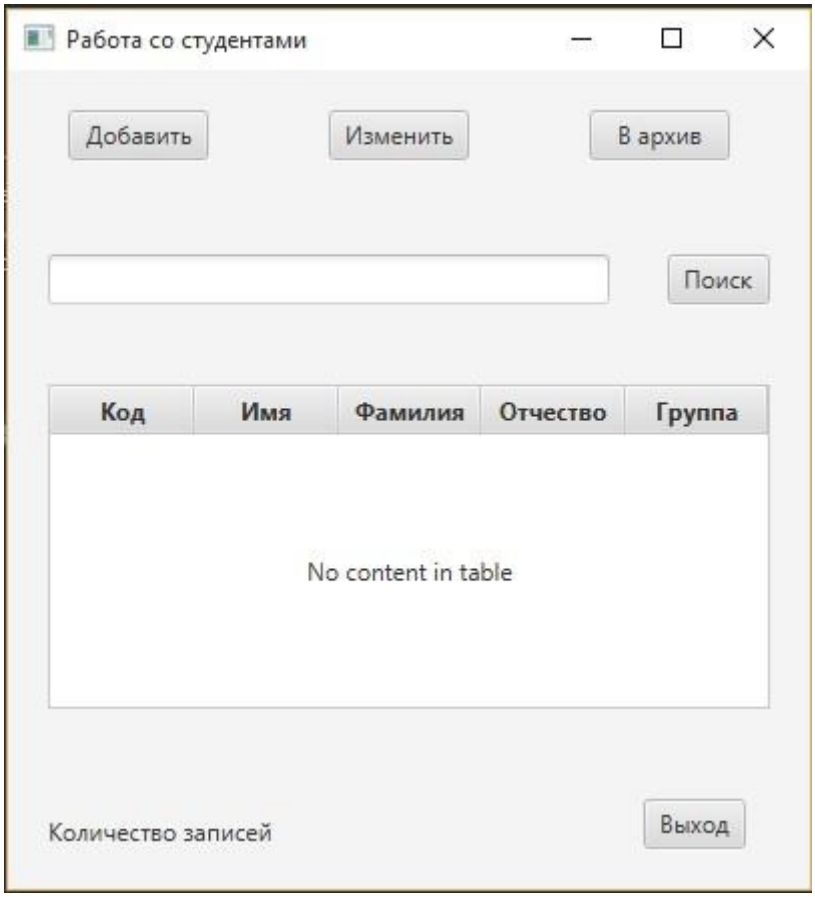

Рисунок 3.6 – Інтерфейс управління НДРС, та студентами

Нижче під кнопками представлено форму пошуку студента по його прізвищу. Отже всі результати пошуку, що відповідають запиту будуть представлені у таблиці нижче. У початку роботи, до того як користувач скористається пошуком в таблиці відображаються всі студенти у базі даних що колись було додано, і які навчаються у поточний час.

Це вікно являє собою головне робоче меню. Якщо робота була закінчена, унизу я кнопка яка дозволяє вийти із середовища.

При натисканні на кнопку «Добавить» (рис.3.6), користувачу представляється вибір: додати нового студента до бази, чи додати новий проект.

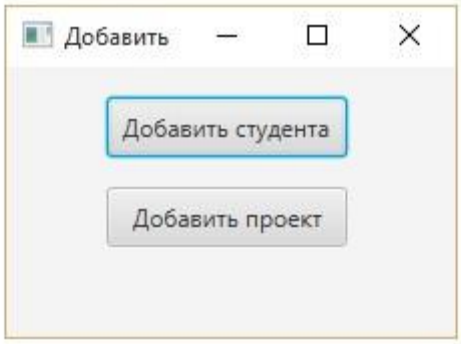

Рисунок 3.6 – Додавання до бази даних

При виборі першого з представленого шляху, користувач може прописати студента що прийматиме участь у науково-дослідницьких роботах. Завести так звану анкету. Після чого цього студента можна буде додавати до участі в проектах, олімпіадах, семінарах, тощо.

При натисканні другої кнопки ми маємо можливість додавати НДРС. Де потрібно буде обрати всі параметри.

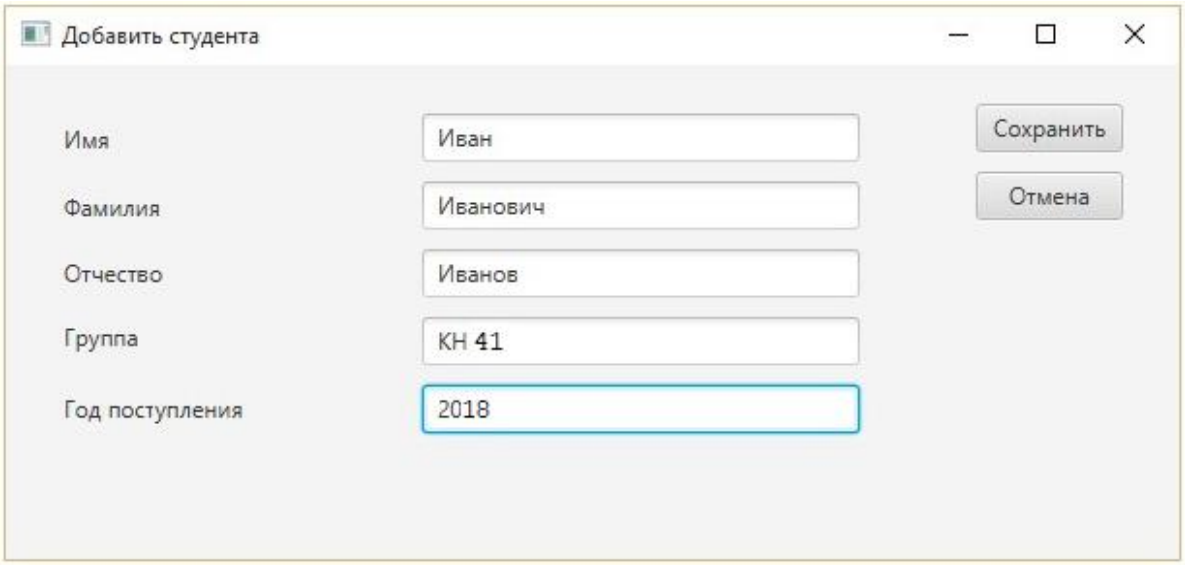

Рисунок 3.7 – Додавання студента до БД

У цьому меню (рис.3.7) треба обов’язково додати усі дані зазначені у

полях, тому що за допомогою цих параметрів зв’язуються дані у базі даних. Інтерфейс дозволяє підтвердити або відмінити зміни. Доки не буде на тиснена кнопка «Сохранить» анкета не потрапить до БД.

Якщо користувач вирішив додати проект, натиснув у відповідному діалозі «Добавить проект» (рис.3.6), потрапляємо до форми описання самого проекту.

Спочатку треба вказати назву. Наступним кроком буде обрати з представленого випадаючого меню елемент НДРС:

- олімпіади;
- конкурс наукових робіт;
- конференції;
- семінари;
- виставки;
- публікації;
- експедиції.

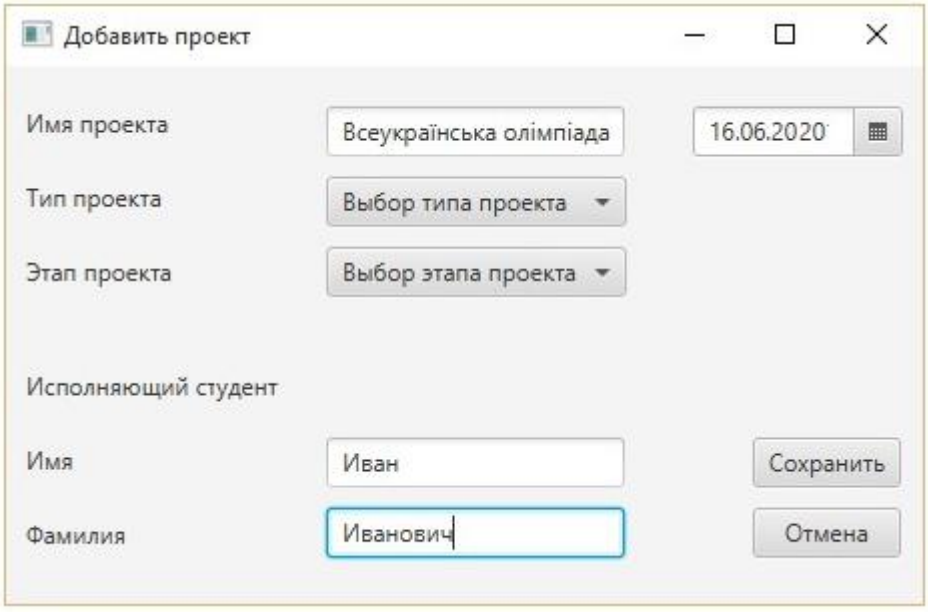

Рисунок 3.8 – Додавання проекту до БД

Також слід обрати етап НДРС. Від цього залежить кількість балів яка буде нарахована студенту після виконання. Обов’язково треба обрати дату,

коли буде виконана робота.

Унизу користувач записує ім’я та прізвище студента, який буде виконувати роботу. Після цього потрібно натиснути «Сохранить», якщо введені параметри дійсні і відповідають задачі. Або «Отмена», якщо треба повернутись у попереднє вікно.

На етапі роботи з початковим меню системи (рис.3.6) у користувача є можливість відредагувати знайденого студента. А також переглянути його анкету і побачити всі НДР в яких він брав участь.

На рис.3.9 показано меню яке втілює ці функції. Після того як натиснута кнопка «Изменить» можна бачимо анкету, де поля з даними студента йдуть «заморожені». Бачити користувач їх може, але поля не редагуються. Для того щоб змінити поля, треба натиснути «Редактировать». Після цього можемо змінити, наприклад, прізвище студентці, якщо в поточний час вона його змінила. Або актуальною проблемою є кожного року у вищому навчальному закладі змінюється шифр групи. Отже за допомогою цього меню це з легкістю можна зробити.

Нижче у таблиці можна побачити всі роботи в котрих брав участь студент. Якщо пройшов час, роботи або олімпіади виконані, фахівець може виставити оцінки які заробив студент. Для цього потрібно у таблиці обрати потрібну графу і натиснути кнопку «Изменить».

Після того як користувач це зробив, він потрапляє до вікна в якому може відредагувати саму роботу, змінити назву або тип, тощо (рис.3.10).

Обов’язково треба вказати зайняте місце (1, 2, 3, «заохочення»). Від цього буде залежати рівень та якість оцінки. Після натиснути «Сохранить».

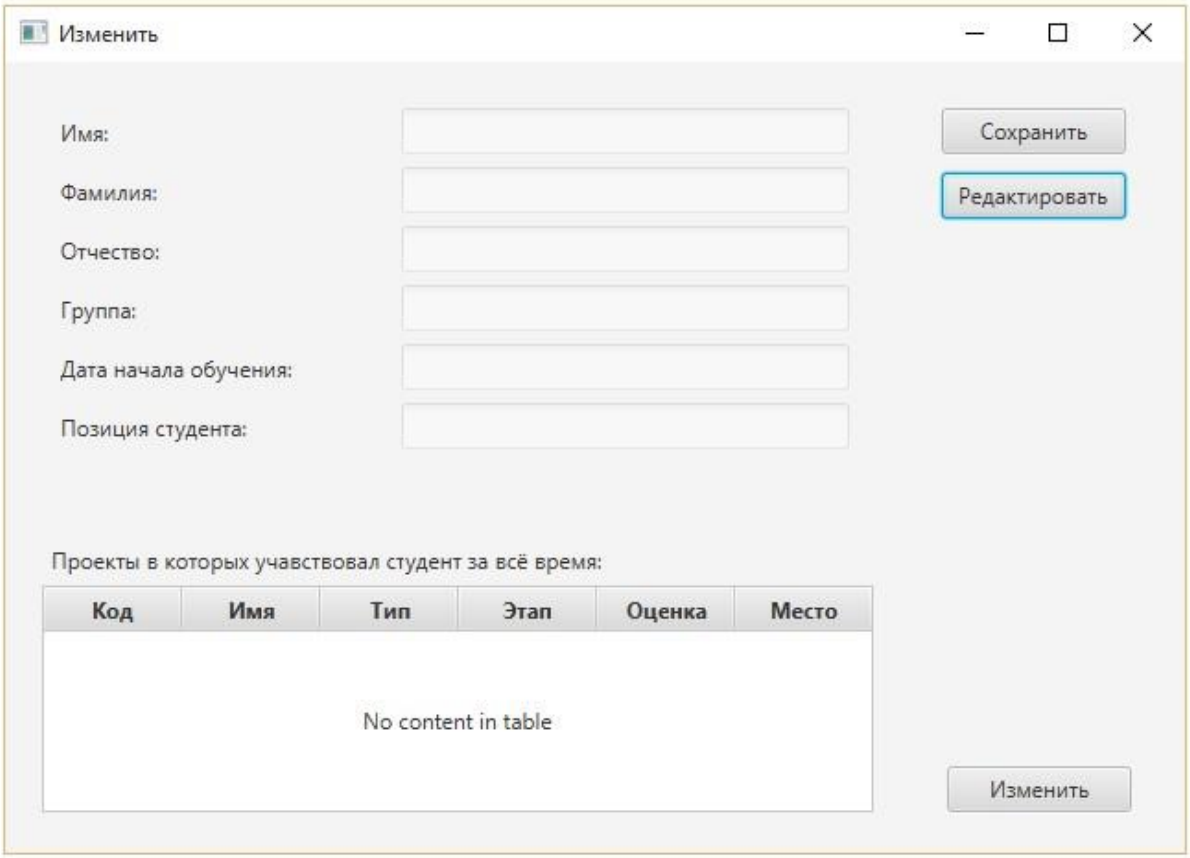

Рисунок 3.9 – Редагування даних

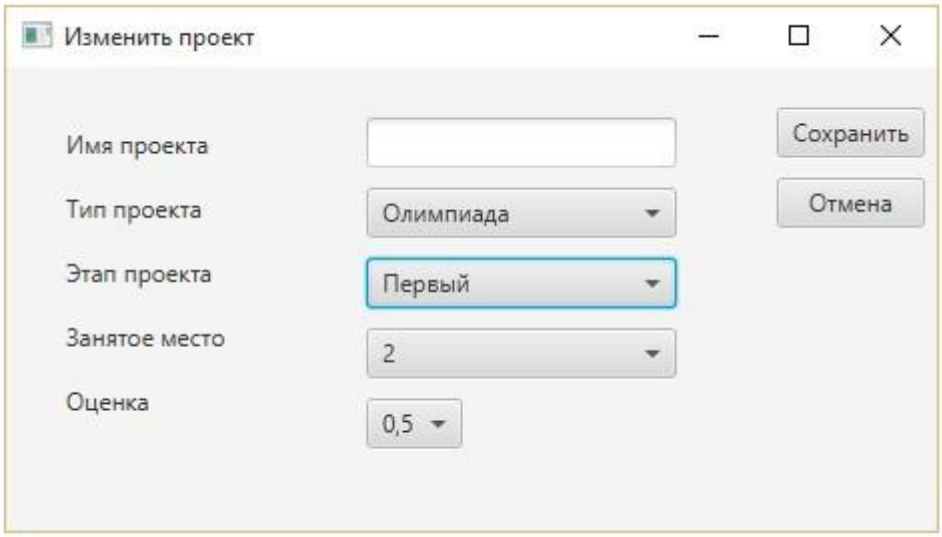

Рисунок 3.10 – Редагування проекту

3.6 Керівництво програміста

Графічній інтерфейс був створений за допомогою нової технології, яка останнім часом набуває популярності JavaFX. Платформа для створення RIA, дозволяє будувати уніфіковані додатки з насиченим графічним інтерфейсом користувача для безпосереднього запуску з-під операційних систем, роботи в браузерах і на мобільних телефонах, в тому числі які працюють з мультимедійним вмістом.

На відміну від технології Swing, JavaFX дозволяє інкапсулювать дані про графічний інтерфейс в окремі fxml файли, написані на мові XML. Це дозволяє розділити логіку програми і графічні елементи. Також, якщо потрібно з легкістю можна підключати CSS таблиці стилів, для кращого керування GUI.

На кожне діалогове вікно до пакету створенному в проекті IDE, та названому «fxml» було створено 7 .fxml файлів, що відповідають за графічний інтерфейс.

Проект збирався за допомогою технології Maven. Отже тут ми налаштовуємо і маємо з’єднання з MySQL сервером.

За допомогою фреймворку Hibernate була налагоджена задача об'єктно-реляційного відображення. На кожну таблицю з БД була створена й описана сутність й розміщена до відповідного класу у пакеті «models».

При проектуванні інформаційної системи виявляються деякі шари, які відповідають за взаємодію різних модулів системи. З'єднання з базою даних є однією з найважливішою складовою програми. Завжди виділяється частина коду, модуль, відповідаючий за передачу запитів в БД і обробку отриманих від неї відповідей. У загальному випадку, визначення Data Access Object описує його як прошарок між БД і системою. DAO абстрагує суті системи і робить їх відображення на БД, визначає загальні методи використання з'єднання, його отримання, закриття та (або) повернення в Connection Pool.

Вершиною ієрархії DAO є абстрактний клас або інтерфейс з описом загальних методів, які будуть використовуватися при взаємодії з базою даних. Як правило, це методи пошуку, видалення по ключу, оновлення і т.д.

Тому у пакеті «DAO» були реалізовані класи-сутності

MainController – клас для зв’язку коду та графічного інтерфейсу.

Main – клас який являється точкою входу программи.

Розробка графічного інтерфейсу відбувалася за допомогою засобу GUI SceneBuilder. GUI SceneBuilder надає можливість використовувати візуальні засоби розробки. Елементи і компоненти можна розташовувати шляхом простого розміщення в потрібних позиціях, вирівнювати положення, змінювати розміри, властивості, назви і так далі. Автоматично генерується код компонентів їх положення і властивостей. Є можливість попереднього перегляду дизайну інтерфейсу до компіляції і запуску.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра була створена система обліку науково-дослідної роботи студентів, яка дозволяє автоматизувати роботу по підрахунку накопичених наукових кредитів студентами за заданий період, а також забезпечує можливість зручного додавання науково-дослідних робіт, пошуку робіт за заданими критеріями, складання звіту за результатами пошуку у табличному вигляді, а також перегляд і редагування таблиць бази даних.

При створенні програмного продукту була докладно вивчена предметна область. Аналіз документообігу та умов обліку НДРС проведений за допомогою CASE-засобів з використанням двох методів IDF0 і DFD.

Побудована логічна і фізична моделі даних БД. Фізична модель бази даних, реалізована для MySQL, яка представляє собою функціонально повну реляционную СКБД. У ній передбачені всі необхідні засоби для визначення і обробки даних, а також для керування ними при роботі з великими обсягами інформації. Одна з основних переваг даної СКБД – безкоштовна ліцензія на версію MySQL Community Edition.

В якості середовище розробки програми обрано середовище IntelliJ IDEA і мова програмування Java, яка є на даний момент одним з найбільш популярних і потужних засобів прискореної розробки додатків. Розробка графічного інтерфейсу відбувалася за допомогою засобу GUI SceneBuilder з використанням компонентів бібліотеки JavaFX.

При створенні програми були враховані всі запропоновані вимоги. Програма має інтуїтивно зрозумілий інтерфейс, що дозволяє швидко заповнити, вибрати, переглянути дані і виконати всі стандартні операції з додавання до БД студентів и НДР.

Тестування підсистеми показало, що вона успішно реалізує поставлені перед нею завдання, тобто скорочує кількість часу на те, щоб отримати дода-

ти запис, здійснити пошук, а також економить час, що витрачається на те, щоб внести зміни в базу даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про порядок нарахування ЄКТС студенту за наукову та науково-технічну діяльність в ОДЕКУ – Офіційний сайт ОДЕКУ. URL: http://odeku.edu.ua/wp-content/uploads/Polozhennya-pro-poryadok-narahuvannya-kreditiv-YEKTS-studentu-za-naukovu-ta-naukovo-tehnichnu-diyalnist-v-ODEKU.pdf (дата звернення 14.04.2021)
2. Автоматизированная система управления ученым заведением. ООО «НПП«МКР». URL: http://mkr.org.ua/modules_ok/index/2/ (дата звернення 14.04.2021)
3. Січко Т.В., Ковальчук О.А. Електронні системи управління вищими навчальними закладами Украхїни. URL: http://econjournal.vsau.org/files/pdfa/2150.pdf (дата звернення 14.04.2021)
4. Науково-дослідний інститут прикладних інформаційних технологій АСУ «ВНЗ». URL: http://ndipit.com.ua/ua/#tab2 (дата звернення 14.04.2021)
5. Електронна система управління навчальним закладом «Сократ». URL: http://www.vsau.vin.ua. (дата звернення 14.04.2021)
6. Програмне забезпечення для вищих навчальних закладів України “ПолітекСОФТ”. URL:http://www.politek-soft.kiev.ua. (дата звернення 14.04.2021)
7. IEEE Spectrum. Рейтинг самих популярних мов програмування. URL: https://learnworthy.net/ieee-spectrum-ranked-the-top-trending-programming-languages/ (дата звернення 14.04.2021)
8. TIOBE Programming Community Index. URL: https://www.tiobe.com/tiobe-index/ (дата звернення 14.04.2021)
9. TIOBE – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/TIOBE#cite_note-3 (дата звернення 14.04.2021)

10. Java – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Java (дата звернення 14.04.2021)
11. Python – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Python (дата звернення 14.04.2021)
12. IntelliJ IDEA – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення 14.04.2021)
13. Підручник з IntelliJ IDEA. URL: https://uk.myservername.com/intellij-idea-tutorial-java-development-with-intellij-ide (дата звернення 14.04.2021)
14. Eclipse – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/Eclipse (дата звернення 14.04.2021)
15. NetBeans – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/ NetBeans (дата звернення 14.04.2021)
16. MySQL – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення 14.04.2021)
17. Microsoft SQL Server – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/ wiki/ Microsoft_SQL_Server (дата звернення 14.04.2021)
18. PostgreSQL – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/PostgreSQL (дата звернення 14.04.2021)
19. Моргунов Е.П. PostgreSQL. Основы языка SQL. СПб.: БХВ-Петербург, 2018. 336 с. ISBN 978-5-9775-4022-3
20. Кравець Р.Б. Інформаційні технології організації бізнесу : навчальний посібник / Р.Б. Кравець, Ю.О. Сєров, О.В. Марковець. Львів: Видавництво Львівської політехніки, 2013. 228 с.
21. Диго С. М. Базы данных: проектирование и использование. Учебник. М.: Финансы и статистика, 2005. 592 с.
22. Верлань А.Ф., Чмырь И.А., Кузниченко С.Д., Коваденко Л.Б. Императивное программирование и объектно-ориентированное модели-

рование: Java, UML, OCL // Учебное пособие для студентов высших учебных заведений. Одесса «Экология», 2013 г.– 432 с.

23. Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Второе издание. ДМК, 2006, 496 с.
24. Буч Г., Якобсон А., Рамбо Д. UML. 2-е издание Классика CS. Питер, 2005, 736 с.

Д О Д А Т К И

ДОДАТОК А

Таблиця нарахування кредитів ЄКТС студентам ОДЕКУ

Таблиця А1 – **Нарахування наукових кредитів ЄКТС студентам ОДЕКУ за видами діяльності**

Вид діяльності	*Кількість кредитів*			
Всеукраїнська олімпіада				
місце	1	2	3	4
І етап	1,0	0,5	0,25	0,10
ІІ етап (заключний)	1,75	1,5	1,25	0,75
Міжнародна олімпіада				
місце	1	2	3	4
Відбірковий етап регіонального рівня	1,0	0,5	0,25	0,10
Відбірковий етап всеукраїнського рівня	1,25	0,75	0,5	0,25
Заключний етап міжнародного рівня	2	1,75	1,5	1
Всеукраїнській конкурс наукових робіт				
місце	1	2	3	участь
І етап	1,25	0,75	0,50	0,25
ІІ етап (заключний)	2	1,75	1,50	1
Примітка: 1, 2, 3 - місця переможців олімпіад, конкурсів; 4 - участь в олімпіадах і конкурсах (для студентів, які набрали не менше 50% від максимальної кількості балів при оцінюванні робіт)				
Наукові гуртки та семінари				
Гурток				0,25
Семінар				0,25
Примітка: за кожну індивідуальну доповідь згідно протоколу засідання гуртка, семінару				
Участь у конференціях				
Університетська				0,25
Всеукраїнська				0,50
Міжнародна				0,75
Примітка: відповідні кредити нараховуються за наявності документів, що підтверджують участь у заході (сертифікат, диплом, грамота, роздруківка титульного аркушу збірника матеріалів та першої сторінки матеріалів, тез доповіді, протокол засідання секції університетської конференції). Якщо матеріали на конференцію представляються декількома студентами, то відповідні кредити поділяються на кількість співавторів				
Наукові публікації				
Статті у зарубіжних наукових виданнях, включених до наукометричної бази Scopus				3,00
Статті у фахових зарубіжних виданнях				2,00
Статті у фахових українських виданнях (включених до переліку ДАК)				1,50
Нефахові видання, збірники статей, матеріалів конференцій				0,25
Примітка: відповідні кредити визначені для всього авторського колективу; для визначення індивідуального внеску – вказані значення поділяються на кількість співавторів				
Участь у виконанні НДР (за навчальний семестр)				
Кафедральна				0,20
за фінансуванням МОН України або госпдоговірній темі				0,25

За міжнародними грантами	0,40
Примітка: кредити нараховуються лише за наявності ПІБ студента в списку співавторів розділів НДР та роботи його не менш 3-х місяців. Списки студентів, яких планується залучити до виконання НДР мають бути подані до відділу наукової роботи студентів на початку семестру або на початку робіт за грантовою та госпдоговірною тематикою.	
Участь у наукових школах, стажуванні за програмами міжнародного освітньо-наукового, наукового співробітництва, академічного обміну студентами	
В Україні	0,50
За кордоном	1
Примітка: відповідні кредити нараховуються за наявності документів, що підтверджують проходження стажування, участь у роботі наукової школи, семінару (сертифікат, диплом)	

ДОДАТОК Б

Фрагмент програмного коду

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" ver-
sion="2.0">

    <persistence-unit name="NewPersistenceUnit">
        <provid-
er>org.hibernate.ejb.HibernatePersistence</provider>
        <class>com.models.StudentEntity</class>
        <class>com.models.CreditEntity</class>
        <class>com.models.GroupEntity</class>
        <class>com.models.ProjectEntity</class>
        <class>com.models.StateEntity</class>
        <class>com.models.TeacherEntity</class>
        <properties>
            <property name="hibernate.connection.url" val-
ue="jdbc:mysql://localhost:3306/db_student"/>
            <property name="hibernate.connection.driver_class"
value="com.mysql.jdbc.Driver"/>
            <property name="hibernate.connection.username" val-
ue="root"/>
            <property name="hibernate.connection.password" val-
ue="1234"/>
            <property name="hibernate.archive.autodetection"
value="class"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>

package com.models;
```

```
import javax.persistence.*;


@Entity
@Table(name = "state")
public class StateEntity {
    private int idState;
    private String position;
    private String archive;

    @Id
    @Column(name = "idState")
    public int getIdState() {
        return idState;
    }

    public void setIdState(int idState) {
        this.idState = idState;
    }

    @Basic
    @Column(name = "Position")
    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    @Basic
    @Column(name = "Archive")
    public String getArchive() {
        return archive;
    }
```

```
    public void setArchive(String archive) {
        this.archive = archive;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;

        StateEntity that = (StateEntity) o;

        if (idState != that.idState) return false;
        if (position != null ? !position.equals(that.position) :
that.position != null) return false;
        if (archive != null ? !archive.equals(that.archive) :
that.archive != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = idState;
        result = 31 * result + (position != null ? posi-
tion.hashCode() : 0);
        result = 31 * result + (archive != null ? ar-
chive.hashCode() : 0);
        return result;
    }
}
package com.models;

import javax.persistence.*;
import java.sql.Date;
```

```
@Entity
@Table(name = "project")
@IdClass(ProjectEntityPK.class)
public class ProjectEntity {
    private int idProject;
    private String projectName;
    private String projectType;
    private String projectLevel;
    private int studentIdStudent;
    private int studentGroupIdGroup;
    private int studentTeacherIdTeacher;
    private Date projectDate;

    @Id
    @Column(name = "idProject")
    public int getIdProject() {
        return idProject;
    }

    public void setIdProject(int idProject) {
        this.idProject = idProject;
    }

    @Basic
    @Column(name = "Project name")
    public String getProjectName() {
        return projectName;
    }

    public void setProjectName(String projectName) {
        this.projectName = projectName;
    }

    @Basic
    @Column(name = "Project type")
```

```
    public String getProjectType() {
        return projectType;
    }

    public void setProjectType(String projectType) {
        this.projectType = projectType;
    }

    @Basic
    @Column(name = "Project level")
    public String getProjectLevel() {
        return projectLevel;
    }

    public void setProjectLevel(String projectLevel) {
        this.projectLevel = projectLevel;
    }

    @Id
    @Column(name = "Student_idStudent")
    public int getStudentIdStudent() {
        return studentIdStudent;
    }

    public void setStudentIdStudent(int studentIdStudent) {
        this.studentIdStudent = studentIdStudent;
    }

    @Id
    @Column(name = "Student_Group_idGroup")
    public int getStudentGroupIdGroup() {
        return studentGroupIdGroup;
    }

    public void setStudentGroupIdGroup(int studentGroupIdGroup)
{
```

```
        this.studentGroupIdGroup = studentGroupIdGroup;
    }

    @Id
    @Column(name = "Student_Teacher_idTeacher")
    public int getStudentTeacherIdTeacher() {
        return studentTeacherIdTeacher;
    }

    public void setStudentTeacherIdTeacher(int
studentTeacherIdTeacher) {
        this.studentTeacherIdTeacher = studentTeacherIdTeacher;
    }

    @Basic
    @Column(name = "Project Date")
    public Date getProjectDate() {
        return projectDate;
    }

    public void setProjectDate(Date projectDate) {
        this.projectDate = projectDate;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;

        ProjectEntity that = (ProjectEntity) o;

        if (idProject != that.idProject) return false;
        if (studentIdStudent != that.studentIdStudent) return
false;
        if (studentGroupIdGroup != that.studentGroupIdGroup) re-
```

```
turn false;
        if (studentTeacherIdTeacher !=
that.studentTeacherIdTeacher) return false;
        if (projectName != null ?
!projectName.equals(that.projectName) : that.projectName !=
null) return false;
        if (projectType != null ?
!projectType.equals(that.projectType) : that.projectType !=
null) return false;
        if (projectLevel != null ?
!projectLevel.equals(that.projectLevel) : that.projectLevel !=
null) return false;
        if (projectDate != null ?
!projectDate.equals(that.projectDate) : that.projectDate !=
null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = idProject;
        result = 31 * result + (projectName != null ?
projectName.hashCode() : 0);
        result = 31 * result + (projectType != null ?
projectType.hashCode() : 0);
        result = 31 * result + (projectLevel != null ?
projectLevel.hashCode() : 0);
        result = 31 * result + studentIdStudent;
        result = 31 * result + studentGroupIdGroup;
        result = 31 * result + studentTeacherIdTeacher;
        result = 31 * result + (projectDate != null ?
projectDate.hashCode() : 0);
        return result;
    }
}
```

```
package com.DAO;

import com.models.StudentEntity;

import javax.persistence.EntityManager;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.util.List;

public class StudentDAO {
    EntityManager entityManager = Persis-
tence.createEntityManagerFactory("NewPersistenceUnit").createEnt
ityManager();
    public void addStudent(StudentEntity studentEntity){
        EntityTransaction transaction =
entityManager.getTransaction();
        transaction.begin();
        entityManager.merge(studentEntity);
      /*  entityManager.persist(studentEntity);*/
        transaction.commit();
    }
    public List<StudentEntity> getAllStudentList(){
        return entityManager.createQuery("SELECT student_alias
FROM StudentEntity  student_alias").getResultList();
    }
}

package com.Service;

import com.DAO.StudentDAO;
import com.models.StudentEntity;

import java.util.List;

public class StudentService {
    private StudentDAO studentDAO;
```

```
    public StudentDAO getStudentDAO() {
        synchronized (StudentService.class) {
            if (studentDAO == null) {
                studentDAO = new StudentDAO();
            }
        }
        return studentDAO;

    }
    public void addStudent(StudentEntity studentEntity){
        getStudentDAO().addStudent(studentEntity);
    }
    public List<StudentEntity> getStudent(){
        return getStudentDAO().getAllStudentList();
    }
}
package com.models;

import javax.persistence.*;


@Entity
@Table(name = "state")
public class StateEntity {
    private int idState;
    private String position;
    private String archive;

    @Id
    @Column(name = "idState")
    public int getIdState() {
        return idState;
    }

    public void setIdState(int idState) {
```

```
        this.idState = idState;
    }

    @Basic
    @Column(name = "Position")
    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    @Basic
    @Column(name = "Archive")
    public String getArchive() {
        return archive;
    }

    public void setArchive(String archive) {
        this.archive = archive;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;

        StateEntity that = (StateEntity) o;

        if (idState != that.idState) return false;
        if (position != null ? !position.equals(that.position) :
that.position != null) return false;
        if (archive != null ? !archive.equals(that.archive) :
that.archive != null) return false;
```

```
            return true;
        }

        @Override
        public int hashCode() {
            int result = idState;
            result = 31 * result + (position != null ? posi-
tion.hashCode() : 0);
            result = 31 * result + (archive != null ? ar-
chive.hashCode() : 0);
            return result;
        }
}
```